Computer Science and Technology

西南科技大学(SouthWest University of Science and Technology)

Experimental Instructor of Computer Operating System

计算机操作系统综合实验指导书

计算机工程系　编写

计算机科学与技术学院

二０一六年

前言

FOREWORD

FOREWORD

FOREWORD

实验课程是理论课程的有力补充，通过操作系统上机实验，培养学生程序设计的方法和技巧，提高学生编制清晰、合理、可读性好的程序的能力，加深对操作系统理论的理解。使学生更好地掌握操作系统的基本概念、基本原理及基本功能，具有分析现代操作系统，设计和开发简单操作系统部分功能的基本能力。对于操作系统理论的掌握与进一步学习具有非常重要的意义。

实验指导书第一章是基础知识概述，主要介绍了实验环境和实验工具。第二章是要求，包括实验选题要求、过程要求、实验报告要求、实验成绩评价和评分标准。第三章是实验内容，计算机操作系统综合实验内容主要围绕计算机操作系统理论课程进行安排，一共包括 10 个实验。实验 1 和实验 2 涉及进程创建和撤销相关理论；实验 3 涉及信号量相关理论；实验 4 和实验 8 涉及进程调度相关理论；实验 5 涉及死锁相关理论；实验 6 涉及页面置换相关理论；实验 7 涉及磁盘调度相关理论；实验 9 涉及内存管理相关理论；实验 10 是一个真实的内核实验，它涉及了可执行文件的装载与重定位。

计算机操作系统综合实验指导书主要帮助学生掌握和理解操作系统原理相关理论，让更抽象的理论在实践中得到更好的理解。适合已经学习过操作系统相关理论的学生使用。

《计算机操作系统综合实验》实验指导书建设小组

2016 年 5 月

目 录

第一章 基础知识概述

1.1 实验环境

硬件环境：PC 机 1 台。

软件环境：Windows2000/xp、Turbo C3.0 或 Microsoft Visual C++ 6.0。

1.2 实验工具

1）Turbo C3.0

该软件是 Borland 公司在 1992 年推出的强大的 C 语言程序设计与 C++面向对象程序设计的集成开发工具。它只需要修改一个设置选项，就能够在同一个 IDE 集成开发环境下设计和编译以标准 C 和 C++语法设计的程序文件。

使用说明：

(1) 找到 TC3.0 安装目录，假设为 C:\TC3；

(2) 打开 Bin 目录，单击 TC 打开 TC3.0 开发环境；

(3) 如图 1-1 所示，可以选择 load 和 new 打开或创建一个新 CPP 文件；

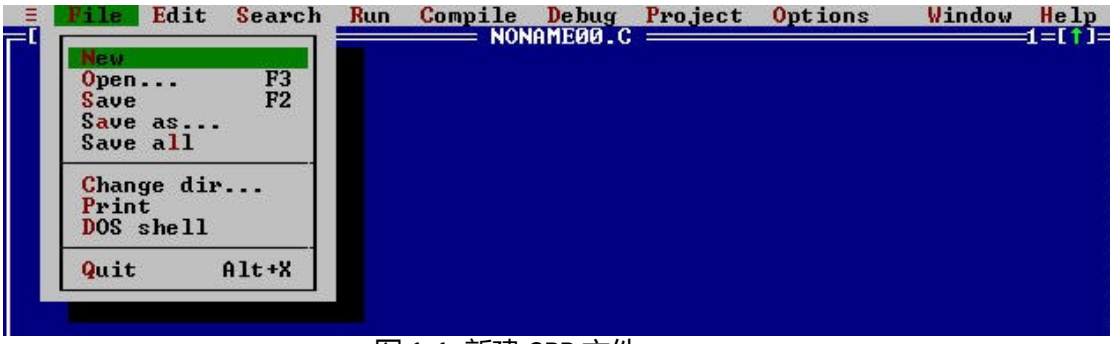

图 1-1 新建 CPP 文件

(4) 对于一个 CPP 文件可以按 F2 或图 1-1 中的 save 菜单项进行保存；

(5) 对于编写好的 CPP 文件按 Alt+F9 进行编译；

(6) 按 Ctrl+F9 运行该程序；

(7) Alt+F5 看运行结果；

(8) Options\Directories 可修改相应默认路径，如 include、输出等，如图 1.2 所示。

图 1-2 TC 路径设置

2）Visual C++6.0

VC6.0 是微软公司开发的面向对象可视化软件制作工具，其提供了强大的 MFC 支持，可以编写基于 Windows 的各类软件，功能强大。在计算机操作系统综合实验过程中，只需

要使用其最基本的编码和编译功能。

使用说明：

(1) 打开 VC6.0 开发环境；

(2) 新建一个 CPP 文件，如图 1-3 所示。

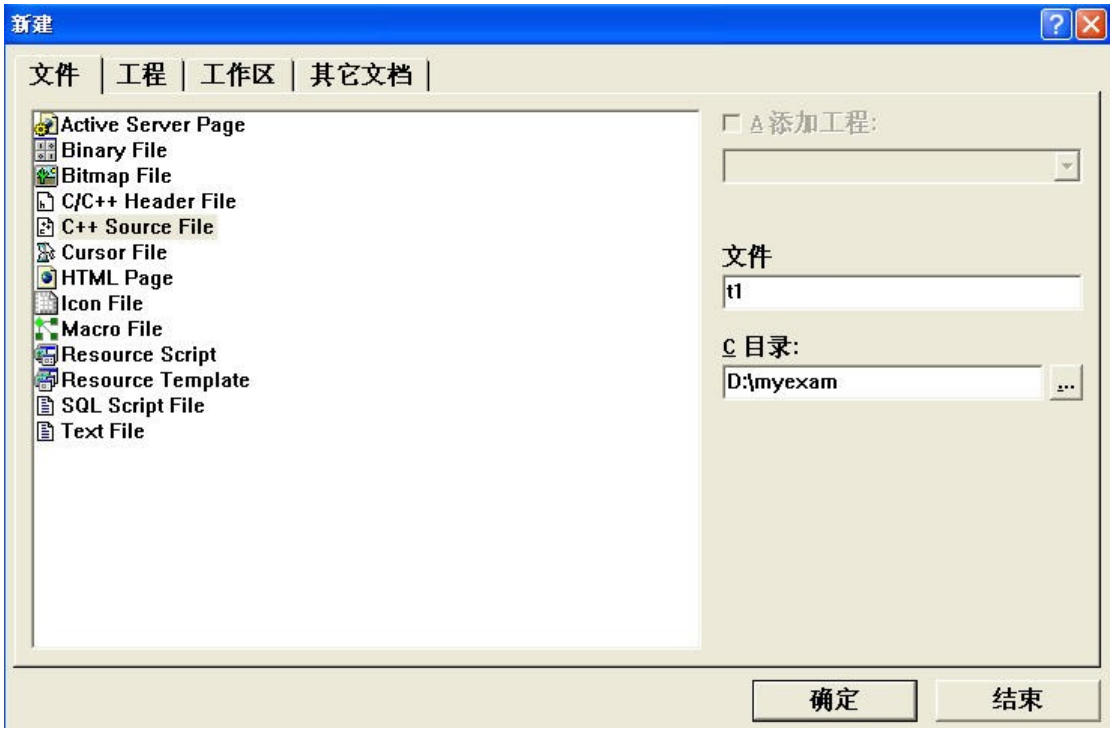

图 1-3 新建一个 CPP 文件

(3) 编写完后按 F5 运行，此时会出现图 1-4 的提示信息，直接按“是”即可运行程序。

图 1-4 单文件运行提示

第二章 实验要求

2.1 选题要求

表 1-1 实验体系结构

序号	实验内容	实验选题	实验类型	实验学时
1	进程创建模拟实现	必做	验证型	2
2	进程撤销模拟实现	必做	设计型	4
3	P、V 原语模拟实现	必做	验证型	2
4	FCFS 进程调度模拟实现	必做	设计型	4
5	银行家算法实现	必做	设计型	4
6	改进型 Clock 页面置换算法实现	必做	设计型	4
7	SCAN 磁盘调度模拟实现	必做	设计型	4
8	基于时间片的高优先级调度实现	必选一题	综合型	8
9	连续动态内存管理实现		综合型	
10	EXE 文件装载实现		综合型	

2.2 实验过程要求

1) 理解实验目的，了解实验内容，掌握相关理论。

2) 实验准备

(1) 验证型实验：掌握实验涉及相关理论，预读指导书提供的相关代码；

(2) 设计型实验：掌握实验涉及相关理论，按照提示算法编写代码；

(3) 综合型实验：掌握实验涉及相关理论，设计算法并编写代码。

3) 实验过程

(1) 验证型实验：按照测试要求进行测试，观察并理解测试结果，完成实验要求和测试要求；

(2) 设计型实验：调试并检查错误，完成实验要求和测试要求；

(3) 综合型实验：调试并检查错误，完成实验要求和测试要求。

4) 实验结果的处理 实验结果正确，达到测试要求。

5) 实验报告的撰写 按照实验报告要求书写实验报告。

6) 实验程序的收集等方面的总体要求 程序运行结果、代码编写或熟悉情况由实验指导教师当场检查，实验代码不用收集。

2.3 实验报告要求

2.3.1 实验报告书写格式

1）封面

封面包括课程名称、实验名称、学号、学生姓名、班级、指导教师、评分、实验日期。

2）主体内容

（1）实验目的 以实验指导书中的实验目的为依据，结合自己的理解情况进行书写。

（2）实验内容 以实验指导书中的实验内容为依据，结合自己的实验完成情况进行书写。

（3）实验要求 实验要求中提出了实验过程要求和实验报告的要求，按照相关要求书写实验报告内容。

（4）实验环境 实验环境包括硬件和软件环境，按照自己实验过程中使用到的环境进行书写。

（5）测试 实验指导书中有测试要求，在实验过程中按照测试要求进行测试，书写相应的测试结果。

（6）实验思考 实验指导书中有实验思考，请仔细理解后查阅资料进行回答并书写。

（7）实验体会 结合实验过程，对整个实验进行总结。

2.3.2 实验报告格式

见附录 B。

2.4 实验成绩评价

2.4.1 实验成绩评价结构及比例

实验成绩评定方式包含实验报告成绩、实验过程成绩两个部分，其中实验过程成绩占60%、实验报告成绩占 40%，如果其中任何一个部分成绩不及格，则总成绩按不及格处理。

2.4.2 考核方式

1）过程成绩考核

（1）验证型实验 根据测试数据量、测试结果理解程度、相关理论知识理解程度进行考核。

（2）设计型实验 根据代码编写规范程度、代码编写正确程度、测试结果正确程度、实验内容完成情况进行考核。

（3）综合型实验 根据算法设计优良程度、代码编写规范程度、测试结果正确程度、实验内容完成情况进行考核。

2）报告成绩考核 根据报告书写规范程度，是否按照实验报告要求进行内容组织，相关流程图是否正确、规范，实验思考回答正确程度，实验体会深刻程度等进行考核。

2.5 评价标准及考核方式细则的确定

期末成绩=实验过程成绩（80%）+实验报告成绩（20%）；实验过程成绩为各实验实践操作成绩的加权平均成绩；实验报告成绩为各实验项目报告成绩的平均值。

2.5.1 过程成绩

实验过程共分为 A、B、C、D、E 五个等级。

1）验证型实验

A：测试数据量充分，测试结果理解好，理论知识清晰； B：测试数据量较充分，较好地理解了测试结果，理论知识较清晰； C：测试数据量适中，测试结果理解程度一般，理论

知识较清晰；

D：测试数据量不够，测试结果理解较肤浅，理论知识不够清晰；

E：测试数据严重不够，无法理解测试结果，理论知识理解混乱。

2）设计型实验

A：代码编写规范、合理，很好地完成了实验内容，测试结果正确； B：代码编写较规范，较好地完成了实验内容，测试结果正确； C：代码编写较规范，完成了实验内容，测试结果基本正确； D：代码编写不够规范，基本完成了实验内容，测试结果基本正确； E：代码编写不够规范，没有完成实验内容。

3）综合型实验

A：算法设计优秀，代码编写准确、规范，测试结果很理想，很好地完成了实验内容；B：算法设计较好，代码编写较准确，测试结果较好，较好地完成了实验内容； C：算法设计较好，代码编写不够紧凑，测试结果较准确，完成了实验内容； D：算法设计一般，代码实现了算法，但不够清晰，测试结果基本正确，基本完成了实

验内容；

E：没有设计出可行的算法，代码编写混乱，测试结果错误，没达到测试要求。

2.5.2 报告成绩

实验报告成绩共分为 A、B、C、D、E 五个等级。占整个实验成绩的20%。

A：报告书写规范、清晰，包含了实验报告要求的所有内容，相关流程图规范、正确，实验思考回答正确、简练，实验体会深刻。

B：报告书写较规范，包含了实验报告要求的所有内容，相关流程图较规范、流程思想较正确，实验思考回答较准确，实验体会较深刻。

C：报告书写较规范，包含了实验报告要求的所有内容，相关流程图完善、基本正确，实验思考回答基本正确，有实验体会。

D：报告书写较规范，包含了实验报告要求的所有内容，相关流程图不够完善，实验思考回答不够准确，有少量实验体会。

E：报告书写不规范，没有包含实验报告要求的所有内容，相关流程图存在较大错误，没有回答实验思考，无实验体会。

A：95　A+：99　A-：90

B：85　B+：89　B-：80

C：75　C+：79　C-：70

D：65　D+：69　D-：60

E：0-59

第三章 实验内容

3.1 进程创建模拟实现

3.1.1 实验类型

验证型（2 学时）。

3.1.2 实验目的

1）理解进程创建相关理论；

2）掌握进程创建方法；

3）掌握进程相关数据结构。

3.1.3 实验描述

本实验针对操作系统中进程创建相关理论进行实验。要求实验者输入实验指导书提供的代码并进行测试。代码简化了进程创建的多个步骤和内容。进程的树形结构采用广义二叉树的方式进行存储。

3.1.4 实验内容

1）输入给定代码；

2）进行功能测试并得出正确结果；

3）分析并掌握测试结果。

3.1.5 实验要求

1）分析进程创建函数 createpc 程序模块；

2）在实验报告中画出 createpc 函数程序流程图；

3）撰写实验报告。

3.1.6 测试要求

1）至少创建 10 个进程；

2）创建进程树中 4 层以上的树型结构。

3.1.7 相关知识

1）进程控制块

为了描述和控制进程的运行，系统为每个进程定义了一个进程控制块（PCB），它是进程实体的一部分，是操作系统管理进程最重要的数据结构。其主要包含四类信息：

(1) 进程标识符

它唯一地标识一个进程。通常包括进程号 pid，父进程号 ppid 和用户号 uid。

(2) 处理机状态

处理器的状态通常由处理机的各种寄存器中的内容组成。PCB 存放中断（阻塞，挂起）时的各寄存器值，当该进程重新执行时，可以从断点处恢复。主要包括：

a) 通用寄存器；

b) 指令计数器；

c) 程序状态字 PSW；

d) 用户栈指针。

(3) 进程调度信息

a) 进程状态；

b) 进程优先级（用于描述优先使用 cpu 级别的一个整数，高优先级的进程先得到cpu，通常情况下，优先值越小优先级越高）；

c) 其它信息（等待时间、总执行时间等）；

d) 事件（等待原因）。

(4) 进程控制信息

a) 程序和数据的地址（程序在内存和外存中的首址）;

b) 进程同步和通信机制；

c) 资源列表（进程除 CPU 以外的所有资源）;

d) 链接指针（进程队列中指向下一个进程的 PCB 首址）。

2) 进程创建流程

(1) 申请空白 PCB

为新进程申请获得唯一的数字标识符，并从 PCB 集合中索取一个空白 PCB。如果无空白 PCB，可以创建一个新的 PCB。在本实验中，每次动态创建 PCB。

(2) 为新进程分配资源 为新进程分配内存空间和栈空间。

(3) 初始化进程控制块

a) 初始化标识信息；

b) 初始化处理机状态信息；

c) 初始化处理机控制信息。

(4) 将新进程插入就绪队列

3) 进程树

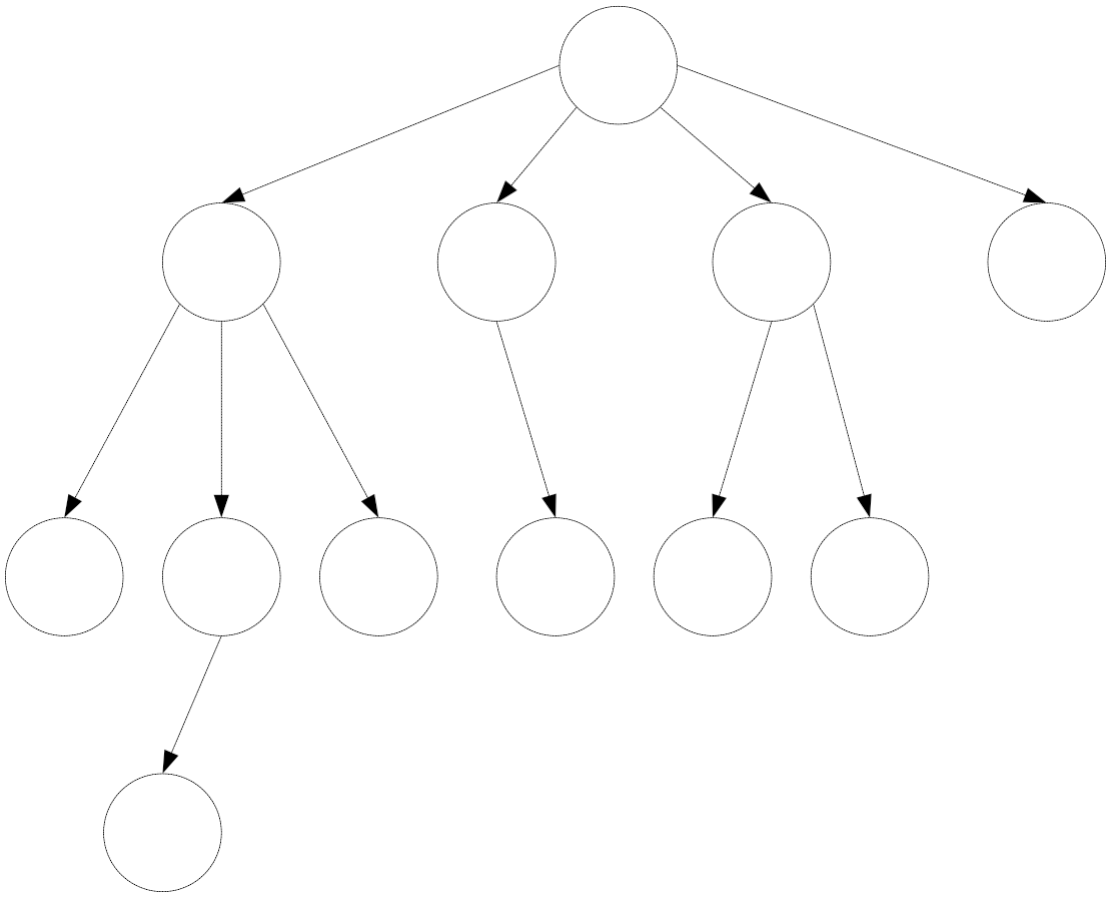

图 3-1 进程树

进程树用于描述进程家族关系，如图 1-1 中可以看出，进程 P1 创建了进程 P2、P3、P4、P5，而 P2 又创建了 P6、P7、P8 …… 。在进程创建过程中，需要对每一个新增加的进程加入到进程树中，有了清晰的父子关系，可以使资源继承或进程删除等操作变得很方便。

4) 进程总链

它是一个 PCB 链表，每一个新创建的进程必须把其 PCB 放入总链中，该总链可以对破坏的进程树进行修复，也方便 PCB 查找。

3.1.8 实验指导

输入实验提供的代码后，可以输入 createpc 命令创建进程，输入 showdetail 显示每个进程及其子进程的信息，测试命令解释如下：

1) createpc 创建进程命令。
   参数： 1 pid（进程 id）、 2 ppid（父进程 id）、3 prio（优先级）。
   示例： createpc(2,1,2) 。创建一个进程，其进程号为 2，父进程号为 1，优先级为 2。
2) showdetail 显示进程信息命令。
3) exit 退出命令行。

3.1.9 实验思考

1) 进程创建的核心内容是什么？
2) 该设计和实际的操作系统进程创建相比，缺少了哪些步骤？

3.2 进程撤销模拟实现

3.2.1 实验类型

设计型（4 学时）。

3.2.2 实验目的

1) 理解进程撤销相关理论；
2) 掌握进程撤销流程。

3.2.3 实验描述

本实验针对操作系统中进程撤销相关理论进行实验。要求实验者设计一个程序，该程序可模拟撤销多个进程及其子孙进程。

3.2.4 实验内容

1) 采用动态或静态方法生成一颗进程树（进程数目≥ 20）；
2) 设计进程撤销算法；
3) 实现进程撤销函数，采用级联方式撤销；
4) 可动态撤销进程；
5) 可动态观察进程树的情况；
6) 测试程序并得到正确结果。

3.2.5 实验要求

1) 编写程序完成实验内容；
2) 在实验报告中画出撤销函数流程图；
3) 撰写实验报告。

3.2.6 测试要求

1) 构建 1 棵进程树，该树至少包含 4 层；
2) 对该树中的多个进程进行删除，至少对一个含有孙子进程的进程进行删除。

3.2.7 相关知识

1)进程创建流程

(1) 从 PCB 链中找到该进程的 PCB，从中读出该进程的状态；
(2) 如果该进程处于执行状态，则终止该进程并置调度标志为真；
(3) 若该进程有子孙进程，需要撤销其子孙进程；
(4) 释放该进程占有的资源；
(5) 从 PCB 链中移出该进程的 PCB。

3.2.8 实验指导

1）进程子树的删除

对于已经创建的进程树（可以参考实验 1 创建进程），在删除的时候，首先需要考虑把该进程及其子孙从整棵树中脱离出来，这样才不会破坏整棵树的完整性。

2）进程总链元素的删除 对于进程树种撤销的所有进程，必须在进程总链中进行删除。

3.2.9 实验思考

1）进程撤销的核心内容是什么？
2）进程总链在进程撤销过程中有什么作用？

3.3 P、V 原语的模拟实现

3.3.1 实验类型

验证型（2 学时）。

3.3.2 实验目的

1) 理解信号量相关理论；
2) 掌握记录型信号量结构；
3) 掌握 P、V 原语实现机制。

3.3.3 实验描述

本实验针对操作系统中信号量相关理论进行实验，要求实验者输入实验指导书提供的代码并进行测试。代码主要模拟信号量的 P 操作（wait）和 V 操作（signal）。

3.3.4 实验内容

1) 输入给定代码；
2) 进行功能测试并得出正确结果。

3.3.5 实验要求

1) 分析 signal 和 wait 函数功能模块；
2) 在实验报告中画出 signal 和 wait 函数流程图；
3) 撰写实验报告。

3.3.6 测试要求

1）至少进行 10 次以上的 wait 和 signal 操作；
2）要求 wait 操作出现进程等待队列；
3）对有等待队列的信号量进行 signal 操作。

3.3.7 相关知识

1) 信号量

信号量也称为信号锁，主要应用于进程间的同步和互斥，在用于互斥时，通常作为资源锁。信号量通常通过两个原子操作 wait(P)和 signal(V)来访问。wait 操作使信号量的值+1，signal 操作使信号量的值-1。

2) 记录型信号量 记录型信号量采用了“让权等待”的策略，存在多个进程等待访问同一临界资源的情况，所以记录型信号量需要一个等待链表来存放等待该信号量的进程控制块或进程号。在本实验 中，使用记录型信号量。

3.3.8 实验指导

实验中提供了 5 个信号量（s0-s4）和 20 个进程（pid 0-19）。在程序运行过程中可以键入 wait 命令，signal 命令和 showdetail 命令显示每个信号量的状态。具体输入解释如下：

1) wait 获取信号量操作（P 操作）。
   参数： 1 sname 2 pid 。
   示例：wait(s1,2) 。进程号为 2 的进程申请名字为 s1 的信号量。
2) signal 释放信号量操作（V 操作）。 参数 1 sname。
   示例：signal(s1)。释放信号量名字为 s1 的信号量。
3) showdetail 显示各信号量状态及其等待队列。
4) exit 退出命令行。

3.3.9 实验思考

1) 如何修改 wait 操作，使之能一次申请多个信号量？

2）在某个时刻，一个进程最多可以等待多少个信号量？

3.4 FCFS 进程调度模拟实现

3.4.1 实验类型

设计型（4 学时）。

3.4.2 实验目的

1）理解进程调度相关理论；

2）掌握 FCFS 进程调度方法。

3.4.3 实验描述

本实验是一个模拟实验，模拟操作系统中进程调度过程。要求实验者设计一个程序，该程序可模拟对 10 个以上的进程进行 FCFS 的方式进行调度。

3.4.4 实验内容

1) 设计可用于该实验的进程控制块，进程控制块至少包括进程号、到达时间和要求服务时间；
2) 动态或静态创建多个（≥ 10）进程；
3) 实现 FCFS 调度算法；
4) 可动态修改进程到达时间；
5) 调度所创建的进程并显示调度结果。

3.4.5 实验要求

1） 编写程序完成实验内容；

2） 在实验报告中画出 FCFS 调度函数流程图；

3） 撰写实验报告。

3.4.6 测试要求

1）要求测试结果包含进程号、到达时间、服务时间、开始执行时间，示例如下：

进程号	到达时间	服务时间	开始执行时间
0	0	3	0
1	1	5	3
2	2	5	8

2）多次修改进程到达时间，进行多次调度并显示结果。

3.4.7 相关知识

1) CPU 调度

一个进程通常需要进行高级、低级调度才能获得 CPU，而一些进程在整个执行周期可能还需要中程调度，在该实验中，我们仅考虑低级调度。

低级调度通常有以下 2 种方式：

非抢占式：采用这种方式时，一旦 CPU 分配给某个进程后，便让该进程一直执行，直至该进程完成或等待某事件而阻塞时，才把 CPU 分配给其他进程。

抢占式：允许调度程序根据某些原则，去暂停某个正在执行的进程，把 CPU 重新分配给另一个进程。抢占的原则有：优先权、时间片等。

2) 先来先服务（**FCFS**）调度算法 每次从就绪队列中选择一个最先进入该队列的进程，为之分配 CPU，使之运行。该进程
一直运行道完成或被某事件阻塞后，才放弃 CPU。

3.4.8 实验指导

1）仔细分析 FCFS 调度算法；

2）参考测试要求并按照输出格式进行输出编码。

3.4.9 实验思考

1) FCFS 调度的优缺点是什么？
2) 短程调度还有哪些调度算法？

3.5 银行家算法实现

3.5.1 实验类型

设计型（4 学时）。

3.5.2 实验目的

1) 理解死锁避免相关内容；
2) 掌握银行家算法主要流程；
3) 掌握安全性检查流程。

3.5.3 实验描述

本实验主要对操作系统中的死锁预防部分的理论进行实验。要求实验者设计一个程序，该程序可对每一次资源申请采用银行家算法进行分配。

3.5.4 实验内容

1) 设计多个资源（≥ 3）；
2) 设计多个进程（≥ 3）；
3) 设计银行家算法相关的数据结构；
4) 动态进行资源申请、分配、安全性检测并给出分配结果。

3.5.5 实验要求

1) 编写程序完成实验内容；
2) 在实验报告中画出银行家和安全性检查函数流程图；
3) 撰写实验报告。

3.5.6 测试要求

1) 进行 Request 请求，输入参数为进程号、资源号和资源数；
2) 进行 3 次以上的 Request 请求；
3) 至少进行 1 次资源数目少于可用资源数，但不安全的请求。

3.5.7 相关知识

1)银行家算法的数据结构

(1) 可利用资源向量 Available。其中每个元素代表每类资源的数目。

(2) 最大需求矩阵 Max。其中每个元素代表每个进程对于每类资源的最大需求量。Max[i，j]=K 表示 i 进程对于 j 类资源的最大需求量为 K。

(3) 分配矩阵 Allocation。其中每个元素代表每个进程已得到的每类资源的数目。

(4) 需求矩阵 Need。其中每个元素代表每个进程还需要的每类资源的数目。

2) 银行家算法

Request［i］[j]=K 表示进程 Pi 需要 K 个 j 类资源。

(1) 如果 Request［i］[j]≤ Need[i，j]，便转向步骤 2，否则认为出错。

(2) 如果 Request［i］[j]≤ Available[j]，便转向步骤 3，否则表示无足够资源，Pi 需等待；

(3) 系统尝试分配资源给 Pi；

(4) 系统进行安全性检查，检查此次资源分配后，系统是否安全。如果安全，则正式分配资源，否则撤销此次分配。

3) 安全性算法

(1) 设置两个向量：工作向量 Work 和 Finish。算法开始时 Work=Available；Finish 表示系统是否有足够的资源分配给进程，使之运行完成，开始时，令 Finish[i]=False；如果有足够的资源分配给进程，则令 Finish[i]=True。

(2) 从进程集合中找到一个能满足下列条件的进程：Finish[i]=False；Need[i，j] ≤

Work[j]，若找到，执行步骤 3），否则，执行步骤 4）；

(3) Pi 获得所需资源后，可顺利执行指导完成，并释放它占有的资源。并执行：

Work[j]=Work[j]+Allocation[i，j]；

Finish[i] = True；

到第 2）步。

(4) 直到所有 Finish[i]=True，表示系统处于安全状态；否则系统处于不安全状态。

3.5.8 实验指导

1）按照相关知识内容仔细理解银行家算法和安全性检查算法；

2）初始化各个数据结构；

3）建立银行家算法涉及的数据结构；

4）建立安全性检查算法涉及的数据结构；

5）实现安全性检查算法函数；

6）实现银行家算法函数；

3.5.9 实验思考

1）针对死锁有哪些可行的解决方案？

2）死锁解除的难点是什么？

3.6 改进型 Clock 页面置换算法实现

3.6.1 实验类型

设计型（4 学时）。

3.6.2 实验目的

1）理解页面置换相关理论；

2）掌握 Clock 置换算法和改进型 Clock 置换算法。

3.6.3 实验描述

本实验主要对操作系统中内存页面置换部分进行实验。要求实验者设计一个程序，该程序采用改进型 Clock 算法对内存页面进行置换，得出置换结果。

3.6.4 实验内容

1）设计页面置换相关数据结构；

2）给一个进程设计多个（≥ 10）页面；

3）设定为每个进程提供的页面数（≤ 5）；

4）可动态修改页面信息（包括调用标志和修改标志）；

5）实现改进型 Clock 页面置换算法；

6）动态给出页面调用序列并进行调度；

7）输出置换结果。

3.6.5 实验要求

1） 编写程序完成实验内容；

2） 在实验报告中画出改进型 Clock 置换函数流程图；

3） 撰写实验报告。

3.6.6 测试要求

1） 修改页面调用标志和修改标志 5 次以上；

2） 进行 3 次以上页面调度。

3.6.7 相关知识

1) 页面置换

在进程运行过程中，若其要访问的页面不在内存而需要调入内存，但分配该进程的内存空间已经用完时，为了保证该进程能继续执行，需要从内存中把该进程的一些页调入外存中的对换区，但调出那个页面，可以采用不同的算法。

2) 改进型 **Clock** 置换算法

由访问位 A 和修改位 M 可以组合成以下四类页面：

1 类（A=0，M=0）：表示该页最近未被访问，也未被修改。

2 类（A=0，M=1）：表示该页最近未被访问，但被修改。

3 类（A=1，M=0）：表示该页最近被访问过，但未被修改。

4 类（A=1，M=1）：表示该页最近被访问过，且被修改过。

改进型 Clock 置换算法如下：

(1) 从指针所在位置开始，扫描循环队列，寻找 A=0 且 M=0 的第一类页面，将遇到的第一个页面作为淘汰页。

(2) 如果 1）失败，开始第二轮扫描，寻找 A=0 且 M=1 的第二类页面，将遇到的第一个页面作为淘汰页，将扫描过的页面 A 置 0。

(3) 如果 2）失败，把指针返回到开始的位置，将所有访问位置 0，到步骤 1）。

3.6.8 实验指导

1）仔细理解相关知识部分的内容；

2）设计页面数据结构，采用位方法进行访问位和修改位的控制；

3）编码实现 A 和 M 的修改函数；

4）编码实现置换算法。

3.6.9 实验思考

1）改进型 Clock 算法和 Clock 算法有何异同点？

2）页面置换还有哪些算法？

3.7 SCAN 磁盘调度模拟实现

3.7.1 实验类型

设计型（4 学时）。

3.7.2 实验目的

1) 理解磁盘调度相关理论；
2) 掌握多种磁盘调度算法；
3) 彻底掌握 SCAN 磁盘调度算法。

3.7.3 实验描述

本实验主要针对操作系统中磁盘调度相关理论进行实验。要求实验者设计一个程序，该程序模拟操作系统的磁盘调度，调度采用 SCAN 算法。

3.7.4 实验内容

1) 设计磁盘调度请求队列（请求数≥ 10）；
2) 设定单位寻道时间；
3) 可动态修改磁盘请求时间和磁道；
4) 实现 SCAN 磁盘调度算法；
5) 实现动态调度并输出调度序列。

3.7.5 实验要求

1）编写程序完成实验内容；
2）在实验报告中画出 SCAN 调度函数流程图；
3）撰写实验报告。

3.7.6 测试要求

1）进行 3 次以上的调度测试；
2）修改 3 次以上开始磁道数据。

3.7.7 相关知识

1) 磁盘调度

磁盘可供多个进程共享，当有多个进程要求访问磁盘时，应采用一种调度算法，以使个进程对磁盘的平均访问时间最小，由于在访问磁盘的时间中，主要是寻道时间，因此磁盘调度的目标就是使磁盘的平均寻道时间最短。

2) SCAN 算法 该算法优先考虑磁头当前的移动方向。如果磁头自里向外运动，则下一个要访问的磁道
为当前磁道之外且距离最近的磁道，直到无更外的磁道时，磁头又改为从外向里运动。

3.7.8 实验指导

1）仔细理解相关知识部分的内容；
2）设计 SCAN 磁盘调度函数。

3.7.9 实验思考

1) SCAN 算法有哪些优缺点？
2) SCAN 与 CSCAN 有什么异同？

3.8 基于时间片的高优先级调度模拟实现

3.8.1 实验类型

综合型（8 学时）。

3.8.2 实验目的

1）理解进程调度相关理论；

2）掌握时间片调度原理；

3）掌握高优先级调度原理。

3.8.3 实验描述

本实验主要针对操作系统中进程调度相关理论进行实验。要求实验者设计一个程序，该程序可以对多个进程进行调度，调度算法采用基于时间片的高优先级调度。

3.8.4 实验内容

1) 设计进程控制块；
2) 设计多个进程队列；
3) 设计多个进程（≥ 20）;
4) 动态生成时间片、执行时间和优先级，将这些信息输出至文件中；
5) 设计基于时间片的多优先级调度算法；
6) 动态调度，并把所有调度信息输出至文件中。

3.8.5 实验要求

1) 编写程序完成实验内容；
2) 在实验报告中画出基于时间片的高优先级调度函数流程图；
3) 撰写实验报告。

3.8.6 测试要求

1) 要求输出每次调度的时间和运行的进程，每次调度结果输出至文件中，文件格式如下：

   0: 进程 2

   6: 进程 8

2) 动态修改时间片 3 次以上。

3.8.7 相关知识

1) 优先级

优先级体现了进程的重要程度或紧迫程度，在大多数现代操作系统中，都采用了优先级调度策略。优先级从小到大（如 0-127），0 优先级最高，127 最低。在本实验中，要求优先级为 0-8。

2) 基于时间片调度 将所有的就绪进程按照先来先服务的原则，排成一个队列，每次调度时，将 cpu 分配给

队首进程，并令其执行一个时间片。当时间片用完时，由一个计时器发出时钟中断请求，调度程序把此进程终止，把该进程放到队尾。

在该实验中，时间片以 100ms 为单位（实际的要小得多）。在调度过程中，需要通过时间函数检测进程的执行时间，当该进程执行时间≥ 时间片大小时，进行调度。

3) 高优先级调度

优先级高的进程优先得到 cpu，等该进程执行完毕后，另外的进程才能执行。

4) 基于时间片的高优先级调度 时间片和优先级调度的结合，在系统中，每个优先级对应一个就绪队列，在每个就绪队

列内，采用时间片调度。当高优先级进程队列调度完成后，才能转入更低优先级的就绪队列

调度。

3.8.8 实验指导

1）设计进程控制块；

2）设计优先级对应的时间片；

3）实现高优先级非抢占式调度算法；

4）实现时间片轮转调度算法；

5）实现基于高优先级的时间片轮转算法。

3.8.9 实验思考

1） 实验过程中的难点是什么？

2） 该调度算法的优点是什么？

3） 和其它调度算法相比，该调度算法还有什么不足？

3.9 连续动态内存管理模拟实现

3.9.1 实验类型

设计型（8 学时）。

3.9.2 实验目的

1） 理解内存管理相关理论；

2） 掌握连续内存管理理论；

3） 掌握动态连续内存管理理论。

3.9.3 实验描述

本实验主要针对操作系统中内存管理相关理论进行实验，要求实验者编写一个程序，该程序管理一块虚拟内存，实现内存分配和回收功能。

3.9.4 实验内容

1）模拟管理 64M 的内存块；

2）设计内存分配函数；

3）设计内存回收函数；

4）实现动态分配和回收操作；

5）可动态显示每个内存块信息。

3.9.5 实验要求

1) 编写程序完成实验内容；

2) 在实验报告中画出内存分配和回收函数流程图；

3) 撰写实验报告。

3.9.6 测试要求

1) 进行 10 次以上内存分配请求；

2) 进行 5 次以上内存释放请求；

3) 在释放过程中必须出现 2 次以上内存连接的情况。

3.9.7 相关知识

连续内存分配：为一个用户程序分配一个连续的内存空间，它分为单一连续分配，固定分区分配和动态分区分配，在本实验中，我们主要讨论动态分区分配。 动态连续分配：根据进程的实际需要，动态地为之分配内存空间。在实现可变分区分配时，将涉及到分区分配中的所用的数据结构、分区分配算法和分区的分配与回收操作这几个问题。

1) 分区分配中的数据结构

(1) 空闲分区表：一张数据表，用于记录每个空闲块的情况，如起始地址、大小，使用情况等。

(2) 空闲分区链：为了实现对空闲分区的分配，把所有的空闲内存块连成一个双向链，便于分配和回收。

2) 分区分配算法

(1) 首次适应算法：从链首出发，寻找满足申请要求的内存块。

(2) 循环首次适应算法：从上次查找的下一个空闲块开始查找，直到找到满足要求的内存块。

(3) 最佳适应算法：在每次查找时，总是要找到既能满足要求又最小的内存块给分配给用户进程。为了方便查找，所有的空闲内存块按从小到大的顺序存放在空闲链表中。

3) 内存分配操作

利用分配算法查找到满足要求的内存块，设请求内存大小为 u.size，而分配的内存块

大小为 m.size，如果 m.size-u.size≤ size （size 为设定的不可再分割的内存大小），则不再切割；反之，按 u.size 分配给申请者，剩余的部分仍留在内存链中。

4) 回收内存 根据回收区地址，从空闲链表中找到相应的插入点。

(1) 回收区与插入点的前一个空闲分区相邻，此时将回收区与前一分区合并，不为回收区分配新表项。

(2) 回收区与插入点的后一个空闲分区相邻，将回收区与后一分区合并成一个新区，回收区的首址最为新分区的首址。

(3) 回收区与前（F1）后(F2)分区相邻，则把三个分区合并成一个大的分区，使 F1 的首址作为新分区的首址，修改 F1 大小，撤销 F2 表项。

(4) 回收区不与任何分区相邻，为回收区建立一个新表项。

3.9.8 实验指导

1）仔细理解相关知识部分的内容；

2）设计内存管理数据结构（链式）;

3）设计内存块数据结构；

4）设计连续内存分配算法；

5）选择一种匹配方式进行分配；

6）完成回收算法。

3.9.9 实验思考

1）连续内存分配和离散内存分配相比有何优缺点？

2）动态连续内存分配的难点是什么？

3.10 EXE 文件装载实现

3.10.1 实验类型

设计型（8 学时）。

3.10.2 实验目的

1) 学会分析 exe 文件头；
2) 理解并掌握文件重定位；
3) 掌握操作系统如何调入可执行文件来运行；
4) 体会操作系统部分核心功能的实现。

3.10.3 实验描述

本实验主要完成 DOS 下的 EXE 文件装载过程，通过分析 EXE 文件头，确定文件的数据区、代码区和堆栈区，对 EXE 文件的各个部分进行装载和重定位。

3.10.4 实验内容

1) 在 DOS 环境下编写一个可显示本人姓名的程序并生成可执行文件；
2) 编写装载程序，把该可执行文件装入内存中。

3.10.5 实验要求

1) 编写程序完成实验内容；
2) 在实验报告中画出装载函数流程图；
3) 撰写实验报告。

3.10.6 测试要求

装载文件 3 次以上，检测并比较运行结果。

3.10.7 相关知识

直接将 EXE 型程序装入内存是不能运行的，因为 EXE 型程序中的地址从 0 开始编号，在实模式下经过重定位后方可运行。

EXE 程序的重定位是根据 EXE 文件头信息进行的。EXE 文件由 2 部分组成：文件头和文件主体。文件头部信息如表 6-1 所示：

表 6-1 EXE 文件头

偏移	说明
00-01	EXE 型程序标志=4D5A
02-03	文件长度 mod512 的余数
04-05	文件扇区数，每扇区 512 字节
06-07	重定位项数
08-09	文件首部长度，以节为单位，每节 16 字节
0a-0b	加载文件所需的最小节数
0c-0d	加载文件所需的最大节数
0e-0f	栈首址，以节为单位，从程序主体开始算起
10-11	SP 值
12-13	文件中所有字的负数相加
14-15	IP 值
16-17	CS 值，以节为单位，从程序主体开始算起
18-19	第一个重定位项的位置，从文件起始单元算起
1a-1b	覆盖号（0 表示程序驻留）
1c-	可变保留区

18h-19h 中存放第一个重定位项的位置，从文件起始单元算起，例如 1E00 表示在文件

的 1E00 位置开始的 4 个字节为程序主体中的第一个重定位项的位置，这 4 个字节若为 07000000，表示从程序主体起始单元算起的 00000007H 位置为第一个重定位项的位置。从文件的 1E00 位置开始的每 4 个字节为一个重定位项的位置。CS 段不用重定位。 注：实验环境采用 DOS 环境，EXE 文件最好在 DOS 环境下编译，在 windows 环境下编译的可 执行文件，头部信息会有所不同。

3.10.8 实验指导

1）仔细理解相关知识部分的内容；
2）查阅关于 DOS PSP 相关知识；
3）使用 C 自带的函数打开文件；
4）分析文件头，确定各段的开始和结束地址；
5）对各段进行分别装载，进行重定位；
6）使用跳转语句执行装载的可执行文件。

3.10.9 实验思考

1) Windows 环境下和 DOS 环境下生成 EXE 文件有何不同？
2) 详细描述装载完毕后，如何运行 EXE 文件？

附录 A 源码

1.实验一源代码

```
#include "basic.h"
pnode *proot;       //system process tree root
pnode *plink;       //system process link head
//create process
int createpc(int *para)
{
    //add your code here
    pnode *p,*p1,*pp;
    int pflag;
    pflag=0;
    for(p=plink;p;p=p->next)
    {
        if(p->node->pid == para[0])    //check if this pid is already exist
        {
            printf("pid %d is already exist!\n",para[0]);
            return -1;
        }
        if(p->node->pid == para[1]) //find parent pcb
        {
            pflag=1;
            pp = p;
        }
    }
    if(!pflag)
    {
        printf("parent id %d is not exist!\n",para[1]);
        return -2;
    }

    //init new pcb
    p1 = new pnode;
    p1->node=new pcb;
    p1->node->pid = para[0];
    p1->node->ppid = para[1];
    p1->node->prio = para[2];
    p1->sub=NULL;
    p1->next=NULL;
    p1->brother=NULL;
```

```
	//add to process tree
	if(!pp->sub)
		pp->sub=p1;
	else
	{
		for(p=pp->sub;p->brother;p=p->brother);
		p->brother=p1;
	}
	// add to process link
	for(p=plink;p->next;p=p->next);
	p->next=p1;
	return 0;
}
//show process detail
void showdetail()
{
	//add your code here
	pnode *p,*p1;
	p=plink;
	for(;p;) //print all pcb info
	{
		printf("%d(prio %d):	",p->node->pid,p->node->prio);
		p1 = p->sub;
		for(;p1;) //print sub pcb
		{
			printf("%d(prio %d)	",p1->node->pid,p1->node->prio);
			p1 = p1->brother;
		}
		printf("\n");
		p = p->next;
	}
	printf("\n");
}

//don't change
void main()
{
	initerror();
	short cflag,pflag;
	char cmdstr[32];
	proot = new pnode;
	proot->node=new pcb;
	proot->node->pid=0;
	proot->node->ppid=-1;
```

```
    proot->node->prio=0;
    proot->next=NULL;
    proot->sub=NULL;
    proot->brother=NULL;
    plink=proot;

    for(;;)
    {
        cflag=0; pflag=0;
        printf("cmd:");
        scanf("%s",cmdstr);
        if(!strcmp(cmdstr,"exit"))    //exit the program
            break;
        if(!strcmp(cmdstr,"showdetail"))
        {
            cflag = 1;
            pflag = 1;
            showdetail();
        }
        else
        {
            int *para;
            char *s,*s1;
            s = strstr(cmdstr,"createpc"); //create process
            if(s)
            {
                cflag=1;
                para = (int *)malloc(3);
                //getparameter
                s1 = substr(s,instr(s,'(')+1,strlen(s)-2);    //get param string
                para=strtoarray(s1);                                    //get parameter
                createpc(para);    //create process
                pflag=1;
            }
        }

        if(!cflag)
            geterror(0);
        else if(!pflag)
            geterror(1);
    }
}
```

2.实验二源代码

```
#include "basic.h"

semphore sem[5];
pnode * pr[20];
//wait operation
void wait(char * sname,int pid)
{
	int fflag,pflag;
	pnode *p,*p1;
	semphore *s;
	fflag=0;
	pflag=0;
	for(int i=0;i<5;i++)
		if(!strcmp(sem[i].name,sname))//find semaphore by name
		{
			s=&sem[i];
			fflag=1;
			break;
		}
	for(i=0;i<20;i++)	//find pcb by pid
		if(pr[i]->node->pid == pid)
		{
			p1 = pr[i];
			pflag=1;
			break;
		}
	if(!fflag)	//semaphore is not exist
	{
		printf("the semphore '%s' is not exist!\n",sname);
		return;
	}
	if(!pflag)	//pid is not exist
	{
		printf("the process '%d' is not exist!\n",pid);
		return;
	}
	s->count--;	//semaphore’s value -1
	if(s->count>=0)	//this pcb get the semaphore
		s->curpid = p1->node->pid;
	else
	{
		if(s->wlist)	//the link is not NULL, add the pcb to the last
```

```
            {
                for(p=s->wlist;p->next;p=p->next);
                p->next=p1;
            }
            Else    //this pcb is the first pcb be added to the wait list
                s->wlist=p1;
        }
}

//signal operation
void signal(char *sname)
{
        int fflag=0;
        for(int i=0;i<5;i++)
            if(!strcmp(sem[i].name,sname)) //find the semaphore by name
            {
                fflag=1;
                break;
            }
        if(fflag)   //find it
        {
            sem[i].count++;
            if(sem[i].wlist)    //there are processes in the wait list
            {
                sem[i].curpid = sem[i].wlist->node->pid;
                sem[i].wlist = sem[i].wlist->next;
            }
        }
        else
            printf("the semphore '%s' is not exist!\n",sname);
}
//show semphore infomation
void showdetail()
{
        int i; pnode *p;
        printf("\n");
        for(i=0;i<5;i++)
        {
            if(sem[i].count<=0)
            {
                printf("%s (curp    %d):    ",sem[i].name,sem[i].curpid);
                p=sem[i].wlist;
                while(p)
```

```
                {
                    printf("%5d",p->node->pid);
                    p=p->next;
                }
            }
            else
                printf("%s :    ",sem[i].name);
            printf("\n");
        }
}

/*****************************************************************************/
/*    don't change                                                           */
void init()
{
    //init semaphore
    strcat(sem[0].name,"s0");
    strcat(sem[1].name,"s1");
    strcat(sem[2].name,"s2");
    strcat(sem[3].name,"s3");
    strcat(sem[4].name,"s4");
    for(int i=0;i<5;i++)
    {
        sem[i].wlist=NULL;
        sem[i].count=1;
    }
    //init process
    for(i=0;i<20;i++)
    {
        pr[i] = new pnode;
        pr[i]->node=new pcb;
        pr[i]->node->pid=i;
        pr[i]->brother=NULL;
        pr[i]->next=NULL;
        pr[i]->sub=NULL;
    }
}

void main()
{
    short  cflag,pflag;
    char   cmdstr[32];
    char *s,*s1,*s2;
```

```
initerror();
init();

for(;;)
{
    cflag=0; pflag=0;
    printf("cmd:");
    scanf("%s",cmdstr);
    if(!strcmp(cmdstr,"exit"))    //exit the program
        break;
    if(!strcmp(cmdstr,"showdetail"))
    {
        cflag = 1;
        pflag = 1;
        showdetail();
    }
    else
    {
        s = strstr(cmdstr,"wait"); //create process
        if(s)
        {
            cflag=1;
            //getparameter
            s1 = substr(s,instr(s,'(')+1,instr(s,',')-1);
            s2 = substr(s,instr(s,',')+1,instr(s,')')-1);
            if(s1 && s2)
            {
                wait(s1,atoi(s2));
                pflag=1;
            }
        }
        else
        {
            s=strstr(cmdstr,"signal");//delete process
            if(s)
            {
                cflag=1;
                s1 = substr(s,instr(s,'(')+1,instr(s,')')-1);
                if(s1)
                {
                    signal(s1);
                    pflag=1;
                }
```

```
                    }
                }
            }
            if(!cflag)
                geterror(0);
            else if(!pflag)
                geterror(1);
        }
}
/*************************************************************************************/
```

3.basic.h 文件

```
#ifndef basic_h
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define basic_h

char *errormsg[256];

//process control block
struct pcb
{
	int pid;			//process id
	int ppid;			//parent process id
	int prio;			//priority
	int state;			//state
	int lasttime;		//last execute time
	int tottime;		//totle execute time
};

//process node
struct pnode
{
	pcb *node;
	pnode *sub;
	pnode *brother;
	pnode *next;
};

//信号量
struct semphore{
	char name[5];	//名称
	int count;		//计数值
	int curpid;		//当前进程 id
	pnode *wlist;	//等待链表
};

#define geterror(eno)	printf("%s\n",errormsg[eno])

void initerror()
{
	errormsg[0] = (char *)malloc(20);
	errormsg[0]="error command!";
```

```
    errormsg[1] = (char *)malloc(20);
    errormsg[1]="error parameter!";
}

//get a substring in string s
char * substr(char *s,int start,int end)
{
    char *s1;
    int len = strlen(s);
    if(start<0 || end>=len || start>end)
        return NULL;
    s1=(char *)malloc(end-start+2);
    for(int i=0;i<=end-start;i++)
        s1[i] = s[i+start];
    s1[i]='\0';
    return s1;
}

//find the location of c in string s
int instr(char *s,char c)
{
    int i;
    for(i=0;i<strlen(s);i++)
        if(s[i]==c)
            return i;
    return -1;
}
//change the string to array data
int * strtoarray(char *s)
{
    int *a,count,i,x1;
    char c, *s1,*s2;
    if(!s)
    {
        printf("string can't be null!\n");
        return NULL;
    }
    count=0; s1=s;
    for(i=0;i<strlen(s1);i++)
        if(s1[i]==',')
            count++;
    count++;
    a = (int *)malloc(count);
```

```
    c=',';
    for(i=0;i<count;i++)
    {
        x1 = instr(s1,c);
        if(x1>=0)
            s2=substr(s1,0,x1-1);
        else
            s2=s1;
        a[i]=atoi(s2);
        if(c==',')
            s1=substr(s1,x1+1,strlen(s1)-1);
    }
    return a;
}
#endif
```

附录 **B** 实验报告格式

西南科技大学

计算机实验报告

课 程 名 称：_____

实 验 名 称：_____

学　　号：_____

学 生 姓 名：_____

班　　级：_____

指 导 教 师：_____

评　　分：_____

实 验 日 期：　　年　　月　　日

1、实验目的

2、实验内容

3、实验要求

4、实验环境

5、测试

6、实验思考

7、实验体会

(School of Computer Science and Technology)

Experimental Instructor

计算机科学与技术学院自编实验指导书

软件项目管理
计算机基本技能训练
络
原理
序语言设计
图形学
理
软件测试技术
编语言程序设计
网络程序设计
反病毒技术
Linux 系统及程序设计
Windows 程序设计
软件开发综合实验

Java 程序设计
网络攻防对抗 计算机网
计算机制图 计算机组成
微机原理及应用 C++程
嵌入式系统实验 计算机
数据库应用设计 编译原
单片机与接口应用设计
信息电器分析与设计 汇
数据结构应用设计
计算机操作系统综合实验
动态网页制作技术
数学实验
面向对象技术应用设计
信息安全综合实验