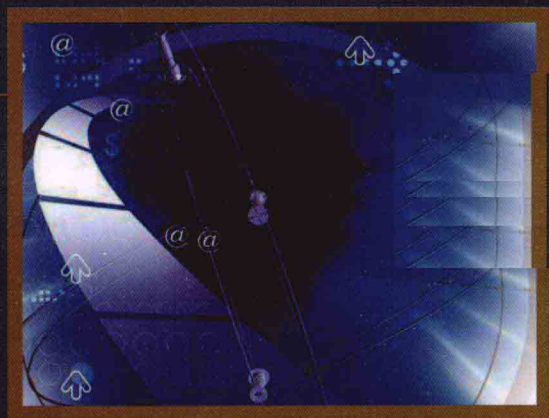




高等学校计算机科学与技术教材

操作系统实验教程



□ 袁宝华 主编
□ 曹红根 毛平 马瑾 刘健 副主编

- 原理与技术的完美结合
- 教学与科研的最新成果
- 语言精炼，实例丰富
- 可操作性强，实用性突出



清华大学出版社

● 北京交通大学出版社

高等学校计算机科学与技术教材

操作系统实验教程

袁宝华 主 编

曹红根 毛 平 副主编
马 瑾 刘 健

北京交通大学出版社

操作系统实验教程

高等学校计算机科学与技术教材

ISBN 958-7-81133

1. ① ② ③

图版本图前 CIP

2889

http://www.tup.com.cn

2414

http://www.tup.com.cn

http://www.tup.com.cn

电话: 010-82328899

http://www.tup.com.cn

电话: 010-2168414

北京交通大学出版社

北京交通大学出版社

182×260 印张: 13.75 字数: 411千字

2010年1月第1次印刷

清华大学出版社

北京交通大学出版社

· 北京 ·

联系电话: 010-2168414, 010-2168408, 传真: 010-2168410, E-mail: press@bjtu.edu.cn

内 容 简 介

操作系统课程的实验环节一直是操作系统教学的难点。通过实验可以加深学生对操作系统工作原理的领会和对操作系统实现方法的理解,并且可以使学生在程序设计方面得到最基本的训练。本书的实验环境基于 Windows 操作系统和 Linux 操作系统;对于进程管理、进程通信、进程调度、银行家算法、存储器管理、设备管理等基本实验,本书都提供了两个操作系统环境下的实验示例代码。另外本书还对 Linux 部分内核代码进行了源码分析。

本书可作为应用型本科、高职高专操作系统实验教学教材,也可以作为从事相关专业人员学习和研究的指导用书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13501256678 13801310933

图书在版编目(CIP)数据

操作系统实验教程/袁宝华主编. —北京:清华大学出版社;北京交通大学出版社, 2009.12

(高等学校计算机科学与技术教材)

ISBN 978-7-81123-952-2

I. ①操… II. ①袁… III. ①操作系统-教材 IV. ①TP316

中国版本图书馆 CIP 数据核字(2009)第 214637 号

责任编辑:杨正泽

出版发行:清华大学出版社 邮编:100084 电话:010-62776969 <http://www.tup.com.cn>

北京交通大学出版社 邮编:100044 电话:010-51686414 <http://press.bjtu.edu.cn>

印刷者:北京瑞达方舟印务有限公司

经 销:全国新华书店

开 本:185×260 印张:12.75 字数:331 千字

版 次:2010 年 1 月第 1 版 2010 年 1 月第 1 次印刷

书 号:ISBN 978-7-81123-952-2/TP·558

印 数:1~4 000 册 定价:21.00 元

本书如有质量问题,请向北京交通大学出版社质监组反映。对您的意见和批评,我们表示欢迎和感谢。

投诉电话:010-51686043, 51686008; 传真:010-62225406; E-mail: press@bjtu.edu.cn。

前 言

操作系统是计算机系统中的核心软件。操作系统教学不但需要讲授操作系统原理与方法,还需要让学生动手进行操作系统编程实践,只有这样才能够让学生真正理解操作系统的精髓。

根据多年的教学实践并吸取国内外操作系统方面的有关内容,编写了这本《操作系统实验教程》。其目的是使学生通过实验,理解和掌握操作系统的基本原理,提高编写和开发系统程序的能力。

本书编写过程中,我们采用循序渐进的方式,对实验内容尽量做到具有独立性,并且对每个实验中用到的知识给出了相关的介绍,这样有利于读者通过自学掌握实验教程中的技术和方法。本书分为3篇:第1篇介绍基于Windows环境下的进程管理和进程调度、银行家算法、虚拟存储器管理、设备管理、SPOOLING技术、文件系统、操作系统接口共8个实验;第2篇介绍基于Linux环境下的进程管理和进程调度、进程间通信、虚拟存储器管理、字符型设备驱动程序、Linux文件系统调用、Shell程序共7个实验;第3篇附录介绍了Linux操作系统的安装、Linux中C语言编译器GCC的使用、Linux中C语言调试器GDB的使用和Visual C++集成开发环境。书中给出的程序示例都已在Windows下Visual C++ 6.0和Linux Red Hat 9.0中经过测试和验证。本书第1、2、9、10章及附录B、C、D由毛平编写,第3、4、11、12、15章及附录A由曹红根编写,第5、6、7、8、13、14章由袁宝华和马瑾编写。

由于编者水平有限,书中难免有不足之处,敬请读者提出宝贵意见。

编 者

2009年12月

目 录

第 1 篇 Windows 系统实验指导

第 1 章 进程管理	3
1.1 实验目的	3
1.2 实验内容	3
1.3 实验准备	3
1.3.1 进程	3
1.3.2 进程控制块	4
1.3.3 进程的创建与撤销	6
1.3.4 进程的阻塞与唤醒	7
1.4 程序示例	7
1.5 运行结果	11
第 2 章 进程调度	14
2.1 实验目的	14
2.2 实验内容	14
2.3 实验准备	14
2.4 程序示例	15
2.5 实验结果	22
第 3 章 银行家算法	27
3.1 实验目的	27
3.2 实验内容	27
3.3 实验准备	28
3.4 程序示例	29
3.5 实验结果	33
第 4 章 虚拟存储器管理	34
4.1 实验目的	34
4.2 实验内容	34
4.3 实验准备	35
4.4 程序示例	36
4.5 实验结果	41
第 5 章 设备管理	43

5.1	实验目的	43
5.2	实验内容	43
5.3	实验准备	43
5.4	程序示例	44
5.5	实验结果	48
第 6 章	SPOOLING 技术	50
6.1	实验目的	50
6.2	实验内容	50
6.3	实验准备	50
6.4	程序示例	52
6.5	实验结果	55
第 7 章	文件系统	58
7.1	实验目的	58
7.2	实验内容	58
7.3	实验准备	59
7.4	程序示例	59
7.5	实验结果	81
第 8 章	操作系统接口	83
8.1	实验目的	83
8.2	实验内容	83
8.3	实验准备	84
8.4	程序示例	86
8.5	实验结果	91

第 2 篇 Linux 系统实验指导

第 9 章	进程管理	95
9.1	实验目的	95
9.2	实验内容	95
9.3	实验准备	95
9.3.1	进程	95
9.3.2	所涉及的系统调用	96
9.4	程序示例	99
9.5	实验结果	100
第 10 章	进程调度	102
10.1	实验目的	102
10.2	实验内容	102
10.3	实验准备	102
10.4	程序示例	104

第 11 章 进程间通信	108
11.1 实验目的	108
11.2 实验内容	108
11.3 实验准备	108
11.4 程序示例	111
11.5 实验结果	114
第 12 章 虚拟存储器管理	116
12.1 实验目的	116
12.2 实验内容	116
12.3 实验准备	117
12.4 程序示例	118
12.5 实验结果	124
第 13 章 字符型设备驱动程序	126
13.1 实验目的	126
13.2 实验内容	126
13.3 实验准备	126
13.4 程序示例	128
13.5 实验结果	130
第 14 章 Linux 文件系统调用	132
14.1 实验目的	132
14.2 实验内容	132
14.3 实验准备	132
14.4 程序示例	133
14.5 实验结果	135
第 15 章 Shell 程序	138
15.1 实验目的	138
15.2 实验内容	138
15.3 实验准备	138
15.4 程序示例	151
15.5 实验结果	152

第 3 篇 附 录

附录 A Linux 操作系统的安装	155
A1 实验目的	155
A2 实验内容	155
A3 实验准备	155
附录 B Linux 中 C 语言编译器 GCC 的使用	174
B1 实验目的	174

80B2	实验内容	174
80B3	实验准备	174
80B4	程序示例	177
80B5	实验结果	178
附录 C	Linux 中 C 语言调试器 GDB 的使用	179
81C1	实验目的	179
81C2	实验内容	179
81C3	实验准备	179
81C4	程序示例	182
81C5	实验结果	183
附录 D	Visual C++ 集成开发环境	186
参考文献		195

82I	附录 E	195
82I	附录 F	195
82I	附录 G	195
82I	附录 H	195
82I	附录 I	195
82I	附录 J	195
82I	附录 K	195
82I	附录 L	195
82I	附录 M	195
82I	附录 N	195
82I	附录 O	195
82I	附录 P	195
82I	附录 Q	195
82I	附录 R	195
82I	附录 S	195
82I	附录 T	195
82I	附录 U	195
82I	附录 V	195
82I	附录 W	195
82I	附录 X	195
82I	附录 Y	195
82I	附录 Z	195

附录 E

82I	附录 E	195
82I	附录 F	195
82I	附录 G	195
82I	附录 H	195
82I	附录 I	195
82I	附录 J	195
82I	附录 K	195
82I	附录 L	195
82I	附录 M	195
82I	附录 N	195
82I	附录 O	195
82I	附录 P	195
82I	附录 Q	195
82I	附录 R	195
82I	附录 S	195
82I	附录 T	195
82I	附录 U	195
82I	附录 V	195
82I	附录 W	195
82I	附录 X	195
82I	附录 Y	195
82I	附录 Z	195

第 1 篇

Windows 系统实验指导

- 第 1 章 进程管理
- 第 2 章 进程调度
- 第 3 章 银行家算法
- 第 4 章 虚拟存储器管理
- 第 5 章 设备管理
- 第 6 章 SPOOLING 技术
- 第 7 章 文件系统
- 第 8 章 操作系统接口



Windows 95 系統安裝與組態

第一章 系統安裝	1
第二章 硬體安裝	2
第三章 網路安裝	3
第四章 網路設備安裝	4
第五章 網路管理	5
第六章 SPOLING 技術	6
第七章 文件系統	7
第八章 系統管理	8

第1章

进程管理

1.1 实验目的

① 理解进程的概念，明确进程和程序的区别。

② 理解并发执行的实质。

③ 掌握进程的创建、睡眠、撤销等进程控制方法。

1.2 实验内容

用C语言编写程序，模拟实现创建新的进程；查看运行进程；换出某个进程；杀死运行进程等功能。

1.3 实验准备

1.3.1 进程

1. 进程的含义

进程是程序在一个数据集合上的运行过程，是系统资源分配和调度的一个独立单位。一个程序在不同数据集合上运行，乃至一个程序在同样数据集合上的多次运行都是不同的进程。

2. 进程的状态

通常情况下，一个进程必须具有就绪、执行和阻塞3种基本状态。

(1) 就绪状态

当进程已分配到除处理器（CPU）以外的所有必要资源后，只要再获得处理器就可以立即执行，这时进程的状态称为就绪状态。在一个系统里，可以有多个进程同时处于就绪状

态,通常把这些就绪进程排成一个或多个队列,称为就绪队列。

(2) 执行状态

处于就绪状态的进程一旦获得了处理器,就可以运行,进程状态也就处于执行状态。在单处理器系统中,只能有一个进程处于执行状态,在多处理器系统中,则可能有多个进程处于执行状态。

(3) 阻塞状态

正在执行的进程因为发生某些事件(如请求输入输出、申请额外空间等)而暂停运行,这种受阻暂停的状态称为阻塞状态,也可以称为等待状态。通常将处于阻塞状态的进程排成一个队列,称为阻塞队列。在有些系统中,也会按阻塞原因的不同将处于阻塞状态的进程排成多个队列。

除了进程的3种基本状态外,在很多系统中为了更好地描述进程的状态变化,又增加了两种状态。

(1) 新状态

当一个新进程刚刚建立,还未将其放入就绪队列的状态,称为新状态。

(2) 终止状态

当一个进程已经正常结束或异常结束时,操作系统已将其从系统队列中移出,但尚未撤销,这时称为终止状态。

3. 进程状态之间的转换

进程状态转换示意图如图1-1所示。

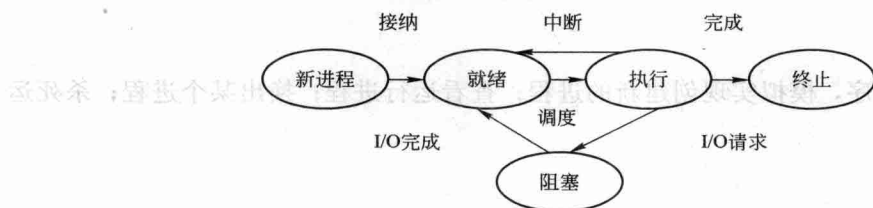


图1-1 进程状态转换示意图

1.3.2 进程控制块

1. 进程控制块的作用

进程控制块是构成进程实体的重要组成部分,是操作系统中最重要的记录型数据,在进程控制块PCB中记录了操作系统所需要的用于描述进程情况及控制进程运行所需要的全部信息。通过PCB,能够使得原来不能独立运行的程序(数据)成为一个可以独立运行的基本单位,一个能够并发执行的进程。换句话说,在进程的整个生命周期中,操作系统都要通过进程的PCB来对并发执行的进程进行管理和控制,进程控制块是系统对进程控制采用的数据结构,系统是根据进程的PCB而感知进程是否存在。所以,进程控制块是进程存在的唯一标志。当系统创建一个新进程时,就要为它建立一个PCB;进程结束时,系统又回收其PCB,进程也随之消亡。

2. 进程控制块的内容

进程控制块主要包括下述4个方面的信息。

1) 进程标识信息

进程标识符用于标识一个进程,通常有外部标识符和内部标识符两种。

(1) 外部标识符

由进程创建者命名,通常是由字母、数字所组成的一个字符串,在用户(进程)访问该进程时使用。外部标识符都便于记忆,如计算进程、打印进程、发送进程、接收进程等。

(2) 内部标识符

是为方便系统使用而设置的,操作系统为每一个进程赋予唯一的一个整数,作为内部标识符。它通常是一个进程的序号。

2) 说明信息(进程调度信息)

说明信息是有关进程状态等一些与进程调度有关的信息,包括以下几个方面。

(1) 进程状态

指明进程当前的状态,作为进程调度和对换时的依据。

(2) 进程优先权

用于描述进程使用处理器的优先级别,通常是一个整数,优先权高的进程将可以优先获得处理器。

(3) 进程调度所需的其他信息

其内容与所采用的进程调度算法有关,如进程等待时间、进程已执行时间等。

(4) 阻塞事件

是指进程由执行状态转变为阻塞状态时等待所发生的事件,即阻塞原因。

3) 现场信息(处理器状态信息)

现场信息是用于保留进程存放在处理器中的各种信息。主要由处理器内的各个寄存器的内容组成。尤其是当执行中的进程暂停时,这些寄存器内的信息将被保存在PCB里,当该进程获得资源重新执行时,能从上次停止的地方继续执行。

(1) 通用寄存器

其中的内容可以被用户程序访问,用于暂存信息。

(2) 指令计数器

用于存放要访问的下一条指令的地址。

(3) 程序状态字

用于保存当前处理器状态的信息,如执行方式、中断屏蔽标志等。

(4) 用户栈指针

每个用户进程都有一个或若干个与之相关的关系栈,用于存放过程和系统调用参数及调用地址,栈指针指向堆栈的栈顶。

4) 管理信息(进程控制信息)

(1) 程序和数据的地址

它是指该进程的程序和代码所在的主存和外存地址,以便该进程再次执行时,能够找到程序和代码。

(2) 进程同步和通信机制

它是指实现进程同步和进程通信时所采用的机制,如消息队列、指针、信号量等。

(3) 资源清单

该清单中存放有除了 CPU 以外,进程所需的全部资源和已经分配到的资源。

(4) 链接指针

它将指向该进程所在队列的下一个进程的 PCB 的首地址。

3. 进程控制块的组织方式

在一个系统中,通常拥有数十个、数百个乃至数千个 PCB,为了能对它们进行有效的管理,就必须通过适当的方式将它们组织起来,目前常用的组织方式有链接方式和索引方式两种。

1) 链接方式

把具有相同状态的 PCB,用链接指针链接成队列,如就绪队列、阻塞队列和空闲队列等。就绪队列中的 PCB 将按照相应的进程调度算法进行排序。而阻塞队列也可以根据阻塞原因的不同,将处于阻塞状态的进程的 PCB,排成等待 I/O 队列、等待主存队列等多个队列。此外,系统主存的 PCB 区中空闲的空间将排成空闲队列,以方便进行 PCB 的分配与回收。

2) 索引方式

系统根据各个进程的状态,建立不同索引表,例如就绪索引表、阻塞索引表等。并把各个索引表在主存的首地址记录在主存中的专用单元里,也可以称为表指针。在每个索引表的表目中,记录着具有相同状态的各个 PCB 在表中的地址。

4. 进程控制原语

原语是指具有特定功能的不可被中断的过程。它主要用于实现操作系统的一些专门控制操作。用于进程控制的原语有以下几种。

1) 创建原语

用于为一个进程分配工作区和建立 PCB,置该进程为就绪状态。

2) 撤销原语

用于一个进程工作完后,收回它的工作区和 PCB。

3) 阻塞原语

用于进程在运行过程中发生等待事件时,把进程的状态改为等待状态。

4) 唤醒原语

用于当进程等待的事件结束时,把进程的状态改为就绪状态。

1.3.3 进程的创建与撤销

1. 进程的创建

一旦操作系统发现了要求创建进程的事件后,便调用进程创建原语按下列步骤创建一个新进程。

① 为新进程分配唯一的进程标识符,并从 PCB 队列中申请一个空闲 PCB。

② 为新进程的程序和数据 and 用户栈分配相应的主存空间及其他必要资源。

③ 初始化 PCB 中的相应信息,如标识信息、处理器信息、进程控制信息等。

④ 如果就绪队列可以接纳新进程,便将新进程加入到就绪队列中。

2. 进程的撤销

一旦操作系统发现了要求终止进程的事件后,便调用进程终止原语按下列步骤终止指定的进程。

- ① 根据被终止进程的标识符,从PCB集合中检索该进程的PCB,读出进程状态。
- ② 若该进程处于执行状态,则立即终止该进程的执行。
- ③ 若该进程有子孙进程,还要将其子孙进程终止。
- ④ 将该进程所占用的资源回收,归还给其父进程或操作系统。
- ⑤ 将被终止进程的PCB从所在队列中移出,并撤销该进程的PCB。

1.3.4 进程的阻塞与唤醒

1. 进程的阻塞

一旦操作系统发现了要求阻塞进程的事件后,便调用进程阻塞原语,按下列步骤阻塞指定的进程。

- ① 立即停止执行该进程。
- ② 修改进程控制块中的相关信息。把进程控制块中的运行状态由“执行”状态改为“阻塞”状态,并填入等待的原因,以及进程的各种状态信息。
- ③ 把进程控制块插入到阻塞队列。根据阻塞队列的组织方式插入到阻塞队列中。
- ④ 待调度程序重新调度,运行就绪队列中的其他进程。

2. 进程的唤醒

一旦操作系统发现了要求唤醒进程的事件后,便调用进程唤醒原语,按下列步骤唤醒指定的进程。

- ① 从阻塞队列中找到该进程。
- ② 修改该进程控制块的相关内容。把阻塞状态改为就绪状态,删除等待原因等。
- ③ 把进程控制块插入到就绪队列中。按照就绪队列的组织方式,把被唤醒的进程的进程控制块插入到就绪队列中。

1.4 程序示例

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct jincheng_type{
    int pid;
    int youxian;
    int daxiao;
    int zhuangtai;//标志进程状态,0为不在内存,1为在内存,3为挂起
```

```
char info[10];
};
struct jincheng_type neicun[20];
int shumu=0, guaqi=0, pid, flag=0;
void create() {
    if (shumu>=20) printf("\n 内存已满, 请先换出或杀死进程\n");
    else {
        for (int i=0; i<20; i++)
            //定位, 找到可以还未创建的进程
            if (neicun[i]. zhuangtai==0) break;
        printf("\n 请输入新进程 pid\n");
        scanf("%d", &(neicun[i]. pid));
        for (int j=0; j<i; j++)
            if (neicun[i]. pid==neicun[j]. pid) {
                printf("\n 该进程已存在\n");
                return;
            }
        printf("\n 请输入新进程优先级\n");
        scanf("%d", &(neicun[i]. youxian));
        printf("\n 请输入新进程大小\n");
        scanf("%d", &(neicun[i]. daxiao));
        printf("\n 请输入新进程内容\n");
        scanf("%s", &(neicun[i]. info));
        //创建进程, 使标记位为 1
        neicun[i]. zhuangtai=1;
        shumu++;
    }
}

void run() {
    for (int i=0; i<20; i++) {
        if (neicun[i]. zhuangtai==1) {
            //输出运行进程的各个属性值
            printf("\n pid=%d", neicun[i]. pid);
            printf("youxian=%d", neicun[i]. youxian);
            printf("daxiao=%d", neicun[i]. daxiao);
            printf("zhuangtai=%d", neicun[i]. zhuangtai);
            printf("info=%s", neicun[i]. info);
            flag=1;
        }
    }
    if (!flag) printf("\n 当前没有运行进程\n");
}
```

```
void huanchu(){
    if(!shumu){
        printf("当前没有运行进程\n");
        return;
    }
    printf("\n 输入换出进程的 ID 值");
    scanf("%d",&pid);
    for(int i=0;i<20;i++){
        //定位,找到所要换出的进程,根据其状态做相应处理
        if(pid==neicun[i].pid){
            if(neicun[i].zhuangtai==1){
                neicun[i].zhuangtai=2;
                guaqi++;
                printf("\n 已经成功换出进程\n");
            }
            else if(neicun[i].zhuangtai==0)printf("\n 要换出的进程不存在\n");
            else printf("\n 要换出的进程已被挂起\n");
            flag=1;
            break;
        }
    }
    //找不到,则说明进程不存在
    if(flag==0)printf("\n 要换出的进程不存在\n");
}

void kill(){
    if(!shumu){
        printf("当前没有运行进程\n");
        return;
    }
    printf("\n 输入杀死进程的 ID 值");
    scanf("%d",&pid);
    for(int i=0;i<20;i++){
        //定位,找到所要杀死的进程,根据其状态做相应处理
        if(pid==neicun[i].pid){
            if(neicun[i].zhuangtai==1){
                neicun[i].zhuangtai=0;
                shumu--;
                printf("\n 已成功杀死进程\n");
            }
            else if(neicun[i].zhuangtai==0) printf("\n 要杀死的进程不存在\n");
            else printf("\n 要杀死的进程已被挂起\n");
            flag=1;
        }
    }
}
```