21 世纪计算机及相关专业课程学习辅导系列

操作系统原理

学习指导与题解

主编　何炎祥

编者　何炎祥　朱骁峰

华中科技大学出版社

内 容 简 介

本书是为计算机及相关专业学生编写的“操作系统原理”课程的学习指导用书。本书是作者根据多年教学经验并参考多种《操作系统原理》教材编写而成的，旨在帮助读者加深对“操作系统原理”课程知识要点的理解与掌握，提高分析问题和解决问题的能力。

全书分为 13 章，分别是操作系统概述、进程描述与控制、并发控制、死锁处理、内存管理、处理机调度、I/O 设备管理、文件管理、分布计算、分布式进程管理、操作系统安全性和操作系统设计原则。各章分为重点与难点、例题解答和自测练习三部分，内容涵盖了整个《操作系统原理》教材的知识面和主要内容，最后还给出了 5 套模拟试题及解答供练习之用。

本书可与华中科技大学出版社出版的“面向 21 世纪计算机专业本科系列教材”中的《操作系统原理》配套，可作为高等院校计算机专业及相关专业本、专科师生的“操作系统原理”课程的教学参考书，也可供参加自学考试、硕士研究生入学考试的各类人员及计算机应用技术人员参考。

前　言

“操作系统原理”是高等院校计算机及其相关专业的一门重要的主干课程。该课程的主要目的是使学生了解并掌握计算机操作系统的原理、结构以及基本实现方法。计算机操作系统是非常重要的系统软件，学好这门课程有助于提高对计算机系统的了解，并为以后进行较深层次的软件研制与开发打下坚实的基础。

本书是为计算机及其相关专业学生编写的操作系统课程的学习指导用书。编者参考了当前各种版本的操作系统教材，结合多年的教学经验，对操作系统的知识要点进行了分析总结，并给出了各种典型例题及解答，以帮助学生加深对知识要点的理解。为了使学生加深对具体操作系统的了解，本书还对当前流行的几种操作系统——Unix System V、Windows NT、IBM MVS 和 Linux 的相关知识做了一些简要的介绍。

全书共分 13 章，分别是操作系统概述、进程描述与控制、并发控制、死锁处理、内存管理、处理机调度、I/O 设备管理、文件管理、分布计算、分布式进程管理、操作系统安全性和操作系统设计原则。章节编排与《操作系统原理》教材相似，各章分为重点与难点、例题解答和自测练习三部分，内容涵盖了整个《操作系统原理》教材的知识面和主要内容。最后还给出了 5 套模拟试题及解答供练习之用。

本书由何炎祥主编，朱骁峰参与编写。在本书编写过程中得到了华中科技大学出版社的大力支持，书中还引用了一些专家学者的成果，在此谨表衷心的感谢。

由于时间仓促，而且编者水平所限，书中的错误与疏漏之处在所难免，恳请读者批评指正。

编　者

2002 年 7 月于武昌珞珈山

图书在版编目(CIP)数据

操作系统原理学习指导与题解/何炎祥 主编
武汉：华中科技大学出版社，2003 年 1 月
ISBN 7-5609-
Ⅰ. 操Λ
Ⅱ. ①何Λ ②朱…
Ⅲ. 操作系统-原理-学习指导-题解
Ⅳ.

操作系统原理学习指导与题解 **何炎祥 主编**

责任编辑：叶见欣 **封面设计：**
责任校对：陈元玉 **责任监印：**

出版发行者：华中科技大学出版社 **武昌 喻家山 邮编：**430074 **电话：**(027)87542624

经销者：新华书店湖北发行所

录排者：华中科技大学惠友文印中心
印刷者：

开本：787×960/16 **印张：** **字数：**
版次：2003 **年** 1 **月第** 1 **版** **印次：**2003 **年** 1 **月第** 1 **次印刷** **印数：**
ISBN 7-5609-- **定价： 元**

目　录

第 1 章　操作系统概述 (1)

1.1　重点与难点 (1)

1.1.1　概述 (1)

1.1.2　知识要点 (2)

1.1.3　系统实例 (6)

1.2　例题解答 (6)

1.3　自测练习 (12)

1.4　自测练习答案 (14)

第 2 章　进程描述与控制 (15)

2.1　重点与难点 (15)

2.1.1　概述 (15)

2.1.2　知识要点 (15)

2.1.3　系统实例 (18)

2.2　例题解答 (19)

2.3　自测练习 (24)

2.4　自测练习答案 (25)

第 3 章　并发控制 (27)

3.1　重点与难点 (27)

3.1.1　概述 (27)

3.1.2　知识要点 (28)

3.1.3　系统实例 (32)

3.2　例题解答 (34)

3.3　自测练习 (56)

3.4　自测练习答案 (59)

第 4 章　死锁处理 …… (63)
4.1　重点与难点 …… (63)
4.1.1　概述 …… (63)
4.1.2　知识要点 …… (63)
4.1.3　系统实例 …… (68)
4.2　例题解答 …… (68)
4.3　自测练习 …… (76)
4.4　自测练习答案 …… (79)
第 5 章　内存管理 …… (81)
5.1　重点与难点 …… (81)
5.1.1　概述 …… (81)
5.1.2　知识要点 …… (82)
5.1.3　系统实例 …… (88)
5.2　例题解答 …… (89)
5.3　自测练习 …… (99)
5.4　自测练习答案 …… (102)
第 6 章　处理机调度 …… (105)
6.1　重点与难点 …… (105)
6.1.1　概述 …… (105)
6.1.2　知识要点 …… (105)
6.1.3　系统实例 …… (109)
6.2　例题解答 …… (110)
6.3　自测练习 …… (117)
6.4　自测练习答案 …… (119)
第 7 章　I/O 设备管理 …… (121)
7.1　重点与难点 …… (121)
7.1.1　概述 …… (121)
7.1.2　知识要点 …… (122)
7.1.3　系统实例 …… (126)
7.2　例题解答 …… (126)
7.3　自测练习 …… (133)

7.4 自测练习答案 (134)
第 8 章 文件管理 (136)
8.1 重点与难点 (136)
8.1.1 概述 (136)
8.1.2 知识要点 (137)
8.1.3 系统实例 (141)
8.2 例题解答 (142)
8.3 自测练习 (145)
8.4 自测练习答案 (147)
第 9 章 分布计算 (149)
9.1 重点与难点 (149)
9.1.1 概述 (149)
9.1.2 知识要点 (150)
9.2 例题解答 (152)
9.3 自测练习 (155)
9.4 自测练习答案 (156)
第 10 章 分布式进程管理 (158)
10.1 重点与难点 (158)
10.1.1 概述 (158)
10.1.2 知识要点 (159)
10.2 例题解答 (161)
10.3 自测练习 (167)
10.4 自测练习答案 (168)
第 11 章 操作系统的安全性 (171)
11.1 重点与难点 (171)
11.1.1 概述 (171)
11.1.2 知识要点 (172)
11.1.3 系统实例 (176)
11.2 例题解答 (176)
11.3 自测练习 (179)
11.4 自测练习答案 (181)

第 12 章　操作系统设计原则 ..... (183)
12.1　重点与难点 ..... (183)
12.1.1　概述 ..... (183)
12.1.2　知识要点 ..... (183)
12.2　例题解答 ..... (184)
12.3　自测练习 ..... (186)
12.4　自测练习答案 ..... (188)
第 13 章　综合测试 ..... (190)
模拟试题Ⅰ ..... (190)
模拟试题Ⅰ参考答案 ..... (192)
模拟试题Ⅱ ..... (194)
模拟试题Ⅱ参考答案 ..... (197)
模拟试题Ⅲ ..... (200)
模拟试题Ⅲ参考答案 ..... (203)
模拟试题Ⅳ ..... (208)
模拟试题Ⅳ参考答案 ..... (210)
模拟试题Ⅴ ..... (213)
模拟试题Ⅴ参考答案 ..... (216)
附录 ..... (219)
思考题 ..... (219)
思考题参考答案 ..... (227)
参考文献 ..... (235)

第 1 章

操作系统概述

1.1 重点与难点

1.1.1 概述

操作系统（OS）是由一系列程序模块组成的，其最基本的功能是资源管理。它管理处理机、内存、I/O 设备和数据等。操作系统的功能通常是由软件来实现的。近年来也采用了由固件来实现的新技术，即把某些经常使用的程序模块的微代码指令固化在高速存储器中，从而提高了处理能力。

操作系统发展至今已有 30 多年。设计操作系统主要有两个目的：第一，为程序的开发和执行提供一个方便的环境；第二，为保证计算机系统顺利执行，操作系统将对各个计算活动进行调度。

操作系统的形成和发展是与计算机硬件发展密切相关的。从最初的手工操作开始，相继产生了批处理系统、执行系统、多道程序系统、多道批处理系统、分时系统和实时系统等。反过来，操作系统的发展又促进了硬件的发展。分时系统之前的系统主要解决资源的合理利用问题，而分时系统的着眼点在于为用户提供良好的工作环境，这大大推动了计算机的应用和普及。

操作系统提供了大量的服务，在最低层是系统调用，多为运行的程序直接调用；在较高层，命令解释程序通过命令的形式为用户提供请求服务的机制，这些命令来自卡片（批处理）或直接来自终端交互式系统或分时系统。系统程序提供了能满足用户请求的另一种机制。

请求的层次不同，其类型也不同，在系统调用层必须提供基本功能，如进程控制和文件管理、设备管理等。较高层的请求由命令解释程序或系统程序满足，它们被翻译成一系列系统调用。所以最终都是由系统调用实现的。系统服务可分

成若干类型：程序控制、状态请求、I/O 请求等，程序错误可看做是隐式的服务请求。

多处理机系统是近几年发展起来的，特别是局部网络和分布式系统，对扩大计算机的应用，丰富计算机理论有重大推动作用。

现在是大型机、小型机和微型机并举的局面，尤其是微型机、超级微型机的发展速度非常快。在 16 位、32 位微型机上，Windows 操作系统占据统治地位。而中、小型计算机的操作系统和部分高档微机的操作系统则以 Unix 操作系统为主。

1.1.2 知识要点

1. 操作系统的有关概念

(1) 操作系统

操作系统是计算机用户和计算机硬件之间的接口程序模块，是计算机系统的核心控制软件。其职能是控制和管理系统内各种资源，有效地组织多道程序的运行，为用户提供良好的工作环境，达到使用方便、资源分配合理、安全可靠等目的。简而言之，操作系统的作用就是提供人机交互界面，组织和管理系统资源。

(2) 操作系统的作用

操作系统提供了程序执行的环境。这种环境是通过它所提供的各种服务设施来体现的。这些服务主要有：

① 程序执行。装入并执行程序（含排错）。

② I/O 操作。执行所有的读、写及相关的操作。

③ 文件系统管理。允许用户创建、删除、打开、关闭、修改文件等。

④ 错误检测。检测并报告 CPU、硬件、指令、设备等的错误。

另外，还有一些服务是用来使系统本身有效运行的。主要有：

① 资源分配。管理多种不同类型的资源，为多个用户和作业分配资源。

② 记账。统计用户使用资源情况。

③ 保护。对信息及作业加以适当控制。

(3) 系统调用

操作系统通过系统调用来实现基本的服务。系统调用可粗略地分为三类：进程及作业控制；设备及文件管理；信息管理。

(4) 促进操作系统发展的因素

操作系统的发展受到多方面因素的影响，首先是硬件的发展，为操作系统的发展提供了物质基础；其次是新的服务要求，人们在使用任何软件时总会感

到这样或那样的不足，就会提出更高的要求，这成为操作系统发展的动力之一；最后，软件总是存在各种错误，对错误的修正也从客观上促使了操作系统的发展。

2. 操作系统的演变

操作系统并不是一开始就存在的，它有一个产生和演变的历史过程。在 20 世纪 40 年代末到 50 年代中期，操作系统还未出现，程序员直接与硬件接触。这个时期的系统有两个问题：一是，上机要事先预约机器时间，人们必须估计自己的程序大概要多少时间；二是，启动时间过长，操作相当繁琐且容易出错。这两个问题会对珍贵的机器时间资源造成巨大的浪费。

(1) 简单批处理系统

简单批处理系统就是人们为了解决上述矛盾而开发出来的。其中心思想是，通过应用一种被称为监视器的软件，使用户不必再直接接触机器，而是先通过卡片机和纸带机向计算机控制器提交作业，由监视器将作业组织在一起构成一批作业，然后将整批作业放入由监视器管理的输入设备上，每当一个程序执行完毕返回监视器时，监视器已自动装入下一个程序。此时的操作系统只是一个简单的计算机程序，它需要一些硬件的支持，如存储器保护、计时器、特权指令和中断等，其缺点是处理器常常闲置，因为 I/O 设备的执行速度比处理器慢得多。

监视器软件的主要部分有：控制卡解释程序、设备驱动程序和装配程序。

(2) 多道程序批处理系统

操作系统进一步向前发展，这时出现了多道程序批处理系统。其设计思想是存储空间中有一个操作系统和多个用户程序，当一个作业等待 I/O 时，处理器可转向另一个作业，这样就不必等待 I/O，它是现代操作系统的主旋律。同简单批处理系统一样，多道程序批处理系统也需要中断和 DMA 等硬件支持。

(3) 分时系统

使用多道程序批处理技术大大提高了系统的效率，但是，这个系统的人机交互性并不好，这是分时系统得以发展的动因。分时系统的基本思想是让多个用户同时通过终端使用系统，而操作系统则在系统内部处理用户程序。

分时系统和多道程序批处理系统都使用了多道程序设计技术。其关键的不同之处如表 1.1 所示。

表 1.1　多道程序批处理系统与分时系统

	多道程序批处理系统	分时系统
策略目标	使处理机使用率最高	使响应时间最小
指令源	作业控制语言提供	在终端上键入命令

(4) 实时系统

实时系统是在响应时间方面有严格制约的专用系统。实时系统与分时系统的区别在于：在分时系统中，快速响应是需要的，但不是必须的；在实时系统中，处理事务必须在适合于此系统的特定时间限额内完成。

3. 操作系统的主要成就

操作系统是现有软件系统中最复杂的软件之一。Denning 认为，到目前为止，在操作系统的研究开发方面主要取得了进程、内存管理、信息保护与安全、调度与资源管理和系统结构等五项成就。

(1) 进程

进程是一个执行的程序，能分配给处理器并在其上执行的实体。它是操作系统的基础。进程由三部分组成：一个可执行的程序；该程序所需的相关数据(变量、工作空间，缓冲区等)；该程序的执行上下文(Context)。进程可以看做是一个数据结构。进程既可以被执行，又可以等待执行。进程的整个状态都保存在其上下文中。可以扩展进程的上下文以允许加入新的信息。

(2) 存储器管理

存储器管理是应用户的使用要求和管理员的管理要求而产生的。操作系统对存储器的管理应遵循以下五个原则：

① 进程隔离。

② 自动分配和管理。

③ 支持组件编程。

④ 长时间存储。

⑤ 保护和存取控制。

操作系统用虚拟存储器(Virtual Memory)和文件系统来满足上述要求。虚拟存储器允许程序以逻辑方法来寻址，而不用考虑物理上可获得的内存大小。这在内存管理一章有更详细的介绍。

(3) 信息保护与安全

安全是指一个信息系统能保证信息的保密性、完整性和可用性的能力。在安全机制的实施方面通常主要考虑对计算机系统和存储在其中的信息的存取控制。与操作系统安全有关的工作主要有访问控制、信息流控制和确认三类。

(4) 调度和资源管理

调度和资源管理是操作系统的核心任务之一，资源分配和调度策略都必须考虑三个因素：公平性、不同敏感性和效率。调度模块是操作系统的关键组件。

操作系统自诞生以来，其大小和复杂性都在不断增加，进而促使人们开始重视操作系统的软件结构。认识到软件必须组件化，对于庞大的操作系统来说，还

要用到体系结构分层和信息抽象技术。现代操作系统的体系结构分层是根据其复杂性以及抽象的水平来分离功能的。可以将系统看成一个分层结构，每层完成操作系统要求的一个功能子集，每层都依赖紧挨着的较低一层的功能，并且为较高层提供服务。

4. 操作系统的主要研究课题

操作系统的主要研究课题及其关系如图1.1所示。

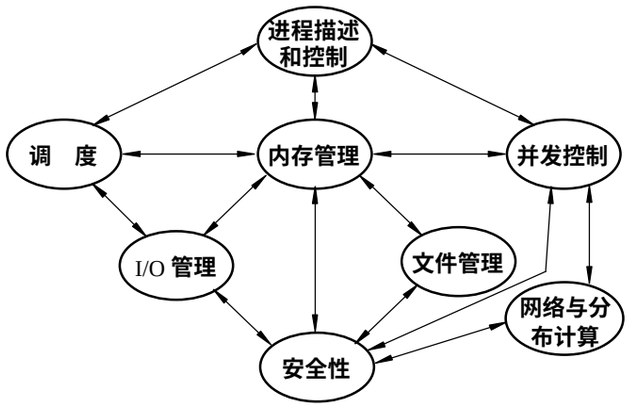


图 1.1

5. 操作系统的其他技术和概念

(1) SPOOLing 技术

SPOOLing 是“ Simultaneous Peripheral Operation On Line ”的缩写。SPOOLing 技术将磁盘作为大容量辅助存储器供存取(输入/输出）数据用。其主要用途是防止两个（或多个）用户将各自的信息交叉地输入到打印机上，发生交叉打印的情况。它也有助于减少 I/O 设备和 CPU 的空闲时间。

(2) 缓冲

开辟一片辅助的存储区用作输入/输出时的缓冲区，使 CPU 不再总是直接与 I/O 设备本身打交道。输入时，在 CPU 需要数据之前，就把数据读到缓冲区，需要时，CPU 可直接存取缓冲区中的数据。输出时，CPU 先把输出信息送入缓冲区，而在 I/O 设备打印输出缓冲区的内容时，CPU 又可做其他工作。

(3) 设备独立性

设备独立性是指系统允许一种（台）输入/输出设备被另一种输入/输出设备所替换且无需修改用户程序。

(4) 中断、中断向量

中断是由设备通过控制线（而不是数据线）向 CPU 发送的一种信号，以指明此时需要某种服务。中断向量就是一个包含每个中断处理程序开始地址的表格。I/O 请求与 I/O 中断的联系在于当一个 I/O 请求完成时引起一次 I/O 中断。

(5) DMA

DMA（Direct Memory Access）允许不经 CPU 干预就可在 I/O 设备与主存储器之间传递数据。这种传递是在 CPU 指令执行期间进行的（称为“ 交叉 ”执行）。

1.1.3 系统实例

系统实例的详细信息可以参考课本[1]中 1.4 节的介绍。

1.2 例 题 解 答

例 1-1 操作系统的主要目标是什么？

解 操作系统的主要目标是：

(1) 为计算机用户提供一个良好的环境，使其能以方便、有效的方式在计算机硬件上执行程序。

(2) 根据解决某给定问题的需要来分配计算机的各种资源。而且，这种分配应尽可能公平、有效。

(3) 作为控制程序，它有如下两种主要功能。

① 监控用户程序的执行以避免发生各种错误和对计算机系统的不合理使用；

② 对 I/O 设备的操作和控制的管理。

(4) 合理地组织计算机系统的工作流程，以改善系统的性能。

例 1-2 列出人们早期在计算机上运行一个程序所必须经过的步骤。

解 人们早期在计算机上运行一个程序必须经过以下四个步骤：

(1) 预约机器时间；

(2) 将程序手工装入内存；

(3) 指定开始地址，启动程序运行；

(4) 从控制台上监控程序的执行。

例 1-3 设计操作系统时是需要一些硬件支持的，试给出在设计一个操作系统时的三种（或三种以上）硬件支持。

---

[1] 见参考文献 1

解　设计一个操作系统需要以下硬件支持：

(1) **监控态／管态；**

(2) **特权指令；**

(3) **定时器；**

(4) **内存保护机制。**

例 1-4　**直到出现直接存取内存通道之后，多道程序概念才变为有用的。解释这是为什么？**

解　以前外围 I/O **设备与主存间的数据传递一直是直接由** CPU **控制的。直接存取内存通道的出现，为** I/O **设备与主存间提供了一个直接的接口，这样就可让** CPU **去执行更为重要的任务，从而为** CPU **执行多道程序提供了基础。**

例 1-5　**在一个多道程序和分时环境中，若干用户同时共享系统，就有可能导致所谓的“安全性”问题。**

(1) **试举出几个这样的问题。**

(2) **我们能保证共享系统和早期的计算机系统会有同样的安全度吗？**

解　(1) **窃取或复制他人的程序或数据、不适当地使用系统资源（**CPU**、内存、硬盘空间、外设）等将有可能导致所谓的“安全性”问题。**

(2) **我们不能保证共享系统和早期的计算机系统具有同样的安全度，因为人们常常难以避免会违反自己所设计的保护方案，而且保护方案越复杂就越难保证能正确地实现它。**

例 1-6　**某些早期系统将操作系统存放在用户和操作系统都不能修改的内存区域，以实现保护的目的。问：这种方案可能会出现什么问题？**

解　操作系统所需的数据（口令、存取控制、统计信息等）可能不得不存储在未受保护的存储区域或通过未受保护的存储区域传递，这样，它们就可能被非授权用户所访问。

例 1-7　**指明下面各类操作系统的本质区别：**

(1) **批处理操作系统；**

(2) **交互式操作系统；**

(3) **分时操作系统；**

(4) **实时操作系统。**

解　(1) **批处理操作系统：将具有类似要求的作业分组，由操作员或作业定序程序将它们在计算机上成组地运行。它通过采用缓冲、**SPOOLing **和多道程序设计等技术，尽可能使** CPU **和** I/O **设备在任何时刻都处于忙状态来提高系统的性能。批处理操作系统适用于运行几乎不需要交互性的大型作业。**

(2) **交互式操作系统：提交的作业由许多较短的事务组成，其中，下一事务的结果可能是不可预测的。因为用户提交了作业后等待马上见到结果（或反应），**

所以响应时间应该较快（响应时间以秒计）。

(3) 分时操作系统。它使用 CPU 调度和多道程序设计技术，提供交互式使用系统的功能和环境。CPU 能从一个用户迅速地转移到另一个用户。用户从终端上输入信息，系统直接将结果（或反应）马上输出到屏幕上。系统对用户的响应时间很短，就好像系统仅由个人独占一样。

(4) 实时操作系统。它常常用于专用系统中。这类系统从传感设备上读取信息并在规定的时间内作出响应。这类系统要求稳定、可靠，对用户的响应时间比较严格。

例 1-8　设计适用于实时环境的操作系统的主要困难是什么？

解　设计适用于实时环境的操作系统的主要困难是，在实时环境规定的时间限额内对用户作出相应的反应。如果系统不能在规定的时间限额内完成指定的任务，就可能导致整个实时系统的崩溃。因此，在设计这类操作系统时，设计者必须保证所采用的调度策略及相关技术不会使响应时间超过实时环境所规定的时间限额。

例 1-9　早期操作系统的缺陷之一是，用户失去了与其作业交互的能力。现代操作系统是如何解决这一问题的？

解　分时操作系统与早期的批处理操作系统不同，它通过划分时间片轮流处理的方法，允许多个用户同时通过终端使用系统，还能使其中的每个用户都能与自己的作业交互。而早期的操作系统是批处理系统，从用户提交作业到作业执行完毕这段时间内，用户无法与自己的作业交互。现代操作系统无一例外地实现了分时操作系统的功能，所以说现代操作系统都使用户有与作业交互的能力。

例 1-10　为什么说 SPOOLing 技术对批处理（多道程序）系统是必须的？它对分时系统也是必须的吗？

解　SPOOLing 技术将磁盘作为大容量辅助存储器供存取输入（输出）数据用。在批处理系统中，它可以防止用户将各自的信息交叉地输入到打印机上，发生交叉打印的情况。它也有助于减少 I/O 设备和 CPU 的空闲时间。因此，SPOOLing 对批处理多道程序设计是必须的。

分时系统一般不需要 SPOOLing，是因为每个事务通常较短，而且每个用户都有自己的终端设备，输出信息一般直接输出到用户自己的打印设备上。但是，在分时系统的共享低速设备（如打印机）管理中，仍要用到 SPOOLing 技术。

例 1-11　列出监控程序认为是非法的一些操作。

解　监控程序认为以下操作是非法的：

(1) 程序设计错误，如非法指令，地址错等。

(2) 某个用户作业阅读了供下一用户作业阅读的控制卡片，致使作业之间发生干扰。

(3) 屏蔽所有中断，使得没有一个中断能发生；接通了若干中断，致使作业与 I/O 间发生干扰。

(4) 停机。

(5) 将用户态改为监控态（即目态改为管态）或反之，导致用户可以控制系统。

(6) 使用内存之外的用户区。

(7) 修改监控程序中的中断向量（可能导致系统崩溃）。

(8) 访问监控程序的存储区，等等。

例 1-12　在多任务操作系统中为什么不用缓冲技术解决多用户打印问题？

解　因为 SPOOLing 技术在以下方面优于缓冲技术：缓冲技术使一个作业的输入／输出与其自身的计算重叠；SPOOLing 技术则使某个作业的输入／输出与其他作业的计算重叠。此外，采用 SPOOLing 技术，系统可从作业缓冲池中选择一种较好的作业搭配（即选择偏重 I/O 的作业和偏重 CPU 的作业的一种合适的搭配），从而使得 I/O 操作和 CPU 计算差不多总是在进行，提高了系统的效率，减少了系统的空闲时间。

例 1-13　列出操作系统控制进程／作业的一些手段。

解　操作系统控制进程／作业的手段如下：

(1) 设置错误级别；

(2) 装入／连接／执行另一程序；

(3) 创建新作业／进程；

(4) 取／存进程属性；

(5) 终止作业／进程；

(6) 等待特定事件或时间；

(7) 内存转储；

(8) 指令追踪；

(9) 创建时间分布图。

例 1-14　早期在计算机系统上执行一个作业的大致过程是哪些？

解　早期在计算机系统上执行一个作业的大致过程如下：

(1) 安装“装入程序”的磁带；

(2) 安装“编译程序”的磁带；

(3) 装入源程序；

(4) 执行编译程序并将输出信息存储到磁带上；

(5) 返绕每个磁带；

(6) 若编译程序的输出为汇编语言程序，则装入“汇编程序”；

(7) 返绕磁带；

(8) 执行汇编程序并将目标程序输出到磁带上；

(9) 返绕磁带；

(10) 从磁带上装入目标程序；

(11) 执行目标程序并在打印机上输出结果信息。

例 1-15　用户与操作系统之间的接口有哪几种？它们各用于什么情况下？

解　用户与操作系统之间的接口主要有广义指令、键盘操作命令和脱机操作命令。广义指令可直接写在用户程序中，它是为了方便地使用系统资源而提供的一种宏指令；键盘操作命令用于联机运行环境，用户从键盘上打入命令，直接控制作业的运行；脱机操作命令则用于脱机运行情况，用户用这些命令编写作业说明书，以告诉系统对作业的控制意图和处理方式。

例 1-16　偏重 I/O 型的程序用于等待 I/O 的时间较长，而使用处理器的时间较短的场合；而偏重处理器的程序则相反。假设有一种短程调度算法满足最近使用处理器较少的进程。试解释为什么这种算法对偏重 I/O 的程序有利，但也不会总是拒绝给予偏重处理器型程序使用处理器的时间。

解　因为此调度算法偏重最近使用处理器最少的程序，而偏重 I/O 型的程序用于等待 I/O 的时间较长，而使用处理器的时间较短的场合，所以偏重 I/O 型的程序在从 I/O 返回后，总能优先得到处理器。另一方面，偏重 I/O 型的程序经常进行 I/O 操作，使处理器空闲，所以偏重处理器的程序也不会一直得不到处理器。

例 1-17　假设有一个支持多道程序设计的计算机系统，其中每个作业都有完全相同的属性。对于一个作业，在一个时间段 T 中，一半的时间用于 I/O，另一半时间用于处理器操作。每个作业总共运行 N 段周期。有几个定义如下。

平均周期(Turnaround time)=平均完成一个作业实际用的时间；

吞吐量(Throughput)=在每一时间段 T 中完成的平均作业数；

处理器使用率(Processor utilization)=处理器处于激活态(非等待)时间的百分比。

计算当有 1 个、2 个或 4 个作业并行执行时的周期、吞吐量和处理器使用率，假设时间段 T 按以下任一种方式分布：

(1)　I/O 在前半段，处理器运行于后半段；

(2) 将 T 分为 4 段，I/O 在第 1、4 段，处理器运行于第 2、3 段。

解　按照题目中所述的要求，分别在 CPU 和 I/O 时间轴上画出时间段，模拟系统调度即可统计得出所要数据，结果如表 1.2 所示。

由表结果可知：当同时运行 2 个作业时系统吞吐量和 CPU 利用率显著增加，表明系统被充分利用。但是当作业增加到 4 个时吞吐量和 CPU 使用率变化不大，但平均周期却增加一倍，表明系统负荷过重，作业处理时间明显增长。其原因是作业的 CPU 处理时间及 I/O 时间各占一半，两个作业正好互补，可以充分利用系统资源。方式 1 和方式 2 的结果相同，可见，这种互补利用系统资源产生的效益仅与多个作业在 CPU 和 I/O 的占用时间上的互补程度有关。

表 1.2　统计结果

任务数	方式（1）			方式（2）		
	平均周期/T	吞吐量	CPU 使用率	平均周期/T	吞吐量	CPU 使用率
1	N	1/N	0.5	N	1/N	0.5
2	$N\frac{1}{4}$	4/(2N+1)	2N/(2N+1)	$N\frac{1}{4}$	4/(2N+1)	2N/(2N+1)
4	(8N–1)/4	8/(4N+1)	4N/(4N+1)	(8N–1)/4	8/(4N+1)	4N/(4N+1)

例 1-18　某计算机用 Cache、内存和磁盘来实现虚拟内存。如果某数据在 Cache 中，访问它需要 t_A(ns)；如果在内存但不在 Cache 中，则需要 t_B(ns)的时间将其装入 Cache 然后开始访问 如果不在内存中，则需要 t_C(ns)将其读入内存，然后用 t_B(ns)读入 Cache。如果 Cache 命中率为(n–1)/n，内存命中率为(m–1)/m，则平均访问时间是多少？

解　根据题目中所给数据，平均访问时间 T_{av} 为：

$$T_{av}=\left(\frac{n-1}{n}\right)\cdot t_A+\frac{1}{n}\left(\frac{m-1}{m}\right)\cdot\left(t_A+t_B\right)+\frac{1}{n}\cdot\frac{1}{m}\cdot\left(t_A+t_B+t_C\right)$$

$$=t_A+\frac{1}{n}\cdot t_B+\frac{1}{m\cdot n}t_C$$

例 1-19　多道程序操作系统的关键组件如图 1.2 所示。现定义等待时间为一个作业停留在短程队列中的时间，请给出周期、处理器忙时间(Processor Busy Time)以及等待时间之间的相关表达式。

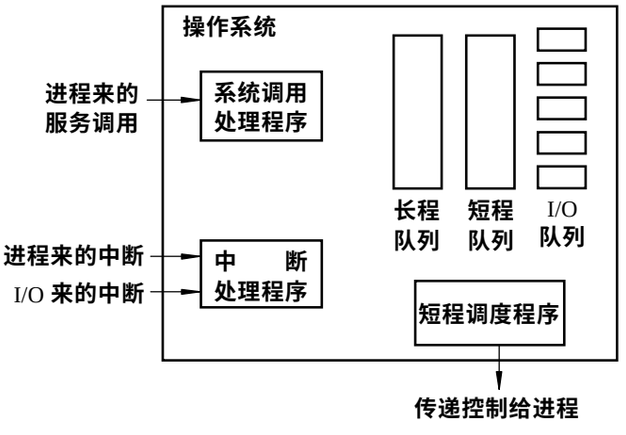


图 1. 2

解　在此结构中，一个程序处在短程队列中，如果它在等待 CPU，则表示

CPU 正忙，否则它就会立即被调度运行，所以，

周期＝∑处理器忙时间＋∑等待时间

1.3 自 测 练 习

一、填空题

1. 人们早期在计算机上运行一个程序，必须经过_____、_____、_____和_____四个步骤。
2. 设备状态表包含_____、_____和_____。
3. 由用户使用的排错辅助手段有_____、_____和_____。
4. 传递由系统调用所需要的参数有_____和_____ 两种。
5. 用不同的 I/O 设备来运行一个程序的能力称为_____。
6. 分时系统的三个特点是_____、_____和_____。
7. 系统服务主要分为_____和_____两种。
8. 文件操作系统程序用以_____、_____、_____、_____、_____、_____和_____等对文件及目录的操作。
9. Denning 认为，到目前为止在操作系统的研究开发方面取得了_____、_____、_____、_____和_____五项主要成就。
10. Unix 操作系统的系统部分通常称为_____。
11. Windows NT 操作系统由_____、_____、_____和_____四层组成。

二、判断题（正确的在括号中记√，错误的记×）

1. 可以将操作系统看做是一个资源分配器（Resource Al1ocator），用来控制 I/O 设备和用户的程序。 ( )
2. 操作系统是一种时间驱动程序。 ( )
3. 设备独立性就是指系统具有使用不同设备的能力。 ( )
4. 多机系统就是由两个或多个计算机相连而成的系统。 ( )
5. 主／辅计算机系统是指主机控制辅机的各种动作的系统。 ( )
6. RJE 代表 Remote Job Entry（远程作业录入）。 ( )
7. 时间分布图是展示每条指令或每组指令执行时间情况的一种图表。 ( )
8. Windows NT 是一个多用户、多任务操作系统。 ( )

三、选择题

1. 下面哪些指令是特权指令？ ( )

a) 置定时器之值；
b) 读时钟；
c) 清内存；
d) 关闭中断；
e) 从用户态（目态）转到监控态（管态）。

2. 保护常驻监控程序是正确管理计算机系统的关键。双态状况、内存和定时器都需要提供相应的保护。但是，为了具有最大的灵活性，希望对用户施加最小的限制。下面列出了通常受保护的一组操作，请指明其中必须予以保护的最小指令集。（　　）

a) 改为用户态；
b) 改为监控态；
c) 读监控程序区；
d) 写监控程序区；
e) 从监控程序区取指令；
f) 开定时器中断；
g) 关定时器中断。

3. 下面哪些属于系统程序的范畴？（　　）

a) 文件操作；
b) 取状态信息；
c) 修改文件；
d) 程序语言支持环境；
e) 程序连接／装配／执行；
f) 应用程序工具。

4. 下列指令哪些属于特权指令？（　　）

a) I/O 指令；
b) 修改界限寄存器指令；
c) 修改时钟指令；
d) HALT 指令；
e) 变更“目态／管态”指令。

四、问答题

1. 操作系统的主要目标是什么？
2. 什么叫 SPOOLing？SPOOLing 技术优于缓冲技术体现在哪些方面？
3. 指明下面各类操作系统的本质区别：

(1) 批处理操作系统；

(2) 交互式操作系统；
(3) 分时操作系统；
(4) 实时操作系统。

1.4 自测练习答案

一、填空题

1. 人们早期在计算机上运行一个程序，必须经过预约机器时间、将程序手工装入内存、指定开始地址启动程序运行和从控制台上监控程序的执行四个步骤。
2. 设备状态表包含设备类型、设备地址和设备当前状态。
3. 由用户使用的排错辅助手段有内存转储、指令追踪和创建时间分布图。
4. 传递由系统调用所需要的参数有寄存器传递参数和寄存器传递参数区始址 两种。
5. 用不同的 I/O 设备来运行一个程序的能力称为设备独立性。
6. 分时系统的三个特点是多路性、交互性和独占性。
7. 系统服务主要分为系统调用和系统程序两种。
8. 文件操作系统程序用以创建、删除、复制、换名、打印、转储和列表等对文件及目录的操作。
9. Denning 认为到目前为止在操作系统的研究开发方面取得了进程、内存管理、信息保护与安全、调度与资源管理和系统结构五项主要成就。
10. Unix 操作系统的系统部分通常称为内核。
11. Windows NT 操作系统由硬件抽象层、内核、子系统和系统服务四层组成。

二、判断题

1. √　　2. √　　3. ×　　4. √
5. √　　6. ×　　7. √　　8. ×

三、选择题

1. a)、c)、d)、e)　　2. b)、c)、d)、g)
3. a)、b)、c)、d)、e)、f)　　4. a)、b)、c)、d)

四、问答题

见重点与难点和例题解答。

第 2 章

进程描述与控制

2.1 重点与难点

2.1.1 概述

现代操作系统中最基本的构件就是进程。操作系统的重要功能就是创建、管理和终止进程。当进程处于活动状态时，操作系统必须保证每一进程都分到处理器执行时间，还要协调它们的活动，管理冲突请求，并分配系统资源给这些进程。要履行其进程管理职能，操作系统必须维持对每一进程的描述。每个进程都是由一个进程映像来表示的，它包括进程执行的地址空间和一个进程控制块。后者包含了操作系统管理该进程所需的全部信息，包括其目前的状态，分配给它的资源、优先级以及其他有关数据。

进程在生命周期中，会在很多状态之间移动。这些状态中最重要的是就绪、运行和阻塞。运行进程可能因中断或执行操作系统的访管而中止，所谓中断是指发生在进程之外并可被处理器识别的事件。在中断和访管这两种情况下，处理器都将执行一个切换操作，将控制转交给操作系统例程。在完成所需工作后，操作系统可能恢复被中止的进程或切换到另一进程。

本章是操作系统中较重要的一章，本章的学习重点是基础概念，这一部分知识为以后的学习打下了基础。

2.1.2 知识要点

1. 进程状态

(1) **进程**

进程是程序在并发环境中的执行过程。其基本特征是动态性、并发性、独立

性、异步性和结构性。进程是一个主动的实体，而程序是被动的实体。进程的执行必须按一种顺序的方式进行，即在任何时刻至多只有一条指令被执行。

(2) 进程状态及转换模型

进程的动态性质是由其状态变化决定的。通常操作系统中进程都具有三种状态：运行态、就绪态和阻塞态。进程的生命周期中通常还有创建和消失两种状态。一个典型的进程状态模型如图 2.1 所示。

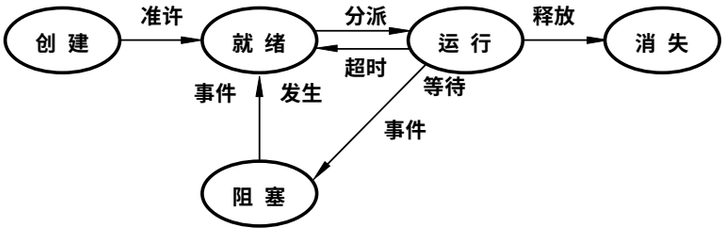


图 2.1

由于 I/O 操作比 CPU 计算慢得多，故常会出现内存中所有进程都等待 I/O 的现象。即使运行多个程序，处理器在大多数时间仍处于空闲状态。为此可采用交换（Swapping）方法，将内存中的一部分进程转移到磁盘中。在进程行为模式中需增加一个新的挂起（Suspend）状态，当内存所有进程阻塞时，操作系统可将一进程置为挂起状态并交换到磁盘，再调入另一进程执行。挂起状态与原有的阻塞和就绪状态结合为阻塞挂起状态（进程在辅存中且等待某事件的发生）和就绪挂起状态（进程在辅存中但只要被调入主存就可执行）。

详细的状态转换信息可以参考课本[1]中的表 2.5。

2. 进程描述

(1) 进程映像

操作系统通过称为进程映像（Process Image）的进程属性集合来控制进程。进程映像中通常包括以下基本内容：

① 用户数据。

② 用户程序。

③ 系统栈。

④ 进程控制块。

(2) 进程控制块

进程控制块（PCB ，Process Control Block）是进程映像中最关键的部分，它含有进程的描述信息和控制信息，是进程动态特性的集中反映，是系统对进程施行识别和控制的依据。进程控制块的主要内容有：

① 进程标识。

② 处理器状态信息。

③ 进程控制信息。

有关进程及进程控制块的详细信息可以参考课本[1]中的表 2.7 和表 2.8。

3. 进程控制

操作系统内核负责控制和管理进程的产生、执行和消亡的整个过程，这主要通过对它们的控制操作来实现。操作系统的进程控制操作主要有：创建进程、撤消进程、挂起进程、恢复进程、改变进程优先级、封锁进程、唤醒进程、调度进程等。

一个进程可通过“创建”原语创建一新进程。创建新进程的工作主要包括：从 PCB 集合中获取一个空闲 PCB；对 PCB 进行初始化；为新进程的数据集分配内存空间并初始化；为新进程的程序文本分配内存并装入程序；将新进程的 PCB 插入到就绪队列中。

当前运行进程的状态要改变时，操作系统将进行以下步骤来完成进程切换：

① 保存处理器内容。

② 对当前运行进程的 PCB 进行更新。包括改变进程状态和其他相关信息。

③ 将这个进程的 PCB 插入到适当的队列(就绪、因事件阻塞、就绪挂起等)中。

④ 挑选其他进程执行。

⑤ 对挑选的进程 PCB 进行更新，包括将其状态改为运行。

⑥ 对存储器管理数据结构进行更新。

⑦ 恢复上次被选择进程移出时的处理器状态。

进程正常结束后，一般将主动终止而进入停止状态，同时向父进程发“完成”消息，等待父进程将其撤消。撤消进程包括回收它的 PCB，释放它所占用的全部资源。

4. 线程和 SMP

进程有两个基本属性：首先进程是一个拥有资源的独立单元，其次进程是一个被处理机独立调度和分配的单元。为了既能提高程序的并发程度，又能减少操作系统的开销，操作系统中引入了线程(Thread)。并且，引入线程的另一个好处是能更好的支持对称多处理（SMP）。

(1) 线程的概念

线程有许多不同的定义，综合起来可以把线程定义为：线程是进程内的一个相对独立的、可独立调度和指派的执行单元。从它的定义可以知道线程有如下性质：线程是进程内的一个相对独立的可执行单元，是基本调度单元，其中包含调

度所需的信息。 一个进程中至少应有一个线程，线程不拥有资源，而是共享和使用包含它的进程所拥有的所有资源，所以进程内的多个线程之间的通信需要同步机制。线程可以创建其他线程，有自己的生命期，也有状态变化。因此，线程在调度、拥有资源、并发性、系统开销上与进程有很大的区别。

(2) 线程与进程

线程与进程的关系，在不同操作系统中的实现也各不相同，有的是一对一的关系，有的是一对多的关系，还有的是多对多的关系。一些具体操作系统的实现可以参考课本[1]中的表 2.11。

线程在调度上与进程的调度相似，不同的系统有不同的设计方式，但就绪、运行、阻塞这三个状态是所有支持线程的系统所必须具备的。

(3) 线程的分类

线程在系统的实现上分为两类：用户级线程和内核级线程，也有的系统把两种线程结合起来实现，以弥补二者的缺陷。

用户级线程（ULT）由用户应用程序建立，由用户应用程序负责对这些线程进行调度和管理，操作系统内核并不知道有用户级线程的存在，MS-DOS 和 Unix 属于此类。ULT 的优点是：线程转换开销要小得多，并且线程的调度算法与操作系统的调度算法无关。其缺点是：一个线程的阻塞会导致进程中的所有线程都被阻塞，并且 ULT 没办法利用多处理器的优势。

内核级线程（KLT）是线程的创建、调度和管理这三个功能全部由操作系统内核负责的线程。其优点是：内核可以使一个进程中的多个线程在多个处理器上并行运行，从而提高了程序的执行速度和效率；并且，进程中的一个线程被阻塞，不会影响到进程中的其他线程；再者内核过程本身也都可以以线程方式实现。其缺点是：同一进程中的线程切换要有两次模式转换。

(4) 对称多处理结构

对称多处理是共享存储器多处理系统(Shared Memory Multiprocessor)的一种，并且每个处理器的地位相同，这种体系结构给操作系统的设计带来了新的要求。其关键是：同时并发执行多进程和多线程，防止死锁式非法操作；由于每个处理器都可独立调度，因此，必须避免冲突；提供互斥和同步机制，保证活动的进程或线程存取共享的地址空间和 I/O 资源；保持存储器硬件和存储管理结构的数据一致性和互斥访问；有很好的可靠性和容错能力。

2.1.3 系统实例

有关 Unix System V、Windows NT 和 IBM MVS 这三个操作系统在进程描述

与控制方面的信息可以参考课本[1]中 2.5 节的内容。在此主要介绍一下 Linux 操作系统有关进程的一些知识。

Linux 系统通过数据结构 task_struct 来描述进程，由一个 task_struct 类型的指针数 Task 来描述系统内的全部进程。task_struct 的主要内容如下。

① 状态：主要有运行、等待、停止和死亡等几种状态。

② 调度信息：供进程调度程序使用，以决定运行的进程。

③ 进程间通信机制：如信号、管道、共享内存、信号灯和消息队列。

④ 链接：指向其他相关进程的 task_struct 的指针。

系统中所有进程由一个双向链表链接，以供系统遍历查询。

2.2　例 题 解 答

例 2-1　进程和程序有哪些主要区别？

解　进程和程序之间的主要区别是：

(1) 程序是静态概念，本身可以作为一种软件资源保存；而进程是程序的一次执行过程，是动态概念，它有一定的生命期，是动态地产生和消亡的。

(2) 进程是一个能独立运行的单位，能与其他进程并发执行，进程是作为资源申请和调度单位存在的；而通常的程序段不能作为一个独立运行的单位。

(3) 程序和进程无一一对应关系。一方面一个程序可由多个进程共用；另一方面，一个进程在活动中又可顺序地执行若干个程序。

例 2-2　表 2.1 所示的是 VAX/VMS 操作系统的进程状态。问：

(1) 存在这么多明显的等待状态，你能给出一个合理的解释吗？

(2) 为什么以下状态没有常驻和换出版本：页面错等待、冲突页等待、公共事件等待、空闲页等待和资源等待？

(3) 画出状态转换图并标明引起每一转换的动作或发生的事件。

解　(1) 由于采用了虚拟内存机制，一部分内存页面被交换到外存上去，所以可计算、睡眠等待、局部事件等待、挂起等待这四个状态又一分为二，变成了八个状态，才会有这么多等待状态。

(2) 题中所述的五个状态，其等待条件通常会在很短的时间内被满足，如果在这些状态的进程被交换出的话，则很快又要被调入内存，给系统增加了很多不必要的 I/O 操作，会严重降低系统性能。

(3) 状态转换图如图 2.2 所示。

例 2-3　Pinkert 和 Wear 定义了下列进程状态：执行(运行)、活动(就绪)、阻塞和挂起。一进程如在等待许可使用某资源时，该进程就处于阻塞(状态)，若它

表 2.1　VAX/VMS 进程状态

进程状态	进程所处的情况
当前正在执行	正在执行进程
可计算(驻留)	就绪并且在内存中驻留
可计算(交换出)	就绪，但是被交换出了内存
页面错等待	进程访问了一个并不在内存中的页，并且必须等待该页被读入
冲突页等待	进程访问了一个共享页，该页是引起另一进程的页面错等待的原因，或者是一个当前正被读入或写出的私有页
公共事件等待	等待共享事件标志(事件标志是单个比特位的进程间信号量机制)
空闲页等待	等待内存中的一个空闲页以加入到分配给这个进程的内存页集合中
睡眠等待(驻留)	进程将自己放入等待状态
睡眠等待(交换出)	睡眠进程被交换出内存
局部事件等待(驻留)	进程在内存中并且等待一个局部事件标志(通常是I/O结束)
局部事件等待(交换出)	进程处于局部事件等待状态并被交换出内存
挂起等待(驻留)	进程被其他进程置于等待状态
挂起等待(交换出)	挂起的进程被交换出内存
资源等待	进程在等待各种系统资源

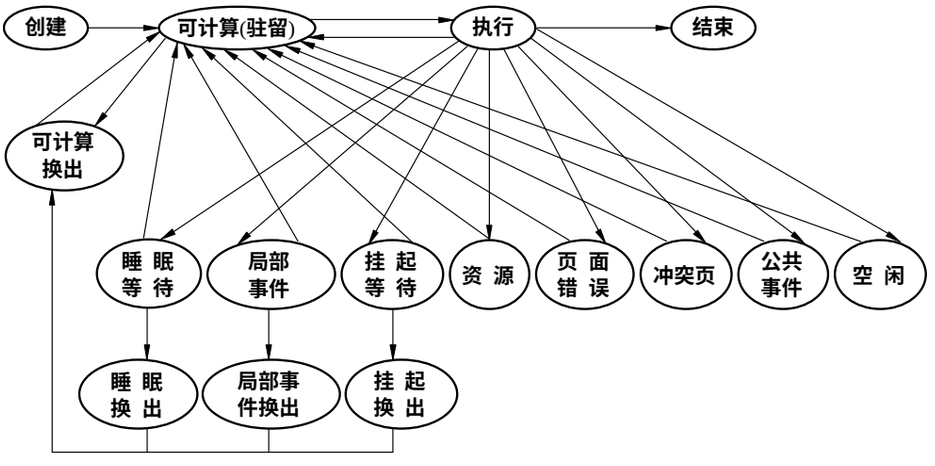


图 2.2

在等待对其已获得资源上的某操作的完成，则该进程处于挂起(状态)。在很多操作系统中，这两种状态合在一起作为封锁状态。比较这两种形式定义的优缺点。

解　下面对两种形式的定义作个比较。

有挂起状态的系统：可以提高 CPU 的利用率，进而提高系统的性能，但这样会增加 I/O 操作，使系统的调度更加复杂。

无挂起状态的系统：系统设计简单，易于实现，所以中小型系统、进程很少进行 I/O 操作的系统多采用这种设计方法。

例 2-4　进程基本状态变迁图如图 2.3 所示。

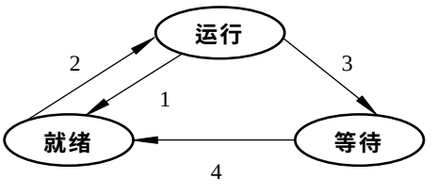


图 2.3

(1) 在什么情况下，将发生下述因果变迁？

a) 2　1；
b) 3　2；
c) 4　1；
d) 1　3。

(2) 在什么情况下，下述变迁不会立即引起其他变迁？

a) 1；
b) 2；
c) 3；
d) 4。

解　(1) 各种因果变迁发生的情况如下：

a) 当一个进程因到时间被阻塞时，另一个处于就绪状态的进程就被调度。
b) 不会出现这种情况。
c) 若无作业处于就绪状态和运行状态（系统空间状态），则一个作业一旦成为就绪状态就立即被调度。
d) 当运行进程等待外部事件和 I/O 事件。

(2) 不引起其他变迁的情况如下：

a) 不引起任何变迁。
b) 这种情况总引起 1 型变迁。
c) 若无作业就绪，则 3 型变迁不引起其他任何变迁。
d) 不引起变迁。

例 2-5　在一个多道程序系统中，一个进程从创建到消失通常要经历哪些阶段和状态？

解　在一个典型的多道程序系统中，进程通常有五种状态，分别是：创建、就绪、运行、阻塞和消失（完成）状态。进程从创建到消失经历的阶段如图 2.1

所示。

例 2-6　图 2.4 所示的状态转换图中，有两个挂起状态。假定现在操作系统要调度一个进程，且存在就绪状态和就绪挂起状态的进程，而且至少有一个处于就绪挂起状态的进程拥有比任何处于等待状态的进程具有更高的调度优先级，两种极端的策略是：

(1) 总是调度一个处于就绪状态的进程以使交换达到最少；

(2) 总是优先考虑具有最高优先级的进程，即使这样可能会引起不必要的交换。

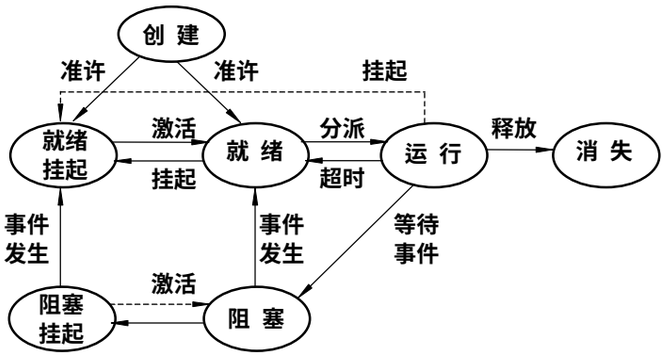


图 2.4

请提出一种折衷策略以平衡优先级与执行情况的关系。

解　对于挂起就绪的进程，在对其进行调度并比较优先级时采用

参加比较的优先级＝实际优先级－常数 C

当 C=0 时，系统就会按照策略 2 调度；

当 C=无穷大时，系统就会按照策略 1 调度。

所以，可使 C 取一个中间值，以平衡两策略的影响。

例 2-7　在某一时刻，一个进程可以因等待某一事件而在一个事件的等待队列中。

(1) 想让一个进程同时等待不止一个事件，这可能吗？请举例说明。

(2) 在这种情况下，该如何修改图中的队列结构，以支持这一新的特点？

解　(1) 当进程进行多个 I/O 时，就可能等待不止一个事件。

(2) 设置等待事件计数器 Counter，当等待事件到达时修改它，如果 Counter=0 就把该进程移进就绪队列，反之则继续等待。

例 2-8　请描述在当前运行进程状态改变时，操作系统进行进程切换的步骤。

解　进程切换的步骤如下：

(1) 保存处理器内容。

(2) 对当前运行进程的 PCB 进行更新，包括改变进程状态和其他相关信息。

(3) 将这个进程的 PCB 插入到适当的队列(就绪、因事件阻塞、就绪挂起等)。

(4) 挑选其他进程执行。

(5) 对挑选的进程 PCB 进行更新，包括将其状态改为运行。

(6) 对存储器管理数据结构进行更新。

(7) 恢复被选择的进程在上次移出时的处理器状态。

例 2-9　如图 2.5 所示的是环保护结构。事实上，简单的内核/用户方案，正如课本中 2.3 节描述的那样，是一种双环结构。Silberschatz 和 Galvin 指出这种结构的一个问题：

这种环(层次)结构的主要不足在于，其不能允许我们坚持需知原则。特别地，如果在域 D_j 中的某对象必须是可访问的，而在域 D_i 中则不行，那么一定有 $j<i$，但这就意味着每个在 D_i 中可访问的段在 D_j 中也可访问。

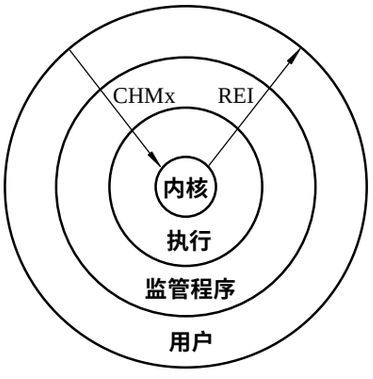


图 2.5　VAX／VMS 存取模式

(1) 请更清楚地解释上面引语中所提到的问题。

(2) 请给出一种使得操作系统能够解决此问题的方法。

解　(1) 这段话的含义是说：为了系统的安全，用户不能随意访问靠内层的对象，但是运行在内层环上的程序却可以随意访问用户的对象，其访问权限过大，对用户的隐私权造成威胁。

(2) 限制某些层次上程序的访问权限，除非经过同意或发生意外情况，不能随意访问用户的数据。

例 2-10　在很多早期的计算机中，由寄存器值引起的中断被存储在与给定中断信号有关的固定位置。试问在何种环境下，该技术是一种实用技术？请解释为什么这种技术通常是不方便的？

解　这种中断分配技术是一种固定的中断分配技术，在这种方式下，中断处理软件总是到固定的地方去读取中断号。它适用于中、小型的系统并且该系统中引起中断的设备是固定不变的，不会经常发生变更，所以，可以为每一个设备分配一个固定的中断号。

当系统中的中断设备通常会发生变更，如：存在移动设备时，就需要采取一种动态分配中断号的方式来解决这个问题，所以题目中的方法就不适合了。

例 2-11　Unix 并不适合实时应用，因为在内核模式下执行的进程不可能被抢占。请对此加以详细描述。

解　在实时系统中不仅要求某个任务在规定的时间内完成，而且任务有强弱之分，强任务是急需处理的，不在规定时间内完成就会造成巨大损失的任务，所以就需要有抢占机制以保证紧急任务优先处理。从这一点上讲，Unix 系统无抢占调度算法，不适合做实时系统。

2.3　自 测 练 习

一、填空题

1. 进程的五个基本特征是_____、_____、_____、_____和_____。

2. 进程的五个基本状态是_____、_____、_____、_____和_____。

3. 进程映像通常包括_____、_____、_____和_____。其中，_____含有进程的描述信息和控制信息，是进程映像中最关键的部分。

4. Unix 系统中的进程控制块由两个结构来实现，这两个结构分别是_____结构和_____结构。

二、判断题（正确的在括号中记√，错误的记×）

1. 为避免出现内存中进程全部处于阻塞状态的情况，操作系统可选择一些进程转移到磁盘，再调入新进程运行。（　　）

2. 用户程序部分包含着进程具体执行的指令，因而是进程映像中最重要的部分。（　　）

3. 操作系统对进程的管理和控制主要是通过控制操作来实现的。（　　）

4. 进程完成后向操作系统发出消息，等待操作系统将其置为停止状态，并在操作系统进行垃圾回收时将其撤消。（　　）

三、问答题

1. 什么是进程？进程和程序的区别是什么？

2. 在一进程中使用多线程的两大优点是：

(1) 在已存在的进程中创建一新线程所需的工作量要比创建一新进程的工作量少；

(2) 同一进程内各线程间的交流被简化了。

那么，是否在同一进程中的两线程之间的上下文切换要比在不同进程中的两线程间的上下文切换所需工作量也要少呢？

3. 进程有哪些基本状态？进程在什么情况下进行各个状态间的迁移？

4. 在 Unix System V 中，表示进程的数据结构是怎样的？把 proc 和 user 结构分开设置有什么好处？

5. 一进程完成其任务并退出后，需要何种基本操作来清除它并继续另一个进程？

四、设计题

试就多进程调度与进程间的消息传递设计一种简单、可行的数据结构。

2.4　自测练习答案

一、填空题

1. 进程的五个基本特征是：动态性、并发性、独立性、异步性和结构性。

2. 进程的五个基本状态是：运行、就绪、阻塞、创建和消失。

3. 进程映像通常包括：用户程序、用户数据、系统栈和进程控制块。其中，进程控制块含有进程的描述信息和控制信息，是进程映像中最关键的部分。

4. Unix 系统中的进程控制块由两个结构来实现，这两个结构分别是 proc 结构和 user 结构。

二、判断题

1. √　　2. ×　　3. √　　4. ×

三、问答题

1. 略（参见知识要点和例题解答）。

2. 同一个进程中的线程之间的切换所需的工作量较少，因为同一个进程中的线程的切换不引起进程的切换，而不同进程的线程的切换会引起进程的切换，线程不占有资源，相对与进程切换，开销要小得多。

3. 略（参见知识要点和例题解答）。

4. 略（参见知识要点和例题解答）。

5. 一个进程结束后操作系统要回收其 PCB，然后释放其占有的所有资源。这些操作完成后，系统会从就绪队列中调度一个进程，分配 CPU 给它使之继续运行。

四、设计题

一种可供选用的数据结构如图 2.6 所示。

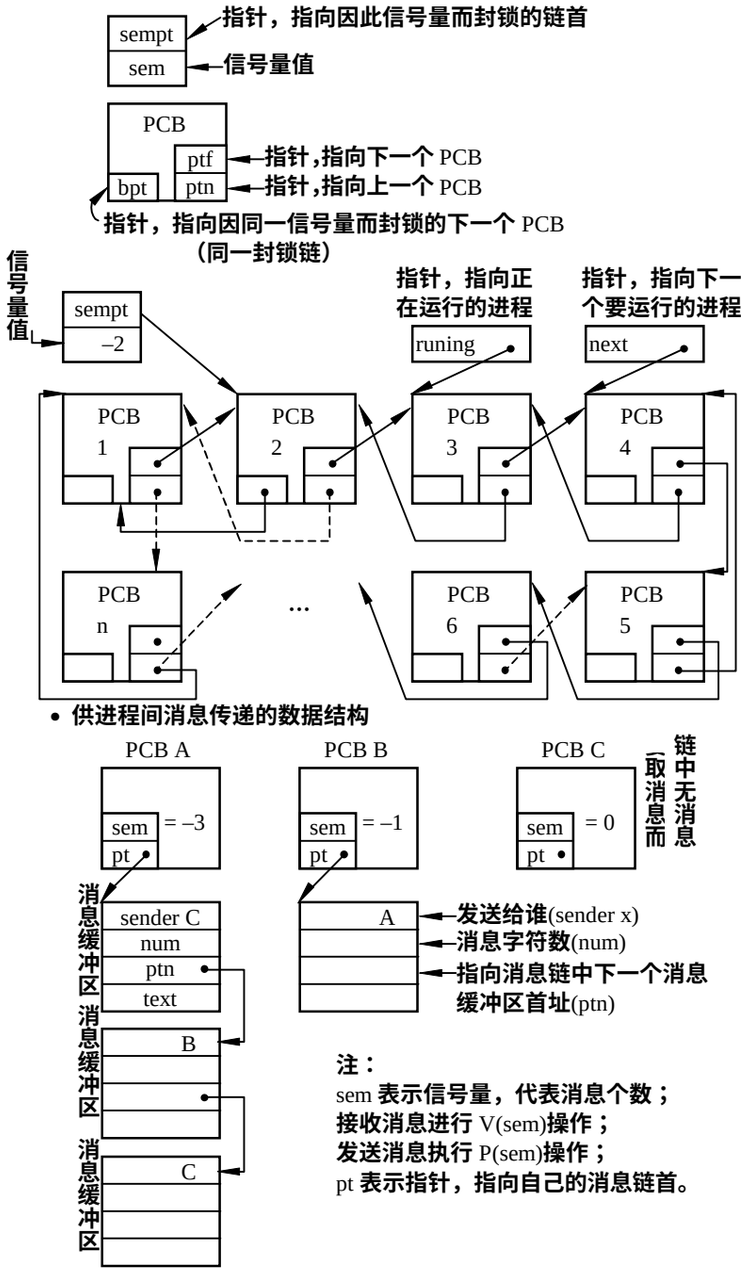

sempt
sem
指针，指向因此信号量而封锁的链首
信号量值
PCB
ptf
bpt
ptn
指针，指向下一个 PCB
指针，指向上一个 PCB
指针，指向因同一信号量而封锁的下一个 PCB
（同一封锁链）
信号量值
sempt
–2
指针，指向正在运行的进程
runing
指针，指向下一个要运行的进程
next
PCB 1
PCB 2
PCB 3
PCB 4
PCB n
PCB 6
PCB 5
• 供进程间消息传递的数据结构
PCB A
PCB B
PCB C
sem = –3
pt
sem = –1
pt
sem = 0
pt
链中无消息（取消息而
消息缓冲区
sender C
num
ptn
text
B
C
A
发送给谁(sender x)
消息字符数(num)
指向消息链中下一个消息缓冲区首址(ptn)
注：
sem 表示信号量，代表消息个数；
接收消息进行 V(sem)操作；
发送消息执行 P(sem)操作；
pt 表示指针，指向自己的消息链首。

图 2.6　消息数据结构

第 3 章

并 发 控 制

3.1 重点与难点

3.1.1 概述

并发包含了大量的设计问题，包括进程间的通信、竞争和共享资源以及多个活跃进程的同步和处理机时间的分配等。这些问题不仅存在于多进程系统和分布式进程系统中，也存在于单处理机多道程序的系统中。

支持并发的基本要求是互斥，即在任一时刻，只能有一个进程访问某一给定的资源或执行某一给定的函数，互斥可用来解决一些冲突，例如：解决竞争资源。同时也可以用于进程同步，使进程能够协同工作。

用软件方法实现互斥的主要缺点是，它们都需要用到“忙碌-等待”（busy-waiting）技术。信号量机制克服了这个不足，它是在操作系统内部实现并向应用程序提供支持，可用来解决各种同步问题，而且能够有效地实现。另一种解决互斥的常用技术是消息传递，它不仅能方便地实现互斥，还可用于进程间的通信。消息传递可以用许多不同的方式来定义。

并发控制是多道程序系统的核心机制，很重要。对于本章的学习有以下要求：首先，要掌握并发原理中进程相互作用、竞争、合作的机制和相关概念，理解互斥的要求及原因；其次，理解实现互斥的各种方法，以及他们的层次和相互关系，会应用各种方法解决问题，这是本章的重点；最后，掌握两类同步互斥问题——生产者/消费者问题和读者/写者问题及它们的联系和差别。

3.1.2 知识要点

1. 并发描述

程序的各个语句之间存在某种优先制约关系，这种关系可用优先图描述。优先图是有向非循环图，可形象地表示这些优先制约关系。两个相继的语句 S_i 和 S_j 并发地执行，并产生与顺序执行相同的结果，必须满足 Bernstein 条件：

① $R(S_i) \cap W(S_j) = \{\}$

② $W(S_i) \cap R(S_j) = \{\}$

③ $W(S_i) \cap W(S_j) = \{\}$

描述并发执行的方式有：Fork/Join 结构、并发语句。

Fork 语句导致语句的并发执行，Join 语句使若干并发计算重新合并成一个计算。并发语句 parbegin S_1；S_2；…；S_n parend 表示位于 parbegin 和 parend 之间的所有语句都可以并发执行。从模拟优先图的角度看，Fork/Join 语句比并发语句的功能强，单独的并发语句不足以实现所有的优先图，但是，其他机制（如信号量机制）的加入使其具有能匹配优先图的功能，并且，并发语句易于加入到块结构的高级语言中，并具有其他结构化控制语句的优点。

2. 并发原理

进程是一个具有独立功能的程序，是代码与相关的数据集在处理机上的执行过程，它具有顺序性、动态性、并发性、独立性和异步性等特征。

优先图展示了优先关系，而进程图展示了进程的创建关系。当操作系统运行时那些不终止的进程称为静态的，可以终止的进程称为动态的。若一个系统仅由有穷个静态进程所组成，则其对应的进程图是静态的，不会改变。动态进程方案比静态进程方案更灵活，但它需要更多的开销，因为进程的创建和删除的代价可能是十分昂贵的。

在单处理机多道程序系统中，进程的并发执行方式是插入执行方式，表面上看起来，进程是同时执行的。在多处理机系统中并发执行方式有插入执行和重叠执行，请参看课本[1]中的图 3.1。

进程之间常常相互作用，存在某种彼此依赖或相互制约的关系：同步和互斥关系。根据进程意识到其他进程的存在程度不同，可将进程间的相互作用划分为：竞争关系、通过共享合作关系和通过通信合作关系。这些相互作用的关系在进程间的察觉程度、相互影响和潜在的控制问题等方面各有不同，详细信息可以参考

课本[1]中的表 3.1。

3. 互斥的要求

系统中一次仅允许一个进程使用的资源称为临界资源。若一个进程正在使用临界资源，则其他欲占用者必须等待，仅当使用者释放该资源后，等待的进程才可使用它。这种若干进程因竞争临界资源而不得不互斥地执行的现象，称为进程的互斥。访问临界资源的那段程序称为临界段。临界段的执行在时间上是互斥的。

解决互斥问题必须满足以下三个要求。

① 互斥执行：当某个进程在其临界段执行时，其他进程就不能在其临界段中执行，并且进程间互不影响对方的执行。

② 空闲让进：若没有进程在临界段中执行，且有某些进程希望进入临界段，则应在有限时间内让其中之一进入临界段。

③ 有限的等待：某个进程在发出进入其临界段的请求后，到该要求被满足之前，存在一个有限的时间界限，在这段时间内允许其他进程进入其临界段。

4. 互斥的软件实现方法

软件方法对并发进程不提供任何支持，因此，无论是在系统程序中还是在应用程序中，进程都要同其他进程合作才能解决互斥。软件方法易引起较高的进程负荷及较多的错误，但有利于深刻理解并发的复杂性。

对实现互斥的软件方法已进行了大量的探索，提出了许多著名算法，如：Dekker 算法、Peterson 算法等。其基本思想是基于内存访问级别的一些基本的互斥，无需要硬件、操作系统和程序设计语言的任何支持，采用 busy-waiting 技术解决互斥问题。

5. 互斥的硬件解决方法

硬件方法通过特殊的机器指令来实现互斥，可以降低开销。主要方法有：

(1) 禁止中断

在单处理机中，禁止进程被中断即可保证互斥，通过操作系统内核定义的禁止和允许中断的原语就可获得这种能力。进程执行临界段时不能被中断。

这种方式的优点是：在单处理机中可保证互斥。缺点是：代价较高，执行效率显著降低；在多处理机系统中，禁止中断不能保证互斥。

(2) 特殊的机器指令

在多处理机系统中，多个处理机共享一个共同的主存，这里并没有主/从关系，也没有实现互斥的中断机制。许多系统都提供了一些特殊的硬件指令，允许我们在一个存储周期内去测试和修改一个字的内容（testset 指令），或者交换两个

字的内容（exchange 指令）等等。这些特殊指令可以用来解决临界段问题。关于这种方式的实现可以参考课本[1]中 3.3.2 的代码。

这种方式的优点是：可用于含有任意数量进程的单处理机或共享主存的多处理机；比较简单，易于验证；可支持多个临界段，每个临界段用各自的变量加以定义。缺点是：采用 busy-waiting 技术，进程等待进入临界段时耗费处理机时间；可能产生饥饿；可能产生死锁。

总之，软件方法和硬件方法实现互斥都存在缺陷，需要寻求其他机制。

6. 信号量

1965 年，Dijkstra 提出了一种新的同步机制——信号量机制。信号量是一个整型变量，除对其初始化外，它可由两个不可中断的 P、V 操作存取。

信号量上的操作如下。

① 信号量的初始化：值为非负。

② P（或 wait）操作：减小信号量的值，若信号量的值为负，则执行 P 操作的进程被阻塞（block）。

③ V（或 signal）操作：增加信号量的值，若信号量的值非正，则被 P 操作阻塞的进程此时可以解除阻塞。

信号量原语定义可以参考课本[1]中的图 3.6 和图 3.7。

在 P/V 操作中，对相应信号量的整型值的修改是不可中断地执行的。信号量机制可以用于解决各种同步问题。引入信号量之后，并发语句的功能同 Fork/Join 结构的一样强。

基本原则：两个或多个进程可通过单一的信号量展开合作，即进程在某一特定的地方停止执行，直到某个特定的信号量到来为止。通过信号量，任何复杂的合作要求都可被满足。

信号量为实现互斥和进程间的协调提供了一个功能强大而灵活的工具，然而使用信号量来编制正确的程序是困难的。因为 P（wait）和 V（signal）操作可能遍布整个程序，并且很难看出它们所引起的全部后果。

7. 管程

管程机制提供了与信号量机制相同的表达能力，但它更容易控制。管程是一种软件模块，包括一个或多个过程、初始化语句和局部数据。具有以下特点：

① 只能通过管程中的过程而不能用其他外部过程访问其局部数据变量。

② 进程通过调用管程的过程而进入管程。

③ 为确保互斥，一个管程中一次只能有一个进程是活跃的，任何其他调用管程的过程都将被挂起直至管程可用为止。

管程机制是一种自动提供适当同步方式的机制。它拥有同步手段，可使用条件变量支持同步，这些变量保存在管程中并且只能在管程内部访问。

操作条件变量的函数：

cwait(c)　调用进程在条件 c 上挂起，管程可被其他进程使用。

csignal(c)　在条件 c 上被 cwait 挂起的进程再次执行。

管程机制和信号量机制的比较：

① 管程本身的结构就实现了互斥，而对于信号量机制而言，实现互斥和同步都是程序设计者的责任。

② 管程的 cwait/csignal 操作与信号量机制的操作不同，如果管程中的进程发信号并且没有一个任务等待条件变量，则这个信号就将丢失。

8. 进程间的通信

进程有如下通信方式。

(1) 共享存储器（shared-memory）

通信的进程共用某些变量，进程通过这些共用变量来交换信息。提供通信的职责在程序设计者。

(2) 消息传递（message-passing）

消息传递的功能允许进程彼此通信而不需要借助共享变量。提供通信的职责在操作系统本身。消息传递能在分布式系统、共享存储器的多处理机系统以及单处理机系统中实现。这种方式提供两种基本的操作：send(destination, message)和 receive(source, message)。

1) 消息寻址方式

- 直接寻址。

发送方：send 原语中明确标明了目的进程。

接收方：预先明确源进程，利于协同；预先不明确源进程，隐含寻址。

- 间接寻址。

消息被送到信箱，接收方从信箱中取回消息。进程和信箱的关系可以是静态的或动态的。信箱的所有权可以归一个进程所有，也可以归系统所有。

2) 同步方式

- 阻塞发送，阻塞接收。要求进程间严格同步。
- 无阻塞发送，阻塞接收。发送进程可向多个目的地发送消息，接收进程在消息到达前被阻塞。
- 无阻塞发送，无阻塞接收。发送和接收方都不需要等待。

3) 通信链容量

- 无容量（0 容量）。通信链不能暂存任何消息。

- 有界容量。队列的长度为 n，至多可存放 n 条消息。
- 无界容量。队列长度无限，可存放任意多个消息。

4) 消息类型

- 固定容量消息。物理实现较简单，程序设计任务困难。
- 可变容量消息。物理实现较复杂，程序设计任务简单。
- 带类型消息。信箱与特定类型的消息相关。

5) 排队规则

- 先进先出方式。
- 按优先级方式。

6) 异常情况

- 进程终止。系统可终止或通知相关进程。
- 丢失消息。操作系统检测并重发消息。
- 受干扰消息。操作系统重发消息或通知相关进程。

9. 读者/写者问题

readers/writers问题的定义如下：一些进程共享一个数据区，readers进程只能读数据区中的数据，而writers进程只能写。并且该问题还要求：任意多个readers可以同时读；任一时刻只能有一个writers可以写；如果writers正在写，那么readers就不能读。

从定义上可以看出读者/写者问题不同于一般的生产者/消费者问题，生产者/消费者问题也不是读者/写者问题在仅有一个读者和写者时的特例。一般的解决办法速度太慢，所以要寻求更优化的算法。以下是两种解决办法。

① 读者优先：一旦有一个读者访问数据区，只要还有一个读者在进行读操作，读者就可以保持对数据区的控制，这就容易导致 writers 饥饿。

② 写者优先：只要有 writers 申请写操作，就不允许新的 readers 访问数据区。

3.1.3 系统实例

1. Windows NT

Windows NT 将线程的同步作为对象结构的一部分。Windows NT 用来实现同步的机制是同步对象家族，包括进程、线程、文件、事件、事件对、信号、记时器和变体。前三个对象类型除用来支持同步外，还有其他用途；其余对象类型是专门为支持同步而设计的。

每个同步对象都可以处于信号态，或者是非信号态。线程可在对象的非信号态上挂起；当对象进入信号态时，该线程被释放。

事件对是与客户-服务器相互作用的对象，它只能被该客户线程和该服务器线程访问。客户或服务器可以将对象置为信号态，从而引起其他线程传送信号。

变体对象用来实现对资源的互斥访问，它一次只允许一个线程对象访问资源。当变体对象进入信号态时，只释放一个等待该变体的进程，在对象进入同步态时释放所有等待线程。

2. Unix System V

Unix 提供了许多机制来进行进程间的通信和同步。管道、消息和共享内存提供了一种在进程间交换数据的方法，而信号量和信号用来触发某些行为。

Unix 对操作系统的发展最有意义的贡献之一就是设置了管道。管道就是一个环状缓冲区，它允许两个进程以生产者/消费者的模式进行通信。因此，它是一个先进先出队列，由一个进程写，另一个进程读。操作系统实施互斥机制，一次只有一个进程能访问管道。

消息就是有相应类型的一个文本段。Unix 提供 msgend 和 msgrw 操作来请求进程传递消息，每个过程都带有一个消息队列，相当于一个邮箱。一个进程在向一个满队列发送消息时，就被挂起。进程读空队列时也会被挂起。如果一个进程要读某一类型的消息而失败了，则它不会被挂起。

Unix 中进程通信的最快方法是共享内存。共享内存是一个被大量进程共享的虚拟内存块。互斥不是共享内存的一部分，它必须由使用共享内存的进程来提供。

Unix 中的信号量系统请求是 wait 和 signal 原语的一般化，可以同时执行几个操作，并且增减值可以大于 1。一个信号量由信号量的当前值、操作信号量的上一个进程的进程 ID、等待比当前信号量值要大的进程数、等待信号量值为 0 的进程数等四个元素组成。信号量带有挂起在该信号量上的进程队列。信号量在进程同步和合作方面相当灵活。

信号就是告诉一个进程发生了同步事件的软件机制。进程间可以相互发送信号，内核可以在内部发送信号。

3. MVS

MVS 根据所用的资源提供两种实行互斥的机制：排队和加锁。排队用在用户控制的资源（如文件）上，而加锁是对于 MVS 系统资源而言的。

排队机制用来控制对共享数据资源的访问。如果资源是只读类的，进程就请求共享控制；如果可以修改数据，就要请求互斥控制。除越权访问外，用户可以规定资源的可用性要被测试，而资源控制不需要测试，或者用户可以规定只有在

资源立即可用时才将资源控制权分配给请求者。这可以用来预防死锁，或者通知操作员出现了问题。

MVS 对系统资源的访问采取加锁方法来保证互斥。一个锁就是公用的虚拟存储器中的一块区域，它通常永久地被保留在主存中。它包含一些用来指示该锁是否是被用的位，如果该锁被用，就表明资源被占用。为预防死锁，可按分层结构来组织锁，处理器只能请求比它当前所拥有锁更高层的锁，用来防止循环等待条件的发生。

4. Linux

信号机制是 Unix 系统使用最早的进程间的通信机制之一，主要用于向一个或多个进程发异步事件信号，信号可以通过键盘中断发出，也可由进程访问虚拟内存中不存在的地址这样的错误来产生。信号机制还可以被 shell 用来向它们的子进程发送作业控制命令。并不是系统中的每个进程都可以向其他进程发消息的，只有内核和超级用户可以做到。信号产生后，并不立即提交给进程，而是必须等到进程再次被调度运行时。

在 Linux 系统中，管道用两个指向同一临时性 VFS 索引结点的文件数据结构来实现。一个例程用于写管道，另一个用于从管道中读数据。从一般读/写普通文件的系统调用的角度来看，这种实现方式隐藏了下层的差异。当写进程执行写管道操作时，数据被复制到数据页面中；而在读进程读管道时，数据又从共享的数据页面中复制出来。Linux 必须对管道进行同步访问，使读进程和写进程步调一致。为了实现同步，Linux 使用锁、等待队列和信号量这三种方式。

Linux 支持 Unix System V 的 IPC 机制。Linux 支持最早的在 Unix System V 中出现的三种进程通信机制：消息队列、信号量和共享存储器。消息队列允许一个或多个进程向队列中写入消息，然后又允许一个或多个进程从队列中读出消息。信号量是内存中的一个区域，它的值可以被多个进程执行原子性操作，即该操作是不可中断的。信号量存在死锁的问题：当一个进程进入关键段，改变了信号量的值后，若因进程崩溃或被中止等原因而无法离开关键段时，就会造成死锁。共享存储器允许一个或多个进程通过在它们的虚拟地址空间中同时出现的存储器进行通信。

3.2 例 题 解 答

例 3-1 什么是优先图？什么是进程图？二者有何区别？

解 优先图是一种有向无环图，其结点对应于独立的语句（组），从结点 i

到结点 j 的有向连线表示语句 i 必须在语句 j 开始执行之前完成。进程图可看做是一种（有向）树结构，其中每个结点是一个进程。从上个结点指向另一结点的有向连线指明前者创建了后者。优先图表征语句之间的优先制约关系，而进程图则展示进程的创建关系。在一个进程图中，P_i 到 P_j 的边并不隐含 P_i 只能在 P_j 之后执行，而是表示 P_i 创建 P_j，P_i 和 P_j 可以并发地执行。

例 3-2 简述 fork 和 join 操作，为什么 join 操作带有一个用来计数的参数？

解 fork L 操作允许当前的语句组继续执行并同时开始执行标号 L 处的语句组。join 操作将两个或多个并发语句组汇合到程序中的某一点，以作为一个进程继续执行下去。join 起了一种延迟作用，以防止任何进程在到达这一点后还试图单独继续执行下去，除非所有相关的语句组（或进程）都到达了这一点，因此，计数是需要的。

例 3-3 为什么应将 join 指令作为一个不可分割的操作来执行？

解 若不这样，则计数器之值可能不正确。例如，计数器 C=3，且有两条并行执行的 join C 指令。这两条指令执行之后，C 之值应为 1，但若 join 不是按不可分割方式执行，那么执行完这两条指令后，C 之值可能为 2。

例 3-4 在 fork/join 机制中为什么需要用 goto 语句？与结构化程序设计方法相比，fork/join/goto 方法有何不足？

解 使用 goto 语句是为了便于 fork 分岔出的语句组将执行流程引向 join 操作。它需要用 goto 语句，这在结构化程序设计方法中是不提倡的，因为毫无约束地利用 goto 语句将使程序难以阅读和调试。

例 3-5 利用以下的语句或指令，将图 3.1 所示的优先图转换成程序形式：

(1) 并发语句；

(2) Fork 和 Join 指令。

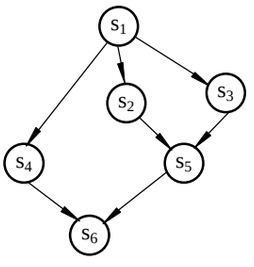


图 3.1

解 (1) 采用并发语句：

```
s1;
parbegin
        s4;
        begin
            parbegin
              s2;
              s3;
            parend
            s5;
```

```
            end
    parend
    s6;
```

(2) **采用** Fork/Join **指令：**

```
    c1:=2;
    c2:=2;
    s1;
    fork L1;
    s4;
    goto L4;
    L1:  fork L2;
            s2;
            goto L3;
    L2:  s3;
    L3:  join c1;
            s5;
    L4:  join c2;
            s6;
```

例 3-6　**利用** parbegin/parend **结构编写一个实现无复制的双缓冲区方案的程序。**

解

```
type item=…;
  var buffer: array[0..n–1] of item;
        in,out: 0..n;
        nextp,nexte: item;
        in:= 0;
        out:=0;
        parbegin
          producer:
            begin
              repeat
                …
                produce an item in nextp
                …
                while (in+1) mod n = out do skip;
                buffer[in]:= nextp;
                in:= (in+1) mod n;
```

```
            until false;
        end;
    consumer:
        begin
            repeat
                while in = out do skip;
                nextc:= buffer[out];
                out:= (out+1) mod n;
                …
                consume the item in nextc
                …
            until false;
        end;
parend;
```

例 3-7 术语 busy-waiting 的含义是什么？busy-waiting 是否是解决进程同步和互斥的 P/V 解决方法的一种特性？

解 busy-waiting **的含义是：一个进程正等待某个事件的发生，而且是通过不断地执行指令来实现等待的。**busy-waiting **不是使用** P/V **操作的程序的一种特性。不过，在某种情况下，它很可能是实现** P/V **操作本身的一部分。**

例 3-8 是否在任何情况下忙碌-等待都比阻塞-等待方法的效率低？试解释之。

解 并非在任何情况下忙碌-等待都比阻塞-等待方法的效率低，例如临界段对低速 I/O **设备进行操作时，这两种方式的效率是一样的。也就是说当处理机的处理能力有非常大的余地时，这两种方法是一样的，甚至忙碌-等待方式还能更快地响应，因为不需要把进程从阻塞队列移到就绪队列的操作。**

例 3-9 考虑过河的例子，设计一个过河算法。要求该算法能够确保若干人从同一岸同时过河而不发生死锁。

解 共享变量是：

```
var
    mutex: semaphore {初值=1}
    wait: array[0..1] of semaphore {初值=0}
    counter: array[0..1] of integer {初值=0}
    turn: (0..1);
```

访问数组 wait **和** counter **时，来自** A/B **岸的人分别使用索引** 0/1 **示出。**

```
    P(wait[i])；
```

```
P(mutex)；
counter[i]:= counter[i] + 1;
if    counter[i]=1 and counter[(i+1) mod 2]≠0
    then    begin
                V(mutex);
                P(wait[i]);
     end;
过河
P（mutex）；
counter[i]:= counter[i]–1；
if    counter[i]= 0 and counter[(i+1) mod 2]≠0
   then    V(wait[(i＋1) mod 2])
   else    V(mutelx)；
```

例 3-10　以下程序并发执行，且数据只有在调入寄存器后才能加 1。

```
const n=50;
var tally: integer;
procedure total;
var count:integer;
begin
   for count:=1 to n do tally:=tally+1
end;
begin (*main program*)
   tally:=0;
   parbegin
      total;total
   parend;
   writeln(tally);
end.
```

(1) **给出该并发程序输出的** tally **值的下限和上限。**

(2) **假定并发执行的进程数为任意个，**(1)**问中的结果会作何变化？**

解　(1) **因为** tally **的加** 1 **操作要有三个步骤：调入寄存器、加** 1**、写回内存，所以在并发执行的时候就会产生不一致的现象。**tally **的下限值是** 50**，上限值是** 100**。**

(2) **假设有** n **个进程在同时并发执行，则下限不变，上限为** n×50 **。**

例 3-11　证明 Dekker **算法的正确性：**

(1) **证明它可以实现互斥。**

(2) 证明它可以避免死锁。

解 (1) 当 P_0 置自身 flag[0]为 true 时，P_1 要进入临界段，就要去查看 turn 的值，即使 turn=1 也必须等到 flag[0]变为 false 才行。所以两个进程不可能同时进入临界段，即临界段的执行是互斥的。

(2) 由于有 turn 变量的存在，两个进程的执行就有一个顺序，当没有轮到自己时，就会主动地置自身的 flag 标识为 false，让别的进程进入临界段。所以不会有死锁的情况发生。

例 3-12 面包店算法(bakery algorithm)如下，它是解决互斥问题的另一软件方法。

```
var choosing: array [0..n–1] of boolean;
    number: array [0..n–1] of integer;
repeat
  choosing[i]=true;
  number[i]:=1+max(number[0],number[1],…，number[n–1]);
  choosing[i]:=false;
for j:=0 to n–1 do
  begin
    while choosing[j] do {nothing};
    while number[j]≠0 and (number[j],j)＜(number[i],i) do {nothing};
end;
  〈critical section〉;
    number[i]:=0
    〈remaider〉;
forever.
```

数组 choosing 和 number 的初始值分别为 false 和 0，每个数组中的第 i 个元素只能由进程 i 进行读/写，而其他进程只能读。(a,b)＜(c,d)的定义为：(a＜c) or (a=c and b＜d)

(1) 用自然语言描述该算法。

(2) 该算法是否能避免死锁，为什么？

(3) 该算法是否能实现互斥，为什么？

解 （1）变量 choosing[i]是保护进程 P_i 在设定 number[i]的时候，其他的进程不能读取 number[i]的值，number[i]是用来表明进程进入临界段的顺序。此算法采取的是 FIFO 策略，先到的进程先进入临界段。所以当一个进程 P_i 想要进入临界段时，先把 choosing[i]设为 true，然后设定 number[i]的值，按 FIFO 策略，P_i 要排在最后，因此 number[i]设为 number 数组中的最大值加一。清除 choosing 的

保护，然后逐个访问其他进程的 number，如果写了保护就等待；如果其他进程要进入临界段（number≠0）并且执行顺序在 P_i 前面，则等待其执行完毕。在 P_i 前面的进程执行完毕后，会置 number 值为 0，让 P_i 进入临界段。P_i 执行完临界段后，置自身的 number[i]为 0，让排在其后的进程执行临界段。

(2) 该算法能避免死锁，因为进程按先来先进的顺序进入临界段，同时到来的按照进程号排序，破坏了死锁的条件，因此不会产生死锁。

(3) 该算法能实现互斥，假设进程 P_i 已处在临界段，则在第二个 while 语句处对其他所有进程逐一做了检查，任一进程 P_j 通过检查有两种可能：第一种可能是检查的时候 P_j 的 number 值为 0，即 P_j 还没有要进入临界段，P_j 设定 number 值时会大于 P_i 的 number 值，即 number[i]<number[j]。因为 P_i 进入临界段后，其 number 就有一个值，P_j 在选定 number 值的时候为 number 数组中的最大值加 1，上述关系成立。所以其他后到的进程会在第二个 while 语句处等待 P_i 执行完毕。另一种可能是(number[j],j)＜(number[i],i)不成立，因为这二者不可能相等，所以只有(number[i],i)＜(number[j],j)，P_j 仍会在第二个 while 语句处等待。即任一时刻，不可能有两个进程同时进入临界段。

例 3-13 证明以下实现互斥的软件方法并不依赖于存储器访问一级的互斥执行。

(1) bakery 算法(面包店算法)；

(2) peterson 算法。

解 不要求存储器一级访问的互斥执行，是指对同一存储单元的读/写可以并行。对一个存储单元同时读不会产生不一致的情况，所以，这里只讨论一个进程在写入一个存储单元时，另一个进程同时在读的情况。在多处理器共享内存的系统中，写操作会正确执行，读操作可能读出一个任意值。

(1) bakery 算法：算法中可能产生读出数据不一致的语句是两个 while 语句，在第一个语句中，进程 P 要访问另一个进程 Q 的 choosing 值，若 Q 向 choosing 写入 true，而 P 读出了 false，P 就会进入第二个 while 语句。在第二个语句中，P 要访问 Q 的 number 值，读的时候若 Q 还未设定或已设定好 number 值都不会产生异常，程序仍然按正确的路线执行。若访问的时候 Q 正在设定 number 的值，就可能读出错误的数据。在正确的情况下 number[P]<number[Q]，若错误的数据也满足此关系，则仍然正常执行，即 P 认定 Q 在其后进入临界段，接着执行 for 语句的下一个循环。若数据使此关系错误，P 就会再次执行 while 循环等待，也就是说，数据错误的最大代价是 P 在 while 语句处循环等待至读出正确数据。

(2) peterson 算法：和 bakery 算法的道理一样，若进程 P 和进程 Q 发生读/写冲突，假设 Q 写的时候 P 在读，那么 P 处在 while 语句的循环中，此时 Q 可能还未进入临界段，在设定 flag 和 turn 值，或已出临界段，在设定 flag 值。在前一种

情况下，P 要么继续循环直到读出正确值，做出正确判断为止；要么进入临界段，此时 Q 还未进入临界段，它会在 while 处忙碌-等待。在后一种情况中，Q 已出临界段，P 读出任何值都不会影响程序执行的正确性。

例 3-14 试说明如果 P/V 操作不是不可分割执行的，就会违反互斥性。

解 假定信号量 S＝1，且进程 P_1 和 P_2 并发地执行 P(S)，那么，下面的执行序列就违反了互斥性。

(1) T_0：P_1 判定 S 之值等于 1。

(2) T_1：P_2 判定 S 之值等于 1。

(3) T_2：P_1 将 S 减 1 并进入临界段。

(4) T_3：P_2 将 S 减 1 并进入临界段。

例 3-15 简述 Dijkstra 的并发语句。它可以利用其他的技术来模拟吗？若可以，则如何模拟呢？

解 Dijkstra 的并发语句又称为结构化并发语句，形如 parbegin P_1；P_2；…；P_m parend ，其中每个 P_i(i＝1，2，…， m)可以是语句（组）、过程或子程序（若为复合语句，则需用 begin/end 括住），它们都可以并发执行。结构化的并发语句形式可以采用 fork/join 机制或信号量来模拟。

例 3-16 任何优先图都可以转换成 fork/join 形式的语句组或结构化并发语句的形式吗？试比较 fork/join 语句和并发语句在这方面的功能。

解 任何优先图都可以转换成 fork/join 形式的语句组，但并不是所有的优先图都能转换成结构化的并发语句形式的。从模拟优先图的角度看，fork/join 语句比并发语句的功能强，单独的并发语句不足以实现所有的优先图，但是，其他机制（如信号量机制）的加入使其能匹配优先图的功能。

例 3-17 创建新进程有哪几种实现方式？

解 创建新进程有如下四种方式：

(1) 父进程继续与其子进程并发地执行；

(2) 父进程暂停执行直至其子进程全部执行完为止；

(3) 父进程和子进程共享所有的公用变量；

(4) 子进程只共享父进程变量的一个子集。

例 3-18 试说明管程和进程有何异同点？

解 管程和进程的异同点是：

(1) 二者都定义了数据结构。进程定义的是私有数据结构 PCB，管程定义的是公共数据结构，如消息队列。

(2) 二者都在各自的数据结构上进行有意义的操作。进程是由顺序程序执行有关操作，管程主要是进行同步操作和初启操作。

(3) 二者设置的目的不同。进程是为了更好地实现系统的并发性而设置的，

管程是为了解决进程的公共变量，为了解决共享资源的互斥使用问题而设置的。

(4) 进程通过调用管程中的过程对共享变量实行操作。此时，该过程就如通常的子程序一样被调用而处于被动工作方式，因此，称管程为被动成分。与此相对应的进程则处于主动工作方式而被称为主动成分。

(5) 由于进程是主动成分，故进程之间能被并发执行，然而管程是被动成分，管程和调用它的进程不能并发执行。

(6) 进程可由“创建”而诞生，由“撤消”而消亡，有生命期，管程是操作系统中的固有成分，无需进程创建，也不能为进程所撤消，只能被进程调用。

例 3-19　简述生产者/消费者问题，并给出几个生产者/消费者的简例。

解　在生产者未“生产”出一个结果之前，不允许消费者使用那个结果；若缓冲区全满，则不允许生产者“生产”任何结果，即在消费者未“消费”一个结果之前，不允许生产者“生产”出“新”结果。比如：编译程序/连接程序，连接程序/装配程序，卡片阅读机/行式打印机等都是生产者/消费者的例子。

例 3-20　什么叫临界资源？什么是临界段？在解决临界段问题时必须遵循哪些原则？

解　在计算机系统中，同时有许多进程，它们共享着各种资源，然而，有许多资源一次却仅能为一个进程所使用。把一次仅允许一个进程使用的资源称为临界资源。例如，物理设备属于临界资源，如打印机、磁带机、输入机等。除了物理设备外，还有很多变量、数据、表格、队列等都是由若干进程所共享的临界资源。进程访问共享临界资源的一段程序，叫做临界段。为了防止两个进程同时进入临界段内，必须互斥，即保证每次至多有一个进程进入临界段，因此，访问时通常应遵循下列原则：

(1) 当有若干进程欲进入其临界段时，在有限时间内有一进程进入。换言之，它们不应互相阻塞而致使彼此都不能进入临界段。

(2) 每次至多有一个进程处于临界段。

(3) 进程仅在临界段内逗留有限的时间。

例 3-21　假定希望顺序执行 s_1，s_2，s_3，但 s_2 需互斥执行。请利用信号量机制和 P/V 操作实现这一要求。

解　设 S＝1（初值），程序段为：

```
s1；
P(S)；
s2；
V(S)；
s3
```

例 3-22　将操作在信号量 S 上的 P/V 原语转换成等价的不含 busy-waiting 的

临界域。

解 将信号量 S 描述为一个共享整型量：

```
var  S： shared integer；
```

用如下代码段实现 P(S)和 V(S)操作：

```
P(S)：region S when S＞0 do S := S – 1;
V(S)：region S do S := S + 1;
```

例 3-23 一个二元信号量是其值仅取 0 和 1 的信号量，如何用二元信号量去实现“一般的”信号量？

解 令 S 是一般的信号量，可用下面的程序来实现它：

```
var   S1：binary-semaphore          {初值＝1}
  S2：binary-semaphore              {初值＝0}
  C：integer                        {初值为 S 之值}
P(S)：P(S1)
  C := C–1;
  if C＜0 then
       begin
          V(S1);
          P(S2);
       end
    else V(S1);
  V(S)：P(S1);
  C := C+1;
  if C≤0 then V(S2);
  V(S1);
```

例 3-24 信号量等待表常常是用（按 FIFO 次序管理）队列实现的，它们可用栈来实现吗？这会引起什么问题？

解 等待表可用栈来实现，但这可能导致饥饿现象发生。

例 3-25 一组除信号量外不可能共享任何变量的顺序进程可以彼此通信吗？

解 不能。一进程只可能执行 P 或 V 操作，V 操作不影响执行它的进程，因此，进程不能从 V 操作得到任何信息。一个 P 操作成功后，执行该 P 操作的进程无法得知它是立即成功还是延迟一段时间后才成功的。若该 P 操作从未成功（因无对应的 V 操作）过，那么，执行它的进程就挂起而且不可能做任何事。

例 3-26 定义两个同步原语 ENQ 和 DEQ。其中，r 是一个资源对象，P 是一个进程，queue(r)是等待获得资源 r 的那些进程的 FIFO 队列，inuse(r)是一个布

尔量。

解 ENQ(r)： if inuse(r) then

begin

将 P **插入队列** queue(r);

阻塞 P;

end

else inuse(r) := true;

DEQ(r)： P := head of queue(r);

if P≠nil then **激活** P

else inuse(r) := false;

例 3-27 **在下面的程序中，将以下语句进行交换会产生什么后果？**

(1) wait(e); wait(s)

(2) signal(s); signal(n)

(3) wait(n); wait(s)

(4) signal(s); signal(e)

```
program boundedbuffer;
const      sizeofbuffer =…;
var        s:semaphore (:=1 );
           n: semaphore (:= 0);
           e:semaphore (:=sizeofbuffer);
procedure producer;
begin
  repeat
    produce;
    wait(e);
    wait(s);
    append;
    signal(s)
    signal(n)
  forever
end;
```

```
procedure consumer;
begin
  repeat
    wait(n);
    wait(s);
    take;
    signal(s);
    signal(e);
    consume
  forever
end;
begin (* main program *)
  parbegin
    producer; consumer
  parend
end.
```

解 (1)**交换语句会导致系统死锁，生产者进入临界段时，如果缓冲区已满，就会在** wait(e)**处阻塞，而消费者又无法进入临界段，系统就会死锁。**

(2) **交换这两个** signal **语句不会产生任何不正确的后果。**

(3) **交换这两个** wait **语句会导致系统死锁，消费者进入临界段时，如果缓冲**

区为空，就会在 wait(n)处阻塞，而生产者又无法进入临界段，系统就会死锁。

(4) 交换这两个 signal 语句不会产生任何不正确的后果。

例 3-28 在使用循环缓冲区的生产者/消费者问题中，生产者的 append 和消费者的 take 如下面程序所示。我们只允许缓冲区中最多有 n–1 个数据项。

问：(1) 为什么要这个限制？

(2) 修改该算法以消除该限制。

```
producer                                    consumer：
   repeat                                       repeat
     produce item v;                              while in=out do {nothing};
     while((in+1) mod n=out) do {nothing};          w:=b[out];
          b[in]:=v;                                 out:=(out+1) mod n;
          in=(in+1) mod n                           consume item w
   forever;                                  forever;
```

解 （1）在这种使用循环缓冲区的生产者/消费者问题中，in=out 或 (in+1) mod n=out 表示缓冲区为空或缓冲区为满，此时，in 和 out 指向的缓冲区不能存储数据项，所以只允许最多有 n–1 个数据项。

（2）采用例 3-26 的方法解决生产者/消费者问题，在原方法的基础上加一个信号量 e 跟踪空缓冲区的个数。并把生产者的 append 和消费者的 take 做如下更改。

```
producer                                    consumer:
repeat                                        repeat
   produce item v;                              w:=b[out];
        b[in]:=v;                               out:=(out+1) mod n;
        in=(in+1) mod n                         consume item w
forever;                                    forever;
```

当 in 和 out 指向同一个缓冲区时，如果生产者在 wait(e)处阻塞，则表明此缓冲区中有可供消费者消费的数据项；如果消费者在 wait(n)处阻塞，则表明此缓冲区是空缓冲区，可供生产者使用。

例 3-29 试用信号量实现 ENQ 和 DEQ，要求适当地实现进程间相互作用时所隐含的次序。可以使用另外的数据结构和变量。

解 假定该系统至多支持 N 个并发进程，每个进程附有唯一的整数 id（0≤id≤N–1）。其代码如下：

```
Var   X：array[0..n–1] of   interger;
      Y：array[0..n–1] of   semaphores          {初值＝0}
         in, out：interger;                     {初值＝0}
```

```
        inuse：boolean;                          {初值＝false}
        mutelx：semaphore;                       {初值＝l}
ENQ：P(mutex);
                    if inuse then
                      begin
                        X[in] := id;
                        in := (in+1) mod n;
                        V(mutex);
                        P(Y(id));
                      end
                    else begin
                           inuse := true;
                           V(mutex);
                         end
DEQ：P(mutex);
                    if in≠out then
                      begin
                        t := X[out];
                        out := (out+1) mod n;
                        V(Y(t));
                      end
                    else inuse := false;
                    V(mutex);
```

例 3-30　考虑图 3.2 所示的优先图，回答下列问题：

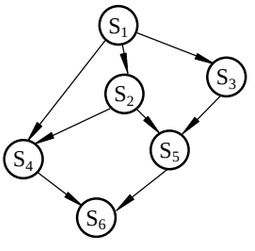


图 3.2

(1) **这个优先图能否仅用并发语句表达？若能，则说明如何表达。**

(2) **如果还可以使用信号量，那么，如何用信号量表达该优先图？**

解　(1) **不能。**

(2) **用信号量表达优先图的代码为：**

```
        var   a,b,c,d,e,f,g,h：semaaphore        {初值=0}
    parhegin
      begin S1; V(a); V(b); V(c); end;
      begin P(a); S2; V(d); V(e); end;
```

```
        begin P(b); S3; V(f); end;
        begin P(c); P(d); S4; V(g); end;
        begin P(e); P(f); S5; V(h); end;
        begin P(g); P(h); S6; end;
      parend
```

例 3-31 一个多元信号量允许 P 和 V 原语同时在若干信号量上操作，在一个不可中断的操作中就可获得和释放几个资源，这样的 P 原语（操作在两个信号量上）可定义如下：

```
P(S,R)：while (S≤1 or R≤1) do skip;
S := S – 1;
R := R – 1;
```

那么，如何用常规的信号量来实现这种多元信号量？

解 可用下面的编码来实现这种多元信号量：

```
var   S1，R1：integer；
 mutex：semaphore；              {初值=1}
 X：semaphore；                  {初值=0}
P(S,R)：  P（mutex）;
                S1 := S1 – 1;
                R1 := R1 – 1;
                if S1<0 or R1<0 then
                  begin
                    V(mutex);
                    P(X);
                  end
                else V(mutex);
V(S,R)： P（mutex）;
                S1 := S1 + 1;
                R1 := R1 + 1;
                if S1≤0 and R1≤0 then V(X);
                V(mutex);
```

例 3-32 假定一个系统支持进程间相互通信，但并未提供信号量，那么，如何编写一个希望使用操作在信号量上的 P/V 原语而不使用 send/receive 的程序？

解 为信号量设立一个信箱。一个信号量上的 P 操作试图从相应的信箱中接收信息；而信号量上的 V 操作则发送信息给相应的信箱。信箱中消息的个数对应于相应信号量之值。

例 3-33 回答以下与公平的理发店问题相关的问题(公平的理发店问题的代

码见课本[1]中的图 3.18)：

(1) 理发师是否必须亲自向其服务的顾客收款？

(2) 理发师是否一直使用同一理发椅？

解 （1）不一定，假设当前仅有一个顾客，则他会在理完发后去唤醒另外两个理发师之一去收钱。

（2）理发师不是一直使用同一理发椅，假如在很短的时间内，先后来了三个顾客，先来的顾客最后一个走，在这三个顾客走了之后，又来了一个顾客，则第三个理发师用第一个理发椅。

例 3-34 下面定义了一个关于分散的固定容量的存储区（如存储器中的页面或磁盘中的块等）的 class。进程执行 acquire(in)可申请空闲页面，执行 release(in)可交还所分配的页面。试扩充 frames，以允许一进程获得 m 个相邻的块。

```
type frames = class
        var free: array[1..n] of boolean;
        procedure entry acquire (var index: integer);
            begin
                for index := 1 to n do
                    if free[index] then
                        begin
                            free[index] := false;
                            exit;
                        end;
                index := –1;
            end;
        procedure entry release (index: integer);
            begin
                free[index] := true;
            end;
        begin
            for index := 1 to n do
                    free[index] := true;
        end.
```

解 可将 frames 扩充为：

```
type frames = class
        var free: array[1..n] of boolean;
        procedure entry acquire (var index: integer，m: integer);
```

```
    var j: integer;
        flag: boolean;
    begin
      for index := 1 to n do
       begin
         flag := true;
         for j := 0 to m do
              if free[(index+j) mod n] then flag := false;
         if flag then
           begin
              for j := 0 to m do
                   free[(index+j) mod n] := true;
              exit;
           end;
       end;
      index := –1;
    end;
  procedure entry release (index: integer，m: integer);
    var j: integer;
    begin
         for j := 0 to m do
              free[(index+j) mod n] := false;
    end;
  begin
    for index := 1 to n do
         free[index] := true;
  end.
```

例 3-35　利用 SCAN 磁盘调度算法，编写一个有关磁盘调度的管程。

解　该管程的定义如下：

```
type diskhead = monitor
    var busy: boolean;
        up: condition;
        headpos, count: integer;
    procedure entry acquire (dest: integer);
        begin
          if busy then
```

```
                    if headpos≤dest
                        then up.wait(dest + count)
                        else up.wait(dest + count + n);
                    busy := true;
                    if dest<headpos
                        then count := count + n;
                        headpos := dest;
                end;
    procedure entry release;
                begin
                    busy := false;
                    up.signal;
                end;
            begin
                headpos := 0;
                count := 0;
            end.
```

例 3-36　下面是实现 C-SCAN 磁盘调度算法的程序，整型量 count 可能溢出，因为它总是增加但从不减少。

```
type diskhead = monitor
  var busy: boolean;
      up: condition;
      headpos, count: integer;
procedure entry acquire (dest: integer);
  begin
    if busy then
     if headpos≤dest then
        up.wait(dest + count)
        else up.wait(dest + count + n);
    busy := true;
    if dest<headpos then count := count + n;
        headpos := dest;
  end;
procedure entry release;
  begin
      busy := false;
```

```
        up.signal;
    end;
        begin
        headpos := 0;
        count := 0;
    end.
```

(1) 能否修改该算法使之保证不会发生溢出；

(2) 假定允许溢出发生，将会发生什么问题？

解 (1) 在过程 acquire 中加入相应的代码去监测 count 的状态，若它变得太大，就不再接收任何调用，且当所有的顾客都被服务过后再置 count 为 0。

(2) 当溢出发生时 count 之值变为 0，其结果是，在一个短时期内，该调度算法将不是按 C-SCAN 次序行事的。

例 3-37 一个文件可由若干不同的进程所共享，每个进程具有唯一的编号。假定文件可由满足下列限制的若干进程同时访问：与并发访问该文件的那些进程相关的唯一编号的总和必须小于 n。试设计一个协调对该文件访问的管程。

解 协调对该文件访问的管程如下：

```
type coordinator = monitor
            var count: integer;
                S: condition;
    procedure acquire (id: integer);
            begin
                while (count + id≥n) do S.wait;
                count := count + id;
            end
        procedure leave (id: integer);
            begin
                count := count – id;
            end;
    begin
            count := 0;
    end.
```

例 3-38 假定用单一结构 await(B)替代管程中的 wait/signal，其中，B 是一布尔表达式，它引起执行它的进程等待直至 B 为真。

(1) 编写一个利用这种方案实现“reader/writer”问题的管程；

(2) 一般说来，这种结构不可能有效地实现，为什么？

(3) **要使** await **语句能有效地实现，需要施加哪些限制？**

解 (1) **利用这种方案实现“** reader/writer **”问题的管程如下：**

```
type reader-writer = class
    var v: shared record
                    nreaders, nwriters: integer;
                    busy: boolean;
                end;
    procedure entry open-read;
      region v do
        begin
          await(nwriters = 0);
          nreaders := nreaders + 1;
        end;
    procedure entry close-read;
      region v do
        begin
          nreaders := nreaders – 1;
        end;
    procedure entry open-write;
      region v do
        begin
          nwriters := nwriters +1;
          await ((not busy) and (nreaders = 0));
          busy := true;
        end;
    procedure entry close-write;
      region v do
        begin
          nwriters := nwriters – 1;
          busy := false;
        end;
    begin
      busy := false;
      nreaders := 0;
      nwriters := 0;
    end.
```

(2) 如果 B 包含形式参数或局部变量，那么，为了唤醒相应的进程，而不得不计算 B，由此所付出的代价以及进程切换的开销都相当高。

(3) 只允许 B 含有全局量。

例 3-39 n 个进程 P_1，P_2，…，P_n 共享一个 CPU。一次最多只可有一个进程使用 CPU。CPU 一旦有空，就可被一个进程使用，当 CPU 被占用时，其他进程必须等待。当多个进程同时等待使用 CPU 时，CPU 一经释放，其中优先权最高的进程就被允许使用它。优先权规定如下：进程 P_i 的优先权为 i(i=1，2，…，n)，优先数较小者的优先权较高。下面是进程 P_i 预定（RESERVE）、使用和释放（RELEASE）CPU 的一个算法：

```
RESERVE(i);
使用 CPU;
RELEASE;
```

试编出其中两个过程 RESERVE(i)和 RELEASE 的程序。

解 过程 RESERVE(i)和 RELEASE 的程序如下：

```
var v: shared record
                    free: boolean;
                    waiting: array[1..n] of boolean;
                end;
procedure entry RESERVE(i: 1..n);
    begin
        region v do
            begin
                if free then free := false
                        else begin
                                waiting[i] := true;
                                await(not(waiting[i]));
                             end;
            end;
    end;
procedure entry RELEASE;
  var k: 1..n;
    begin
        region v do
            begin
                for k := 1 to n do
```

```
                    if waiting[k]
then  begin
                        waiting[k] := false;
                        exit;
                      end;
              free := true;
            end;
        end;
```

例 3-40　有一个仓库可以存放 A、B 两种物品，每次只能存入一件物品（A 或 B）。存储空间充分大，只是要求：–n<A 的件数减 B 的件数<m，其中，n 和 m 是正整数。试用存入 A、存入 B 和 P/V 操作，描述物品 A 和物品 B 的入库过程。

解　设置三个信号量，初值分别为：$S_0=1$**；**$S_A=m-1$**；**$S_B=n-1$**。**

物品 A 入库过程：

```
…
P(SA)
P(S0)
存入 A
V（S0）
V（SB）
…
```

物品 B 入库过程：

```
…
P(SB)
P(S0)
存入 B
V（S0）
V（SA）
…
```

例 3-41　证明管程与信号量的功能等价：

(1) **用信号量实现管程。**

(2) **用管程实现信号量。**

解　(1)**用信号量实现管程。假设** P_i **是管程的任一过程，程序如下：**

```
monitor proof;
  c1…cn: condition;
  mutex: semaphore;
  c1_mutex…cn_mutex: semaphore;
```

```
procedure  Pi(parameter);
  begin
     wait(mutex);
     user's code;
     signal(mutex);
  end;
```

csignal 和 cwait 函数实现如下：

```
procedure  csignal(c);
var c: condition;
  begin
       case  c
             c1: signal(c1_mutex);
             c2: signal(c2_mutex);
             …
             cn: signal(cn_mutex);
             default: drop c;
       end case;
    end;
procedure  cwait(c);
var c: condition;
  begin
       case  c
             c1: wait(c1_mutex);
             c2: wait(c2_mutex);
             …
             cn: wait(cn_mutex);
             default: drop c;
       end case;
    end;
```

(2) 用管程实现信号量。在这里只实现一般信号量机制，其他的信号量也是一样。signal()和 wait()原语分别是管程的一个过程，程序如下：

```
monitor proof;
  type semaphore = record
       count:integer;
       queue: list of process
    end;
```

```
var s: semaphore;
procedure   wait(s);
  begin
    s.count:=s.count–1;
    if   s.count＜0   then
      begin
        将该进程置入s.queue中;
        阻塞该进程;
      end;
  end
procedure   signal(s):
  begin
    s.count:=s.count+1;
    if   s.count≥   then
      begin
        将进程P从s.queue中移出；
        将进程P置入就绪队列中
      end;
  end;
```

3.3 自 测 练 习

一、填空题

1. 当处理机空闲时，进程调度程序从_____中选出一个进程来执行。
2. 优先图展示了语句间的一种_____关系，而进程图展示的是进程的_____关系。
3. 采用软件方法实现互斥需要用到_____技术，降低了系统的效率，而信号量机制则可以克服这个不足。
4. P 和 V 操作原语是在_____上操作的。

二、判断题（正确的在括号中记√，错误的记×）

1. 进程是一段独立的程序。 ( )
2. fork/join 语句可以模拟所有可能的优先图，而且总可以用它来模拟并发语句。 ( )

3. 单独的并发语句可以完成模拟所有的优先图的功能。 ( )

4. 在一个进程图中，P_i 到 P_j 的边隐含 P_i 只能在 P_j 之后执行。 ( )

5. 管程、条件临界域和信号量三者在用它们实现同步问题的意义下是等价的。 ( )

6. 在多处理机系统中，禁止中断不足以保证互斥。 ( )

7. 尽管管程确保了互斥，但其中的过程必须是再入式的。 ( )

8. 信号量等待表如果用栈来实现，则可能导致死锁的发生。 ( )

三、选择题

1. 下面哪些是进程的特征？ ( )

a) 动态性； b) 静态性； c) 并发性；

d) 独立性； e) 异步性； f) 结构特性。

2. 下面哪些是创建新进程的可能的实现方式？ ( )

a) 父进程继续与其子进程并发地执行；

b) 父进程暂停执行直至其子进程全部执行完为止；

c) 父进程和子进程共享所有的公用变量；

d) 子进程只共享父进程变量中的一个子集。

3. 进程间的通信方式是哪些？ ( )

a) 共享存储器； b) 事件触发；

c) 消息传递； d) 过程调用。

4. 管程具有哪些特征？ ( )

a) 只能通过管程中的过程而不能用其他外部过程访问其局部数据变量；

b) 一个管程中一次只能有一个进程是活跃的，任何其他调用管程的过程都将被挂起直至管程可用为止；

c) 进程通过调用管程的过程而进入管程；

d) 可由“创建”而诞生，由“撤消”而消亡，有生命期。

5. 下列关于进程与管程间关系的描述正确的是哪些？ ( )

a) 管程有多个活动进程；

b) 管程有一个活动进程；

c) 进程有多个管程；

d) 外部进程可调用管程中的变量。

四、问答题

1. 什么是临界段？什么是临界段问题？列举出 Dijkstra 在解决临界段问题时所施加的制约。

2. 考虑有三个吸烟者进程和一个经销商进程的系统。每个吸烟者连续不断地做烟卷并吸他做好的烟卷。做和吸一支烟卷需要烟草、纸和火柴三种原料。这三个吸烟者分别掌握烟草、纸和火柴。经销商源源不断地提供上述三种原料，但他只将其中的两种原料放在桌上，具有第三种原料的吸烟者就可做烟卷并吸烟。当桌上的两种原料做完后给经销商发信号，然后经销商再拿出两种原料放在桌上，如此重复。试设计一个使经销商和吸烟者同步的算法。

3. 假定一系统由进程 P_1，P_2，…，P_n 组成，每一进程有一个唯一的优先数。编写一个管程，它将三台同样的行式打印机分配给这些进程并利用其优先数来决定分配的次序。

4. 有 4 个人（即 4 个并发进程 P_1~P_4），分别共享其相邻的两根筷子，如图 3.3 所示。每一个进程，都按下述步骤动作：

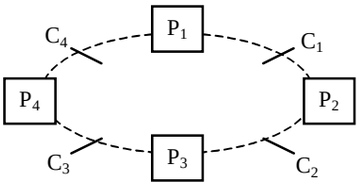


图 3.3

(1) 申请对邻近两根筷子的独占；

(2) 输入和输出；

(3) 释放占用的两根筷子；

(4) 进行计算；

(5) 重复动作(1)。

问：若用下述方法实现，存在什么问题？如何修改，使其克服所存在的问题？

P_i 进程：

```
Repeat
    REQUEST(LEFT[i]);
    REQUEST(RIGHT[i]);
    IN and OUT;
    RELEASE(RIGHT[i]);
    RELEASE(LEFT[i]);
    COMPUTING;
Until false;
```

5. 某车站售票厅，最多可容纳 20 名购票者进入，当售票厅中少于 20 名购票者时，其厅外的购票者可立即进入，否则，需在外面等待。若把一个购票者看做一个进程，请回答下列问题：

(1) 写出用 P/V 操作管理这些并发进程时信号量的初值以及信号量的各种取值的含义。

(2) 根据所定义的信号量，把应执行的 P/V 操作填入下述方框中，以保证进程能够正确地并发执行。

```
procedure Pi      ( i=1, 2, … );
```

```
        begin
            ①  ;
            进入售票厅;
            购票;
            退出售票厅;
            ②  ;
        end;
begin
    parbegin
        Pi    ( i=1, 2, … )
    parend
end.
```

(3) **若欲购票者最多为 n 个人，试写出信号量取值的可能变化范围（最大值和最小值）。**

3.4　自测练习答案

一、填空题

1. **当处理机空闲时，进程调度程序从就绪中选出一个进程来执行。**
2. **优先图展示了语句间的一种优先制约关系，而进程图展示的是进程的创建关系。**
3. **采用软件方法实现互斥需要用到忙碌-等待技术，降低了系统的效率，而信号量机制则可以克服这个不足。**
4. **P 和 V 操作原语是在信号量上操作的。**

二、判断题

1. ×　　2. √　　3. ×　　4. ×
5. √　　6. √　　7. √　　8. √

三、选择题

1. a)、c)、d)、e)、f)　　2. a)、b)、c)、d)　　3. a)、c)
4. a)、b)、c)　　5. b)

四、问答题

1. **临界段就是一次仅允许一个进程执行的程序段。临界段问题是指设计一种算法，以允许一次至多只有一个进程进入临界段且不产生死锁。**Dijkstra **在解决临界段问题时所施加的制约有如下几种：**

(1) **进程的同时执行等价于它们按一种不知道的次序顺序执行；**

(2) **进程的执行速度是彼此无关的；**

(3) **位于非临界段的进程不可能防止其他进程进入临界段；**

(4) **选择并允许一个进程进入临界段的决策不可能被无限期地推迟。**

2. **共享数据结构是：**

```
var   a: array[0..2] of semaphore;        {初值=0}
          agent: semaphore;               {初值=1}
```

关于经销商的代码段：

```
repeat
    Set i,j to a value between 0 and 2;
    P(agent);
    V(a[i]);
    V(a[j]);
Until false;
```

分别用整型量 r **和** s **表示每个吸烟者进程所需要的两种原料（**r **和** s **在** 0～2 **间取值）。**

吸烟者进程的代码段为：

```
repeat
    P(a[r]);
    P(a[s]);
    “smoke”
    V(agent);
until false;
```

3. **管程如下：**

```
type …
    var P: array[0..2] of boolean;
        X: condition;
    procedure acquire (id: integer, printer_id: integer);
    begin
        if P[0] and P[1] and P[2] then X.wait(id);
```

```
        if not P[0] then printer_id := 0
            else if not P[1] then printer_id := 1
                                 else printer_id := 2;
        P[printer_id] := true;
    End;
procedure release (printer_id: integer);
    begin
        P[0]:=false;
        P[1]:=false;
        P[2]:=false;
    end.
```

4. **每当** P_1 **得到其左边的筷子，再申请右边的筷子时，就会发生死锁。用有序的控制分配，可避免上述问题：**

```
var C1，C2，C3，C4: semaphore (初值都为 1)
P1:   P(C1);
      P(C4);
      IN and OUT;
      V(C4);
      V(C1);
P2:   P(C1);
      P(C2);
      IN and OUT;
      V(C2);
      V(C1);
P3:   P(C2);
      P(C3);
      IN and OUT;
      V(C3);
      V(C2);
P4:   P(C3);
      P(C4);
      IN and OUT;
      V(C4);
      V(C3);
```

5. (1) **定义一个信号量 S，其初值为 20，S 取值的含义如下：**

$$\begin{cases} S > 0 & S \text{ 的值表示可继续进入售票厅的人数} \\ S = 0 & \text{表示售票厅中已有 } 20 \text{ 名顾客(购票者)} \\ S < 0 & |S| \text{的值为等待进入售票厅的人数} \end{cases}$$

(2) ① P(S)

② V(S)

(3) S 的最大值为 20，S 的最小值为 $20-n$。

第 4 章

死 锁 处 理

4.1 重点与难点

4.1.1 概述

死锁是由进程间相互竞争系统资源或通信而引起的一种阻塞现象。如果操作系统不采取特别措施，这种阻塞将永远存在，最终可能导致整个系统处于瘫痪状态。因此，死锁问题是操作系统中需要考虑的重要问题。

处理死锁没有一个通行的办法，通常的方法有：死锁预防、死锁避免、死锁检测和死锁恢复。处理死锁的各种方法都是以不同程度地牺牲系统效率为代价的。这些基本方法可以针对个别系统的特点组合使用，从而形成一个最有效的方法。

本章的知识点包括：资源的类型特点、产生死锁的条件、死锁处理的各种方法等相关概念；会用银行家算法解决实际问题；深刻理解哲学家用餐问题的模型本质，会用所学知识解决此问题；了解一些实际系统的知识。

4.1.2 知识要点

1. 死锁概述

资源可分为两大类：可重用资源和消耗型资源。可重用资源在任一时刻只能由一个进程使用而且使用后的资源完好如初，如：处理机、I/O通道、存储器、设备和文件、数据库、信号量等。消耗型资源是可以生产和消耗掉的资源，当资源被进程占用后，就不存在了，如：中断、消息、I/O缓冲区中的信息等。

死锁是多个进程为竞争系统资源或彼此间通信而引起的永久性的阻塞现象。陷入死锁状态的进程称为死锁进程。系统产生死锁有如下四个必要条件：

① 互斥。进程访问的是临界资源，即任一时刻只能有一个进程使用某一资源。

② 占用并等待。进程占用部分资源，并等待得到其他的资源。

③ 非强占。资源只能被占用它的进程自愿释放，而不能被其他进程抢占。

④ 循环等待。存在一个进程环路{P_0，P_1，…，P_n}，P_0 等待着 P_1 占用的资源，P_1 等待着 P_2 占用的资源……P_n 等待着 P_0 占用的资源。

不像并发进程管理中的其他问题，死锁没有一个通行的解决方法。课本[1]中的表 4.1 总结和比较了不同的死锁解决办法。

2. 死锁处理

(1) 死锁预防

根据产生死锁的四个必要条件，设置一些限制来破坏其中至少一个条件，就可防止死锁的发生。为此，可以采用如下三种预防措施。

1) 破坏“占用并等待”条件

采用资源的静态预分配策略，一次申请所有的资源。

优点：

- 简单安全，易于实施；
- 在进程的活动较单一时性能好；
- 无须抢占。

缺点：

- 资源利用率低；
- 启动进程慢，效率低；
- 有“饥饿”现象存在。

2) 破坏“非抢占”条件

一种方法是，若拥有某种资源的进程在申请其他资源时遭到拒绝，则它必须释放其占用的资源，以后若有必要可再次申请上述资源。另一种方法是，当一进程申请的资源正被其他进程占用时，可通过操作系统抢占该资源，此方法在两个进程优先级相同时，不能防止死锁。

优点：

- 对状态容易保留和恢复的资源较为方便。

缺点：

- 实现困难，恢复现场代价高；
- 会导致过多的不必要抢占；

- 易导致循环重启。

3) 破坏“循环等待”条件

采用资源定序方法，将所有资源按类型线性排队，并按递增规则编号。进程只能以递增方式申请资源，因而不会导致循环等待。

优点：

- 资源的申请与分配逐步进行，比预分配策略的资源利用率高；
- 易实现编译期间的检查；
- 无须执行时间，在系统设计阶段问题就已解决。

缺点：

- 要严格限制资源的顺序性，不允许增加资源请求；
- 在使用资源的顺序与系统规定不一致时，资源利用率降低；
- 不能抢占。

对于非共享资源来说，“互斥”条件是必须满足的，以便正确地实现互斥使用临界资源。因此没有通过破坏“互斥”条件来预防死锁的方式。

死锁预防是设法至少破坏产生死锁的必要条件之一（除互斥条件之外），从而消除产生死锁的任何可能性，严格地防止死锁的出现。但方法过于保守，对资源限制严格，使资源利用率和进程执行效率大大降低，它是以降低处理速度为代价的。

(2) 死锁避免

死锁避免并不是严格限制产生死锁必要条件的存在，而只是防止系统进入不安全状态，从而避免死锁的发生。死锁避免算法就是避免系统进入不安全状态的算法。所谓安全状态就是：如果系统能按某种次序为每个进程分配其所需资源，并满足各进程的最大需求而不会造成死锁，则称此时的状态是安全的。银行家（Banker）算法是最著名的死锁避免算法。

系统处于安全状态，仅存在一个安全进程序列$<P_1，P_2，\cdots，P_n>$。若不存在这样的进程序列，则称系统状态 S(t)是不安全的。

死锁避免有两种方法：

1) 避免启动新进程

如果进程对资源的申请可能导致死锁，则不启动该进程。

2) 避免给可能导致死锁的进程分配资源

如果进程对资源的申请可能导致死锁，则不给进程分配该资源。

Banker 算法的主要思想：

① 若进程 P_i 的申请超过了其申报的最大需求数，则报错。

② 若进程 P_i 的申请超过了可用资源数，则 P_i 必须等待。

③ 系统暂时为进程 P_i 分配其所需要的资源，修改资源分配状态。

④ 调用安全算法检查系统当前状态，若会导致不安全状态，则推迟这种分配；否则，实施分配。

安全性算法的主要思想：

① 置向量 Work 的初始值为可用资源向量 Available，向量 Finish 为 false；

② 找一个进程 P_i 使得 $Finish_i$ = false，$Need_i \leq Work_i$，若未找到，则跳转到④；

③ 将 $Allocation_i$ 加到 $Work_i$，置 $Finish_i$ 为 true，跳转到②；

④ 若对所有的进程都有 Finish = true，则状态是安全的，否则是不安全的。

优点：

- 无须抢占；
- 比死锁预防限制少，允许更多的进程并发执行。

缺点：

- 必须知道未来的资源请求信息，要预先声明资源的最大需求量；
- 进程必须是独立的，其执行顺序不受同步要求的约束；
- 资源和进程的数目必须固定；
- 可能导致进程的长时间阻塞。

(3) 死锁检测

死锁检测对访问资源和进程没有任何约束，只要有可能就将资源分配给发出请求的进程。操作系统只是周期地执行检测循环等待条件的算法，判断死锁是否已发生。一旦死锁发生了，就要进行死锁恢复。

依据下面两个因素调整检测算法的使用时机：

① 死锁的发生频度；

② 死锁波及的进程数。

检测时机有如下几种：

① 进程对某资源的请求不能立即得到满足时，进行检测；

② 系统规定合理的检测时间间隔，如一小时一次；

③ 当 CPU 利用率下降到 40%以下时，进行检测。

优点：

- 不会推迟进程的启动；
- 在线处理比较便利。

缺点：

- 丢失了固有的抢占；
- 执行检测算法需要一定的 CPU 开销。

(4) 死锁恢复

一旦检测到死锁，就要立即设法解除死锁。主要有以下两种方法。

1) 撤消进程

撤消一个或多个进程以断开循环等待链，解除死锁。下面是两种可能的方法（其复杂度递增）：

① 撤消所有死锁进程，缺点是代价太高。

② 逐个撤消死锁进程，直至死锁解除为止。撤消顺序根据进程优先数或最小代价原则确定。这种方法要考虑调用死锁检测算法的开销。

2) 挂起进程

使用挂起/激活机制挂起一些进程，抢占其占用的资源以解除死锁，待以后条件满足时，再激活被挂起的进程。需要选择挂起进程、保留现场、重新运行，增加了开销。

选择撤消或挂起进程时应根据下列因素进行：

- 进程的优先级最低；
- 进程已运行的时间最短；
- 进程完成其工作还需要的时间最长；
- 进程已使用的资源数最少；
- 进程已产生的输出量最少；
- 涉及到的进程数最少。

(5) 处理死锁的混合方法

处理死锁的方法有多种，它们各有优缺点，在不同的环境中使用不同的方法就比只用一种方法要好得多。

① 对不同类型资源，使用资源定序的方法。

② 对同一类资源，使用最佳的处理方法。

例如，对于内部资源，采用资源定序方法；对于主存，使用抢占主存的方法；对于作业资源，采用死锁避免方法；对于可交换空间，使用预分法。

3. 哲学家用餐问题

哲学家用餐问题最早是由 Dijkstra 提出并解决的，它是一个死锁和饥饿的典型问题，也是一大类并发控制所面临的问题。

使用信号量解决此问题的一种方法是：每个哲学家先拿其左边的筷子后拿其右边的筷子。用完餐后再将筷子放回原来的位置。这种解决方法将导致死锁。当所有的哲学家都想用餐时，他们都会拿起各自左边的筷子，这时他们的右边就不再有筷子，于是所有的哲学家只有挨饿。

为避免死锁，可以只允许不超过4个哲学家同时用餐。这样就至少有一个哲学家可以得到两根筷子。课本[1]中的图4.8给出了一种使用信号量的解决方法。这个方法可避免死锁和饥饿。

4.1.3 系统实例

系统实例的详细信息可以参考课本[1]中 4.4 节的介绍。

4.2 例 题 解 答

例 4-1 产生死锁的四个必要条件是什么？试以过河问题中产生的死锁为例加以说明。

解 产生死锁的必要条件如下。

(1) 互斥：系统中至少有一个（类）资源必须采用非共享方式。过河问题中，每一块垫脚石任一时刻仅能为一个人占用。

(2) 占用并等待：一个进程至少占用一个资源，并且，正等待得到其他资源，而这些资源已被其他进程所占用。过河问题中，相遇的两个人各自踏在一块垫脚石上，并同时等待踏上对方占用的那一块。

(3) 非抢占：资源不可能被抢占。在过河问题中，由于垫脚石不能强行移动，“非抢占”条件满足。

(4) 循环等待：存在循环等待情况。从东岸来的人等着从西岸来的人从垫脚石上移开脚，而从西岸来的人则等着从东岸来的人从垫脚石上移开脚。

于是，大家都不能过河，每人都等待对方从其占用的垫脚石上移开脚，具备以上四个条件，死锁发生。

例 4-2 产生死锁的四个必要条件是否都是独立的？能否给出一个必要条件的最小集合？

解 四个必要条件并非是绝对独立的。“占用并等待”条件蕴含“互斥”条件；“循环等待”条件蕴含“互斥”、“占用并等待”以及“非抢占”条件。前三个条件是导致死锁的必要而非充分条件，它们潜藏着第四个条件。这四个条件一起构成了死锁的充要条件。

例 4-3 什么是饥饿? 死锁和饥饿的主要差别是什么？

解 饥饿是系统并没有死锁，但至少有一个进程被无限期地推迟。饥饿不同于死锁。死锁是这样一种情形，其中某进程正等待一个决不会发生的事件。而饥饿现象是指某进程正等待这样一个事件，它发生了但总是受到其他进程的影响，以致一直轮不到（或很难轮到）该进程得到 CPU。

例 4-4 设计一个不可能出现饥饿和死锁的过河算法。

解　利用红绿灯和一个计数器来进行设计。当有人开始过河时，计数器增值；过河后该计数器减值。仅当该计数器之值为 0 时才开绿灯，否则为红灯，岸上的人看到绿灯方能过河。

例 4-5　怎样预防死锁发生？常用的方法有哪些？

解　根据产生死锁的四个必要条件，可以设法使其中之一不能成立，死锁就不会出现。因此，死锁预防就是预先排除死锁发生的可能性。常用的方法有：预分法、抢占法和资源定序法等。

例 4-6　试列出两种以破坏“循环等待”条件的方式来防止死锁发生的方法。

解　假定对系统资源已进行了线性定序，则有以下两种破坏“循环等待”条件的方法：

(1) 进程只能申请具有较高序号数的资源；

(2) 当进程申请某个资源时，它必须释放所有较低序号数的资源。

例 4-7　什么是系统资源分配图（SRAG）？SRAG 中如何表示“申请”信息？如何表示“分配”信息？

解　SRAG 是这样一种图，其中展示了进程、资源以及进程和资源之间的关系。例如，哪些（个）进程已占用哪类（个）资源并正申请哪类（个）资源等。从进程向资源框画一条有向连线，指明“申请”信息。从资源框中的某个资源向一进程画一有向连线，指明“分配”信息。

例 4-8　举例说明什么是系统资源分配图（SRAG）？什么是等待图（wait-for graph）？二者的关系如何？

解　在所有资源仅含单一例示的系统中，等待图是从 SRAG 中除去资源类结点并合并相应的边而得到的。等待图中存在边(P_i，P_j)，当且仅当对某个资源 R_p，相应的 SRAG 中包含两条边(P_i，R_p)，(R_p，P_j)。图 4.1 和图 4.2 所示的分别是某系统的 SRAG 和等待图。系统中存在死锁的条件是，当且仅当等待图中出现环路。

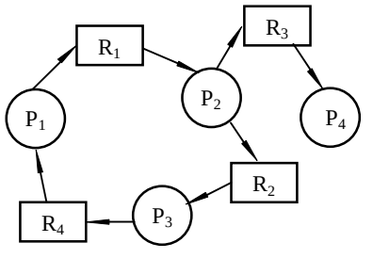


图 4.1

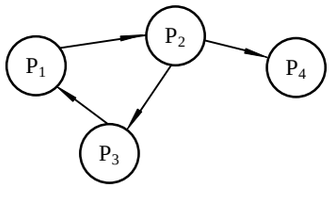


图 4.2

例 4-9　Banker 算法需要哪些数据结构？

解　Banker 算法所需的数据结构有：

(1) **可用资源向量** Available(m)；

(2) **最大需求矩阵** Max(n, m)；

(3) **资源分配矩阵** Allocation(n, m)；

(4) **剩余需求矩阵** Need(n, m)；

(5) **资源申请向量（有关进程** i **的** Request）；

(6) **最大需求向量（有关进程** i **的** Need）。

例 4-10　**试简述** Banker **算法。**

解　Banker **算法是著名的死锁避免算法，其步骤是：**

(1) **若进程** P_i **的申请超过了其申报的最大需求数，则报错；**

(2) **若进程** P_i **的申请超过了可用资源数，则** P_i **必须等待；**

(3) **系统暂时为进程** P_i **分配其所需要的资源，修改资源分配状态；**

(4) **调用安全算法检查系统当前状态，若导致不安全状态，则推迟这种分配；否则，实施分配。**

例 4-11　**试简述安全性算法（**Safety Algorithm）。

解　**请参考本章“知识要点”中“死锁避免”的相应内容。**

例 4-12　**硬件并不总是可靠的。若设备发生故障，那么，它对处理死锁的三种方法（预防、避免和检测）有何影响？**

解　**对死锁预防无影响；对死锁避免有影响，该系统可能不再处于安全状态；对死锁检测有影响，虽不会产生死锁，但必须更新等待图以反映系统状态的变更。**

例 4-13　**如何解除死锁？**

解　**一旦检测到死锁，就要立即设法解除死锁。以下是一些可能的方法，它们以复杂度增加的次序排列。**

(1) **撤消所有的死锁进程，这是操作系统最常用的方法。**

(2) **所有的死锁进程都退回到原来已定义的检测点，然后重新执行所有进程。这要求系统有撤回和重启机制。这种方法的危险性在于原来的死锁可能再次出现，而并发进程的不确定性常使死锁不再发生。**

(3) **逐个撤消死锁进程直至死锁解除为止。撤消进程的顺序根据最小代价原则确定。每撤消一个进程后都要调用检测算法检测死锁是否存在。**

(4) **逐个抢占其他进程的资源直到死锁不再存在为止。与第** (3) **种方法一样，也要根据开销来选择哪个进程被抢占，在抢占之后要执行检测算法。资源被抢占的进程必须撤回到它拥有该资源之前的某一点。**

例 4-14　**在选取撤消的进程或抢占的进程时，涉及到哪些项目？**

解　**在选取撤消的进程或抢占的进程时，应考虑以下因素：**

(1) **优先级；**

(2) 完成一个进程的执行所剩下的时间；

(3) 该进程已经花掉的时间；

(4) 解除死锁所需要的资源类和个数；

(5) 在该次撤消中所涉及到的进程数等。

例 4-15　试列出资源的四种类型，并给出其对应的处理死锁的典型方法。

解　(1) 内部资源：资源定序方法。

(2) 主存：抢占和交换的方法。

(3) 作业资源：死锁避免方法。

(4) 可交换空间：预分法。

例 4-16　适当的 SPOOLing 技术可能消除死锁。它肯定可以消除读卡机、绘图机、行打机等的竞争，甚至对磁带也可以采用 SPOOLing 技术。这样剩下的资源是 CPU 时间、存储区和盘空间。问：是否会出现涉及到这些资源的死锁？若会，那么在什么情况下发生死锁？哪种（些）处理死锁的方法最适合于处理这类死锁？

解　仍然可能出现死锁。例如，进程 P_1 占用一些由进程 P_2 所请求的内存页面，而 P_2 占用 CPU 且 P_1 正在申请 CPU。消除这类死锁的最好方法是“抢占式”方法。

例 4-17　假设三个进程共享相同类型的四个资源，每个进程一次只能申请或释放一个资源，每个进程至多需要两个资源，证明该系统不会发生死锁。

证　假定该系统死锁，那么就隐含其中的每一进程已占用一资源并正等待另一资源。由于该系统只有三个进程且有四个资源，因此，必有一进程能获得两个资源，不必等待。于是该进程不再申请资源，而且当它执行完后将归还它占用的资源。故该系统不会发生死锁。

例 4-18　假设系统中有 m 个同类资源，并被 n 个进程所共享，进程每次只申请或释放一个资源，如果

(1) 每个进程至少需要一个资源，且最多不超过 m 个资源，即对于 i=1，2，…，n，有 $0<Need\leq m$。

(2) 所有最大需求量之和小于 m＋n。

证明该系统不会发生死锁。

证　依题意，对于$\forall i\in(1，n)$，有

$$1\leq Max_i\leq m$$

$$\sum_{i=1}^{n} Max_i < m+n$$

$$\sum_{i=1}^{n} Need_i = \sum_{i=1}^{n} Max_i - \sum_{i=1}^{n} Allocation_i$$

反设系统处于死锁状态，则有

$$\sum_{i=1}^{n} \text{Allocation}_i = m$$

所以，

$$\sum_{i=1}^{n} \text{Need}_i < (m+n) - m = n$$

因此，至少存在一个进程 P_i，有 $Need_i=0$。与题意矛盾，假设不成立。所以，该系统不会产生死锁。

例 4-19　假定某系统处于不安全状态。证明存在这种可能性，即进程完成它们的执行而不进入死锁状态。

解　考虑该系统有如表 4.1 所示的瞬时情况，通过允许 P_1 执行完并将其资源归还到 Available 表列中，系统就可以满足 P_3 的申请。当 P_3 归还其资源时，Available 便包含 2886，因此，允许所有余下的进程以任何次序去完成它们的执行。

表 4.1

进　　程	Allocation	Max	Available
P_1	0　0　1　2	0　0　1　2	1　5　2　0
P_2	1　0　0　0	1　7　5　0	
P_3	1　3　5　4	2　3　5　6	
P_4	0　6　3　2	0　6　5　2	
P_5	0　0　1　3	0　6　5　6	

例 4-20　在一个实际的计算机系统中，资源可以更新和增减，进程可以创建和撤消。如果系统用 Banker 算法处理死锁，那么，在什么情况下，下列改变可以安全地进行而不会引起死锁发生？

(1) 增加 Available（增添新资源）；

(2) 减少 Available（资源永久性地从系统中删除）；

(3) 增大 Max（对一进程而言，它可能希望更多的资源）；

(4) 减少 Max（一进程决定不需要那么多资源）；

(5) 增加进程数；

(6) 减少进程数。

解　(1) 任何时候都不会引起死锁发生；

(2) 仅当每一进程的 Max 请求数不超过可用资源的总数时，系统才保持安全状态；

(3) 仅当每一进程的 Max 请求数不超过可用资源的总数时，系统才保持安全

状态；

(4) 任何时候都不会引起死锁发生；

(5) 任何时候都不会引起死锁发生；

(6) 任何时候都不会引起死锁发生。

例 4-21　考虑一个每月运行 5000 个作业而未采用死锁预防或避免措施的系统。死锁大约每月发生两次，而且每当发生死锁时都要求操作员终止或重新运行大约 10%的作业，每个作业大约耗费 20 元（按 CPU 时间计算）且每个被终止的作业在被撤消时往往都运行了一半。

系统程序员估计，在该系统上配置死锁避免算法（如 Banker 算法）会使每个作业平均执行时间增加大约 10%。由于机器当前仍有 30%的空闲时间，所以，每个月仍可以运行 5000 个作业（尽管平均周转时间增加了大约 20%）。

问：(1) 赞成配置死锁避免算法的理由是什么？

(2) 反对配置死锁避免算法的理由是什么？

解　(1) 为了有效地判定在特定环境下是否出现死锁，必须配置死锁预防或避免算法。通过配置死锁避免算法，减少平均等待时间。

(2) 若不把重点放在使等待时间变为最少这一点上，那么，不配置死锁预防或避免算法会减少成本。

例 4-22　考虑下面的死锁分配策略：允许在任何时候申请和释放资源；若因没有可用资源而不能满足对资源的申请，则检查那些因等待资源而被阻塞的进程；若它们占用所需的资源，则从中收回这些资源并将这些资源分给申请这些资源的进程。增加等待进程正等待的资源向量，使其包含那些被收回的资源。例如，考虑这样一种系统，它有 3 类资源，且向量 Available 的初值为（4，2，2）。若进程 A 申请（2，2，1），则它将得到满足；若进程 B 申请（1，0，1），则它也得到满足；然后若进程 A 又申请（0，0，1），则它被阻塞（无可用资源）。如果进程 C 现要申请（2，0，0），则它得到一个可用资源（1，0，0）和一个曾分配给 A 的资源（因 A 已被阻塞）。此时，A 的 Allocation 向量变为（1，2，1），而其 Need 向量则变为（1，0，1）。试说明：

(1) 在这种环境中会出现死锁吗？若会，试举一例；若不会，那么哪个必要条件不可能成立？

(2) 会出现无限期地阻塞吗？

解　(1) 由于采用的是抢占性策略，破坏了“非抢占”条件，所以不可能出现死锁。

(2) 会的。一个进程可能决不会获得它所需要的全部资源，如果资源被一系列类似过程 C 的申请接连不断地抢占的话。

例 4-23　当发生死锁而撤离一进程时会出现什么困难？

解 该进程必须完全地撤离（中止重新开始）或只是退回到前面的安全态。对于第一种情形，这种撤离的开销由其优先数、已运行的时间、在完成前所需要的时间以及所需的资源类型等因素来决定；对于第二种情形，为了确定进程最近的安全态，系统需要保留有关每一进程的许多信息。

例 4-24 一个系统能否检测它的哪些进程正处于饥饿状态？若能，则说明如何进行这种检测。若不能，则叙述系统是如何处理饥饿问题的。

解 不能，因为检测饥饿现象事实上需要“未来知识”，因为，对进程“过去知识”的统计资料还不能判定进程“前进”与否，但通过使进程“老化”可预防饥饿现象。这意味着对每一进程管理需设置一个撤消计数器，而且在选择进程作为“抢占／撤消”对象时把它作为开销因素的一部分来考虑。

例 4-25 单资源类的 Banker 算法可从一般的 Banker 算法获得，这只需将各数组的维数减为 1 即可。试举例说明，多资源类的 Banker 算法不能通过把单资源类的 Banker 算法单独地应用到每一资源类来实现。

解 考虑一系统，它具有资源类 A，B，C 和两个进程 P_0，P_1 以及表 4.2 所示的瞬时状态。该系统不是处于安全态。但是，若把单资源类的 Banker 算法单独地应用到每一资源类，那么，就得到：

- 序列<P_0，P_1>对资源 A 满足安全性要求；
- 序列<P_0，P_1>对资源 B 满足安全性要求；
- 序列<P_0，P_1>对资源 C 满足安全性要求。

表 4.2

Allocation	Max	Need	Available
1 2 2	2 3 4	1 1 2	1 1 1
1 1 2	2 3 3	1 2 1	

这样，该系统就应处于安全态了，显然与实际情况不符。

例 4-26 假定已经设计好了死锁避免安全性算法，现在希望实现死锁检测算法。问：能否通过简单利用已设计好的死锁避免安全性算法和重新定义 $Max_i = Waiting_i + Allocation_i$（其中，$Waiting_i$ 是一个向量，它指明进程 i 正在等待的那些资源）来实现？

解 在安全性算法中重新定义 Max，可以重新定义 $Need_i = Waiting_i$。这样，Need 就与死锁检测算法中的 Request 相同，而且这些算法是等价的。

例 4-27 设当前的系统状态如下：

available	R_1	R_2	R_3	R_4
	2	1	0	0

process	current allocation				maximum demand				still needs			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	0	2	0	0	1	2				
P_2	2	0	0	0	2	7	5	0				
P_3	0	0	3	4	6	6	5	6				
P_4	2	3	5	4	4	3	5	6				
P_5	0	3	3	2	0	6	5	2				

(1) 计算各个进程的“still needs”。

(2) 系统是否处于安全状态，为什么？

(3) 系统是否死锁，为什么？

(4) 哪些进程可能死锁？

解　(1) 计算结果如表 4.3 所示。

表 4.3

process	current allocation				maximum demand				still needs			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	0	2	0	0	1	2	0	0	1	0
P_2	2	0	0	0	2	7	5	0	0	7	5	0
P_3	0	0	3	4	6	6	5	6	6	6	2	2
P_4	2	3	5	4	4	3	5	6	2	0	0	2
P_5	0	3	3	2	0	6	5	2	0	3	2	0

(2) 不安全，因为没有一个进程可以顺利完成。

(3) 系统不一定死锁，当前系统状态只是不安全，并不代表死锁。

(4) 5 个进程都可能互相等待而死锁。

例 4-28　假设有两种类型的哲学家，一种类型的哲学家总是首先拿其左边的筷子，称为“lefty”，另一种总是首先拿其右边的筷子，称为“righty”，lefty 的行为已在课本[1]中的图 4.7 中定义过，righty 的行为定义如下：

```
beign
  repeat
    think;
    wait (fork [(i+1) mod 5]);
    wait (fork [i]);
    eat;
```

```
            signal (fork [i]);
            signal (fork [(i+1) mod 5])
        forever
    end;
```

证明以下结论(在餐桌上至少各有一名 lefty 和 righty 型的哲学家)：

(1) 任何一种座位排列都能避免死锁。

(2) 任何一种座位排列都能避免饥饿。

解 (1) 从题目和程序中可知，有 5 个哲学家和 5 根筷子，对应 5 个进程和 5 个资源，又因为这 5 个哲学家分为左撇子和右撇子两类，所以哲学家在同一时刻最多需要 9 个筷子。根据例 4-18 的结论，如果每个进程至少需要一个资源，且最多不超过 m 个资源，所有最大需求量之和小于 m＋n，那么系统就不会死锁。在此问题中 9<5+5=10，因此，系统不会死锁。

(2) 由问题（1）的证明，题目中所述的情况不会导致系统死锁，所以任何一个哲学家总能得到他所需要的筷子，从而不会有饥饿的情况发生。

4.3 自 测 练 习

一、填空题

1. 资源可分为 _____ 和 _____ 两大类型，前者包括处理机、I/O 通道、存储器、设备和文件等，后者包括中断、消息和 I/O 缓冲区中的信息等。

2. 通常情况下，进程对资源的利用是按照 _____资源、使用资源、_____资源的顺序进行的，如打开、使用和关闭文件，分配、占用和回收内存空间等。

3. 死锁的四个必要条件是 _____、_____、_____ 和 _____。

4. 处理死锁的方法通常有 _____、_____、_____ 和 _____。

5. 为破坏 _____ 条件，可采用资源的静态预分策略，系统对进程申请的资源进行一次性的分配，然后才启动该进程运行。

6. Banker 算法是典型的 _____ 算法，要求系统必须知道未来的资源请求信息，进程要预先声明资源的最大需求量。

二、判断题（正确的在括号中记√，错误的记×）

1. 死锁就是循环等待。 ( )

2. 不存在只涉及一个进程的死锁。（　　）

3. 在一个 SRAG 中，若不存在环路，则表明不存在死锁。（　　）

4. 在一个 SRAG 中，若存在环路，则表明存在死锁。（　　）

5. 当系统中每一资源类只有一个例示时，系统若存在任何环路，则其状态是不安全的。（　　）

6. 死锁避免比死锁预防对资源的限制更加严格，设置限制条件来破坏产生死锁的必要条件，可消除产生死锁的任何可能性。（　　）

7. 死锁是多个进程为竞争系统资源或彼此间通信而引起的一种临时性的阻塞现象。（　　）

三、选择题

1. 下面哪些资源是不可共享资源？（　　）

a) 只读文件；　　b) 可更新文件；

c) 共享的程序和库；　　d) 打印机；

e) 磁带设备；　　f) 卡片阅读机。

2. 关于产生死锁的现象，下面的描述哪一个最准确？（　　）

a) 每个进程共享某一个资源；

b) 每个进程竞争某一个资源；

c) 每个进程等待着某一个不能得到且不可释放的资源；

d) 某个进程因资源而无法进行下去。

3. 下列方法中哪些可用于中止一个死锁？（　　）

a) 避免互斥数据；

b) 夭折一个进程；

c) 抢占某个进程的资源；

d) 退回原来已定义的检测点。

4. 关于死锁与不安全状态的关系，下列描述正确的是哪些：（　　）

a) 死锁是一种不安全状态；

b) 系统处于不安全状态，一定产生了死锁；

c) 不安全状态是死锁的必要条件；

d) 不安全状态是死锁的充分条件。

5. 在选取撤消的进程或抢占的进程时，应尽量选择哪个：（　　）

a) 进程优先级最高的；

b) 进程已运行的时间最短的；

c) 进程完成其工作还需要的时间最短的；

d) 进程已使用的资源数最少的；

e) 进程已产生的输出量最少的；

f) 所涉及到的进程数最少的。

6. 系统使用的资源，如进程控制块（PCB）一般采用下列哪种典型方法处理死锁？ ( )

a) 预分法；　　b) 抢占和交换的方法；

c) 死锁避免方法；　　d) 资源定序方法。

7. 下列哪些是资源定序方法的特点？ ( )

a) 资源的申请与分配逐步进行，比预分配策略的资源利用率高；

b) 易实现编译期间的检查；

c) 必须知道未来资源请求信息，因为要预先声明资源的最大需求量；

d) 严格限制资源的顺序性，不允许增加资源请求；

e) 不能抢占。

四、问答题

1. 什么是过河问题中的死锁？试说明四个预防算法都可应用于过河问题。

2. 考虑图 4.3 所示的交通死锁情况。

(1) 说明图中导致死锁的四个必要条件成立。

(2) 提出一个避免死锁的简单规则。

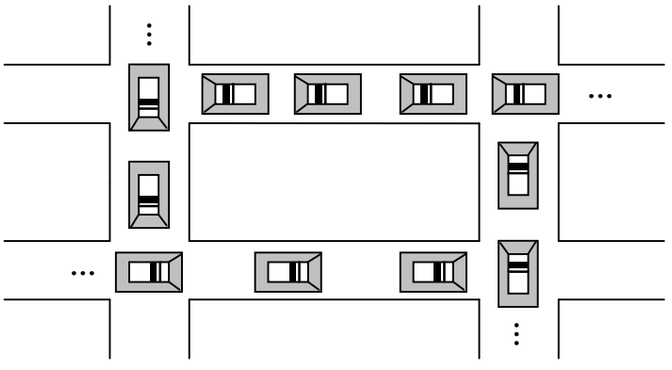


图 4.3

3. 考虑表 4.4 所示的系统瞬时状态，利用 Banker 算法回答下面的问题：

(1) 数组 Need 的内容是什么？

(2) 该系统处于安全态吗？若是，给出一安全序列。

(3) 若进程 P_1 的请求（0420）到达，该请求是否能立即满足？

表 4.4

进　　程	Allocation	Max	Available
P_0	0012	0012	1520
P_1	1000	1750	
P_2	1354	2356	
P_3	0632	0652	
P_4	0014	0656	

4.4　自测练习答案

一、填空题

1. 资源可分为可重用资源和消耗型资源两大类型，前者包括处理机、I/O 通道、存储器、设备和文件等，后者包括中断、消息和 I/O 缓冲区中的信息等。

2. 通常情况下，进程对资源的利用是按照请求（申请）资源、使用资源、释放资源的顺序进行的，如打开、使用和关闭文件，分配、占用和回收内存空间等。

3. 死锁的四个必要条件是互斥、占用并等待、非抢占和循环等待。

4. 处理死锁的方法通常有死锁预防、死锁避免、死锁检测和死锁恢复。

5. 为破坏占用并等待条件，可采用资源的静态预分策略，系统对进程申请的资源进行一次性的分配，然后才启动该进程运行。

6. Banker 算法是典型的死锁避免算法，要求系统必须知道未来的资源请求信息，进程要预先声明资源的最大需求量。

二、判断题

1. ×　　2. √　　3. √　　4. ×
5. √　　6. ×　　7. ×

三、选择题

1. b)、d)、e)、f)　　2. c)　　3. a)、b)、c)、d)
4. a)、c)　　5. b)、d)、e)、f)　　6. d)
7. a)、b)、d)、e)

四、问答题

1. 从河两岸分别过河的两人在河中相遇并要踩上同一垫脚石。

① 互斥：允许人们共享同一垫脚石。

② 占用并等待：在能过河前，人们必须申请使用所有的垫脚石。

③ 抢占：允许一个人在别人过河时走到一旁的歇脚石上（撤离到安全态），或强行要求过河的另一方撤回（完全撤离）。

④ 循环等待：这实质上意味着只允许用单行道方式过河。为了让两个方向的人都能过河，得铺设另一串垫脚石供另一方向专用。

2. (1) **此例中导致死锁的四个条件成立：**

① 互斥。每条道路只能被一辆车占用。

② 占用并等待。每辆车都占用了一段道路，并等待其前方的道路被释放。

③ 非抢占。资源不可抢占。单行线，汽车不能抢路超车。

④ 循环等待。每辆车都等待着前方的汽车把路让出来，且形成了一个环路。

(2) **在每个十字路口设置红绿灯，当南北方向的路通车时，东西方向路上汽车等待。反之亦然。**

3. (1) **由于** Need = Max – Allocation，**所以** Need **的内容是：**

0000

0750

1002

0020

0642

(2) **是处于安全态，序列**<P_0，P_1，P_2，P_3，P_4>**满足安全性要求。**

(3) **能立即得到满足，因为**

① (0420) ≤ Available = (1520)

② (0420) ≤ Max_i = (1750)

③ **分配后的新系统状态如表** 4.5 **所示，且序列**<P_0，P_1，P_2，P_3，P_4>**满足安全性要求。**

表 4.5

Allocation	Max	Need	Available
0012	0012	0000	1100
1420	1750	0330	
1354	2356	1002	
0632	0652	0020	
0014	0656	0642	

第 5 章

内 存 管 理

5.1 重点与难点

5.1.1 概述

用于多道程序操作系统的存储管理算法种类很多，包括从最简单的分区方法到复杂的段页式管理。在一个特定系统中采用什么策略的决定性因素是硬件提供的支持。由 CPU 生成的所有地址都必须进行合法性检查，并尽可能映射到物理地址上，由于效率原因，这种检查用软件不能实现，因此必须用硬件来完成。

各种存储管理算法（如常驻监控程序、MFT、MVT、分页、分段以及段页结合）在很多方面各有差别，在比较不同存储管理算法时主要是从硬件支持、性能、碎片、重定位、交换、共享和保护这七个方面来考虑。

虚拟存储技术允许把大的逻辑地址空间映射到较小的物理内存上，这样就提高了多道程序并发执行的程度，增加了 CPU 的利用率。

请求分页式存储管理是根据实际程序执行的顺序，动态申请存储块的，并不是把所有页面都放入内存中。这种方式允许一个程序，即使它的整个存储映像并没有同时在内存中，也能正确运行，只要缺页率足够低，其性能还是很好的。

请求分页可用来减少分配给一个进程的块数，这就允许更多进程同时执行，而且允许程序所需内存量超出可用内存总量。所以，各个程序是在虚拟存储器中运行的。

当总内存的需求量超出实际内存量时，为释放内存块给新的页面，需要进行页面淘汰。有各种页面淘汰算法可供使用，例如：FIFO 算法、OPT 算法、LRU 算法等。

除了页面替换算法外，还需要块分配策略。分配可以是固定的，如局部页面

替换，也可以是动态的，如全局替换。工作集模型就是一个常用的块分配策略。如果一个进程的内存块数不足以应付工作集的大小，那么它就发生抖动。

段式虚拟存储管理是在分段管理基础上加入虚拟存储技术形成的，一个作业的各个模块可根据调用需要进行动态连接和动态装入。

把段页式存储管理和虚存技术结合起来，就形成了带虚存的段页式系统，它兼顾了分段式存储管理在逻辑上的优点和请求调页式存储管理在存储管理方面的长处，是最通用、最灵活的方式。

内存管理是操作系统中很重要的一部分，要熟悉各种概念，理解掌握各种内存管理方法以及它们的特点。掌握并会应用各种内存置换算法解决相关的问题。

5.1.2 知识要点

1. 概述

(1) 多道程序内存管理

在单道程序系统中，内存一次只调入一个用户进程，在多道程序系统中，多个作业可以同时装入内存，因而对存储管理提出了更多的要求，即要求存储管理具有以下功能：

① 内存空间管理。

② 地址转换。

③ 内存扩充。

④ 内存的共享和保护。

(2) 虚拟存储器

虚拟存储器是系统为了满足用户对存储器容量的巨大需求而采用软、硬件技术虚设的一个非常大的以逻辑方式存在的存储空间，从而用户在编程序时无须担心存储器的容量不足。

一个系统文件结构和系统的虚拟存储区之间存在一定的相似性：页表类似于索引文件系统中的索引块，分段式存储系统类似于连续文件系统。

使用虚拟存储器可以提高存储器的利用率，且能得到比物理地址空间更大的逻辑地址空间，但是这要增加硬件装置，且存取速度较慢。

虚拟存储器最显著的特性是虚拟性，在此基础上它还有离散性、多次性和交换性等基本特征。

(3) 重定位

重定位分为两种：静态重定位和动态重定位。静态重定位的地址转换只在装

入时一次完成；动态重定位的转换动作在作业执行期间随着每条指令的数据访问自动地、连续地进行，因此需要硬件支持。

动态装入是指仅当调用一个子程序，且该子程序又不在内存时，才装入它：即该子程序一直不在内存中，直到第一次调用它后才装入内存。

2. 存储管理的基本技术

基本的存储管理技术有四种，分别是分区式管理技术、可重定位分区式管理技术、覆盖技术和交换技术。

(1) 分区式管理技术

分区法把内存划分成若干个大小不同的区域，除操作系统占用一个区域之外，其余由多道环境下的各并发进程共享。分区法又分为：固定分区法和动态分区法两种。

(2) 可重定位分区式管理技术

为了减少分区法所产生的碎片，定时或在分配内存时把所有的碎片合并成为一个连续区，移动某些已分配区的内容，即紧凑技术。可重定位分区法就是采用紧凑技术进行内存分配的，但是需要硬件及动态重定位技术的支持。

(3) 覆盖技术

覆盖技术是在程序运行过程中，把同一存储区在不同时刻分配给不同的程序段或数据段来共享的一种存储分配技术。覆盖技术的缺点是：程序员必须小心地设计程序及其数据结构，使得要覆盖的段块具有相对独立性，不存在直接联系或相互交叉访问。

(4) 交换技术

交换技术是指将一个进程从内存拷贝到磁盘上，以腾出空间给其他进程使用的技术。需要时，再将该进程调入内存。在交换系统中，交换所占用的时间相当多。减少交换时间主要有以下一些方法：

① 减少一个进程被交换的内存量。

② 提高用于交换的磁盘的速度。

③ 将程序执行和交换重叠进行。

3. 分页存储管理

(1) 分页存储管理的基本概念及其特点

内存的分页存储管理是指，将程序分成一组称为“页”的、固定的、容量大小相同的分区，并允许这些分区是非连续的；将内存分割为一组称为“帧”的固定容量大小的物理内存块，每块的容量大小同一页的容量大小相同。在作业调度时将程序的一部分分配到选定的帧中。

逻辑地址被分成两部分：右边部分给出位移量；左边部分指明页号。要实现页到帧的转换，可根据页号扫描页表，所查找到的表项即帧号，帧号与页位移量合并即得到物理地址。采用分页存储管理技术会出现内部碎片，就平均情况而言，差不多每个作业的最后一页的一半是碎片。

每个作业有自己的页表，页表中包含每个帧的基址及相应的页号。页表一般有三种实现方法：用专用寄存器实现、用主存中的某个区域实现、用联想寄存器实现。

分页具有共享公共代码和减少碎片的优点。分页的另一优点是可以通过共享页面的方式共享公用代码。共享代码必须是可重入代码。可重入代码是在执行期间不允许更改的一种代码，又称为纯代码。这种代码可同时由若干用户使用。

(2) 分页存储管理的分类

分页存储管理（简称分页系统）系统分为纯分页系统和请求式分页系统。

纯分页系统在调度一个作业时，必须把它的所有页一次装入到内存的物理块中，物理块不足时，则该作业必须等待，直到有足够的物理块为止。

请求式分页是指仅当需要一个页面上的信息时，才将该页面信息调入内存。因此，当程序运行时需要一个特定的页，而该页又不在内存中时就会引起缺页中断。这个时候，系统中若没有空余帧可用，则进行页面替换。请求式分页的优点是减少了交换时间和空闲物理存储区的总量，允许系统具有较高的多道程序的度数。

页面替换是选择一个帧（最好不在使用的）作为一个牺牲者，将它交换出去，将所需页交换进来并安置到这个帧中，然后重启该程序。纯页面替换中需要作两次交换：一次用于换出，一次用于换入。可使用设置“修改位”的方法减少开销：即只换出那些修改位被设置了值的页，没有被置值的页不需要换出。

在请求式分页系统中常使用“有效/无效”位（valid/invalid bit），其目的是为了指明某个地址是否是无效的，或某页是否应交换出去。它保存在页/帧表中。

处理一次缺页中断包括如下六个步骤：

① 查看 PCB 中的页/帧表。

② 若地址无效，则夭折该程序；若地址有效，但其所在页不在内存中，则调入它。

③ 查找空闲帧。

④ 请求 I/O 装入所需的页面。

⑤ 更新 PCB 中的页/帧表。

⑥ 重新启动该指令。

在一个请求式分页系统中，可用下面的公式计算有效存取时间：

$$有效存取时间=(1-p)\times t+p\times f$$

其中，p 为缺页中断的概率；t 为内存存取时间；f 为缺页中断时间。

决定缺页中断时间的主要因素是：

① 中断服务时间；

② 交换页面的时间；

③ 重启进程的时间。

解决非自由帧不足的方法：

① 夭折用户程序（这是下策）；

② 将整个程序交换出去，空出它所占用的帧；

③ 替换某些特定的帧。

分页环境下的存储保护是由页保护位完成的。页保护位是与每一页相关的标识位，用以标明“读/写”或“只读”特性。

操作系统利用帧表来记录帧分配的踪迹。

4. 分段存储管理

分段存储管理是将程序分成若干逻辑段，并对这些段分别分配存储空间。这些段的长度可以不同，也不必连续进行分配。分段和分页存储管理系统虽然在很多地方有相似之处，但二者在概念上完全不同，主要体现在：第一，页是信息的物理单位，而段是信息的逻辑单位，并且两者的目的也各不相同；第二，作业地址空间在分页式存储管理中是一维的，而在分段式存储管理中是二维的；第三，页的长度由系统确定，是等长的，而段的长度是不固定的。

相对于分页方法而言，分段方法的主要优点是：

① 可以对重要的段进行保护，而不必保护其他的项。

② 可以方便地与其他用户共享代码和数据。

和分页存储管理一样，分段存储管理要求系统必须有一定的硬件支持，以完成快速请求分段的功能，这包括段表机制、地址转换机构和缺段中断机构的共同支持。

段是按逻辑意义来划分的，可以按名存取，所以，分段存储管理可以方便地实现内存信息共享，并进行有效的内存保护。在分段方法中共享代码时应注意：对每个用户而言，他们共享的代码必须具有相同的段号。

分页方法和分段方法可以组合到一个操作系统中使用，一般有两种实现方法：

① 先分大页，在每页中再采用分段方法；

② 先分大段，在每段中再进行分页。

5. 段页式存储管理

段页式存储管理是分页和分段两种存储管理方式的结合，它同时具备了二者

的优点，既能有效地提高内存的利用率，又能很好地满足用户的需要。

为了实现从逻辑地址到物理地址的转换，系统要为每个进程或作业建立一个段表，并且还要为该作业段表中的每一段建立一个页表。这样，作业段表的内容是页表长度和表地址，为了指出运行作业的段表地址，系统有一个段表地址寄存器，它指出作业的段表长度和段表起始地址。

在段页式存储管理系统中，面向物理存储器的地址空间是用页式划分的，而面向用户的地址空间是用段式划分的，因此它具有双重优越性。

6. 虚拟内存的置换算法

要实现请求式分页，必须解决的主要问题是选择合适的页面替换算法和块分配算法。

在替换页面时，通常选择这样一些牺牲者，即在对它们进行替换时，可达到最低缺页中断率的。可利用访问串（又称引用串）来对替换算法的性能进行评价。访问串是由程序规定的内存地址访问表列。

(1) FIFO 方法

FIFO 方法是将最先进入队列的页号所对应的页面最先选择为牺牲者。这种方法易于理解，但性能不是在任何场合都是好的。

使用 FIFO 方法可能会出现 Belady 异态，这是一种在增加帧的情况下反而使缺页中断率增加的异常情况。

(2) OPT 方法

较理想的页面替换方法是优化（OPT）或最小（MIN）缺页中断方法。这种方法总是替换最长将来时间不被使用的那个页面。它需要访问将来知识。通常用来同其他方法进行比较。

(3) LRU 算法

LRU 算法选择最近最少使用（Least Recently Used）的页作为页面替换的牺牲者。

在实现 LRU 算法时，通常使用下面的方法来确定哪一页是牺牲者：

① 利用硬件计数器和（或）时钟的方法。页表包含页每次使用之后被更改的时间，具有“最老”时间的页即为牺牲者。

② 利用页号栈的方法。每次使用某页时，将对应的页号压入栈顶，已在栈中的页号则依次下移，具有栈底页号的那个页即为牺牲者。

③ 利用一个访问位的方法。每次使用某页时，就将其访问位置值，并在约定的时间间隔后清除该位，牺牲者即为其访问位已被清除的某个页。

④ 利用一个访问字节的方法。每到给定的时间间隔，每个页的访问字节就右移，但在使用某页时，其对应的访问字节的符号位已置值。牺牲者即具有最低

值访问字节的那个页。

LRU 需要额外的硬件，否则，系统将花更多的时间去选择牺牲者而不是去运行程序。

NUR（Not Used Recently）算法是一种较流行的 LRU 近似算法。该算法认为在最近一段时间内未使用过的页在未来也不太可能被使用。在实现中，系统为每个页面增加两个硬件位：引用位（R）和修改位（M）。初始时将所有页面的引用位和修改位清零，当访问某页时，该页的引用位置 1，修改某页时，该页的修改位置 1。选择淘汰页面时先选择未引用的页面，再选择未修改的页面。为避免发生到某时刻大多数页面的引用位都为 1 的情况，系统需要周期性地将所有引用位置 0。

(4) 第二次机会替换（Second-Chance Replacement）方法

这种方法将页面按 FIFO 次序安排，且每个页有一个访问位。先选择“最老”的页，若其访问位被清除，则它就是牺牲者；若它的访问位已置值，则先清除它，然后选择下一页，重复前述过程。

(5) 时钟页面置换算法

这种算法和第二次机会替换算法非常类似，它们的区别仅是实现不同。时钟页面置换算法是把所有的页面保存在一个类似于钟表盘的环形链表中，用一个指针指向最老的页面来实现的。

(6) LFU 方法

LFU（最低使用频率）方法记录对每页访问的次数，并在给定的时间间隔后右移一次该计数值，具有最低计数值的页即为牺牲者。

7. 抖动（thrashing）

抖动是这样一种系统状态，即系统花在页面替换上的时间远远多于执行进程的时间。通常在系统内运行的作业数过多时会出现这种情况。

程序在运行时会有明显的局部性，主要体现在如下两方面：

① 时间局部性，即一旦某条指令或数据被访问，它常常很快又被再次访问。

② 空间局部性，即一旦某个位置被访问到，那么它附近的位置也可能很快被访问到。

Denning 提出了工作集理论来描述和研究这种局部性。所谓工作集，是指在最近时间周期内所访问的一个页面集，又称为工作集窗口（Working Set Window）。工作集模型的优点是允许对每个进程确定比较优化的长度，分配比较合理的帧，以防止抖动现象发生，同时还可以使系统具有较高的多道程序度数。

5.1.3 系统实例

1. Unix System V **和** Linux

Unix **系统和** Linux **系统关于内存管理的详细信息可以参考课本**[1]**中** 5.7 **节的内容。**

2. Windows NT

Windows NT **是由虚拟内存管理器控制内存的分配和执行分页。内存管理器被设计为能在各种不同平台上进行操作的，因而其使用的页面大小可以从** 4KB **到** 64KB **不等。**

每个 Windows NT **用户进程都有一个单独的** 32 **位地址空间，允许使用** 4GB **内存。在缺省情况下，内存由操作系统保留，每个用户有** 2GB **的可用虚拟地址空间，而所有进程共享** 2GB **系统空间。**Windows NT 4.0 **还允许将用户空间增加至** 3GB**，留下** 1GB **作系统空间。**

用户进程的缺省虚拟地址空间分为以下四个区域。

- 0X00000000 **到** 0X00000FFF**：为程序预留，用以捕获空指针。**
- 0X00010000 **到** 0X7FFEFFFF**：可用地址空间，此空间分页调入内存。**
- 0X7FFF0000 **到** 0X7FFFFFFF**：用户不可访问的检验页面，用来供操作系统检查越界指针。**
- 0X80000000 **到** 0XFFFFFFFF **:系统地址空间。此** 2GB **空间供** Windows NT **执行体、微内核和设备驱动程序使用。**

Windows NT **采用了局部范围可变分配的常驻集管理模式。当进程第一次被激活时，系统为其分配一定数目的内存帧作为工作集。进程引用一个不在内存的页面时，系统将该进程的一个驻留页面换出，而将另一新页换入。此外，系统还会根据下列规则来调整活动进程的工作集：**

- **当内存足够时，虚拟内存管理器允许活动进程的常驻集增长，即在产生缺页时调入新页但不换出旧页。**
- **内存不足时，虚拟内存管理器将最近较少使用的页面移出工作集以减小常驻集。**

3. IBM MVS

IBM **的** System 370/XA**（简称** 370/XA**）和** System 370/ESA**（简称** 370/ESA**）**

的体系结构中用了一个两级内存结构，这两级指的是段和页，页的大小为 4KB，段的大小为 1MB。370/XA 用段表和页表来管理存储器，每一个虚拟的地址空间都有一个段表。这样，每一个虚拟的地址空间都包含一些段，用段表中的一项来表示一段。每一个页表项都包含一个有效位（I），表示相应的页是否在物理内存里；包含一个保护位（P），设置时允许对相应的页进行读操作，但不允许写操作。

MVS 充分利用了 370/XA 和 370/ESA 中专为内存管理而设计的硬件的特点。它使用一个全局范围的页面替换策略，系统保留一个可使用帧的表列。当需要装入一个新的页时，就使用这个表列中的一帧。当可使用帧的数目下降到低于一定的阈值时，操作系统就会执行一个页面获取策略，把一些活跃页转化为可使用页。根据处在物理帧中的页的使用历史来决定获取哪一个页。较好的候选页是那些在相对较长时间内没有访问过的页，这类似于执行一个最近最少使用策略。

MVS 每隔一秒钟检查一次物理内存中每一个帧的引用位。如果引用位没有被设置（该帧没被引用），系统就增加这一帧的未访问间隔计数（UIC）。如果引用位置 1，MVS 就把引用位清 0，并把 UIC 置 0。当需要获取一页时，MVS 选择最大的 UIC 进行替换。还可用存储隔离（storage isolation）来修改替换策略，存储隔离要求每一个地址空间都至少保留一个最小数目的页。当获取一页会使某一地址空间的页数低于这个地址空间的最小页数时，就不从这个地址空间中获取页。

5.2 例 题 解 答

例 5-1 叙述支持分页功能且有合理开销的硬件配置。

解 对具有较少页面（少于 256KB）的机器，可用一组专用的高速寄存器来构成页表。当页表较大时，可将页表存于主存并通过页表基址寄存器和一组高速联想寄存器来访问页表。联想寄存器应不断更新以使其包含最近被访问的页号和对应的块号，从而减少对每一内存请求都得去访问页表的情况。

例 5-2 为什么利用分段技术来共享可再入的模块比利用纯粹的分页方法更容易？

解 因为分段技术是基于存储区的逻辑划分而不是基于物理划分而形成的，因此，用户只要通过段表中的一个表项就可共享任意大小的段。在分页方法中，对于每一共享页面，在页表中必须有一公共表项。

例 5-3 设有 8 页的逻辑地址空间，每页有 1024 字，它们被映射到 32 块的物理存储区中。试问：

(1) 逻辑地址应占多少位？

(2) 物理地址应占多少位？

解 (1) **逻辑地址占** $\log_2(1024\times8)=13$ **位；**

(2) **物理地址占** $\log_2(1024\times32)=15$ **位。**

例 5-4 **允许页表中的两个项目同时指向内存中相同页块有什么优点？**

解 **允许页表中的两个项目同时指向内存中相同页块，用户就可共享代码和数据。若代码是可重入式的，那么通过共享使用较大的程序（如正文编辑程序、编译程序、数据库系统等等）就可节省大量存储空间。**

例 5-5 **考虑页面大小为** 100 **字的分页系统。写出下列汇编语言程序（假定地址从** 0 **开始编址）所产生的存取内存地址序列（包含数据指令访问）。**

0. **从** 263 **中取数；**
1. **存入** 264**；**
2. **存入** 265**；**
3. **从** I/O **设备上读信息；**
4. **若** I/O **设备忙，则转至地址** 4**；**
5. **存入** 901**；**
6. **从** 902 **取数；**
7. **停机。**

解 **上述汇编语言程序（从** 0 **开始编址）所产生的存取内存地址序列（包含数据指令访问）如表** 5.1 **所示。**

表 5.1

指令	数据	页号	位移
0	--	0	0
--	263	2	63
1	--	0	1
--	264	2	64
2	--	0	2
--	265	2	65
3	--	0	3
4	--	0	4
…	…	…	…
5	--	0	5
--	901	9	1
6	--	0	6
--	902	0	2
7	--	0	7

例 5-6 对于所给定的段表如表 5.2 所示，下面逻辑地址所对应的物理地址是什么？

(1) 0, 430

(2) 1, 10

(3) 1, 11

(4) 2, 500

(5) 3, 400

(6) 4, 112

表 5.2

段	基址	长度
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

解 (1) 219 + 430 = 649

(2) 2300 + 10 = 2310

(3) 2300 + 11 = 2311

(4) **非法地址访问，自陷入操作系统。**

(5) 1327 + 400 = 1727

(6) **非法地址访问，自陷入操作系统。**

例 5-7 为什么分段方法和分页方法有时要合并成一种方法？

解 为了改进分段方法和分页方法的性能，常常将它们合成为一种方法使用。当页表较大时，采用段页式方案较好。可将页表中一片较大的未用连续区压缩成一个具有页表地址为 0 的段表项目。页段式方法适用于处理段较长的情况。通过将段分页，减少由外部碎片所浪费的存储空间，而且还可以简化分配工作。

例 5-8 为什么 IBM360 上的 OS/MVT 不能紧缩？CDC6600 系统是如何解决这一问题的？

解 在 OS/MVT 系统中，要由用户为其作业加载基址寄存器，由于这不是特权指令，所以用户可能在程序中会将该寄存器用于另外目的，比如说跳转到子程序。若此时正在交换该作业，然后系统紧缩，修改该作业的基址寄存器，把它安排到新的地址空间。当作业再次交换进入系统执行，就恢复基址寄存器原来之值并继续执行，那么它可能会访问其当前程序所在存储位置之外的存储地址。所以在 OS/MVT 系统中不能紧缩。

为了能紧缩，CDC6600 利用“基址/界限寄存器”实现动态重定位，并由操作系统控制这些寄存器。这样可在内存中重新安排作业来实现紧缩，而且在作业交换完成之后不装入这些寄存器原来之值。

例 5-9 在一个动态链接分段系统中，进程之间可以共享段但不必有相同的段号。

(1) **定义一个系统，它允许静态链接及段的共享而不需要相同的段号。**

(2) 提出一种分段方案，它允许共享页面而不需有相同的页号。

解　这两个问题可归结为这样一种程序，它可以访问它自己的代码及其数据的地址而不知道与这些地址相关的段号或页号。该问题可通过把每一进程与 4 个寄存器相连来获得解决（见 Multics 系统）。一个寄存器保留当前程序段的地址，一个保留栈的基址，另一个保留全局数据的基址，等等。其基本思想就是所有的访问都得通过一个寄存器间接进行。该寄存器将相关的访问映像到当前的段号或页号中。通过改变这些寄存器的内容，不同的进程可以执行相同的代码而不必有相同的页号或段号。

例 5-10　什么时候发生缺页中断？当缺页中断发生时，操作系统应采取什么行动？

解　当访问一个还未调入（或不在）内存区的页面时便发生缺页中断。当发生这种中断时，操作系统应检验该访问是否合法。若不合法，则中止该程序；若合法，则查寻出一空闲块并请求 I/O 把所需的页面读入该空闲块，一旦 I/O 完成就更新进程表和页表并重新执行这一指令。

例 5-11　对于下面的页面替换算法：

(1) LRU；

(2) FIFO；

(3) Optimal；

(4) Second Chance。

按照缺页中断从劣到优的顺序排列它们，并区分出它们是否有 Belady 异态。

解　各算法按缺页中断率排列如表 5.3 所示。

表 5.3

次序	算法	有无 Belady 异态
1	Optimal	无
2	LRU	无
3	Second Chance	有
4	FIFO	有

例 5-12　大多数机器都没有分页或分段的硬件机制。如果提交的作业需要非常大的地址空间。可用软件实现分页或分段功能吗？如果可以，请简述如何实现。就实现而言哪种管理方案更容易些？

解　为了模拟硬件分页或分段功能，须将每一机器地址映像到某个适当的数据结构（页表或段页表）中以提供有效地址。这一点可通过解释而不是编译所有代码的方式实现。若用户申请较大的地址空间，那么采用分页方案较好，因为较

大的段可能需要分页。由于页面的大小是一致的，所以在任何情况下，实现分页比较容易。

例 5-13 某计算机系统提供 2^{24} 字的虚拟存储空间，该计算机有 2^{18} 字的物理存储区，虚拟存储器是通过分页方法实现的，且页面的大小为 256 个字。假定一用户程序产生了虚拟地址 11123456（八进制）。说明该系统是如何产生对应的物理地址的？

解 这个虚拟地址的二进制形式为：

001 001 001 010 011 100 101 110

由于页面大小为 2^8，页表容量为 2^{16}，因此，上述二进制形式中的低八位“ 00 101 110 ”用作页面的位移量，而其余的十六位“ 001 001 001 010 011 1 ”用作页表的位移量。根据页表起始地址和页表的位移量找到页表项即帧号，然后将帧号与页位移量合并就得到了物理地址。

例 5-14 假定你的替换策略（在分页系统中）是：周期性检查每一页面，若某一页面自上次检查起至此未被使用，则淘汰它。与 LRU 或 Second Chance 相比，你选用这种策略的优、缺点是什么?

解 LRU、Second Chance 以及大多数页面替换算法都是在仅当需要新页时才引用，而使用这种周期性检查页面策略时，系统在内存中不保留不再使用的页面，即使对这些存储块没有需求也是如此。这样，在大多数情况下，都有一些可供使用的空闲块且不需要采取页面替换。当然，在所有的页都同时使用的情况下，则可能需要更多的附加内存。

例 5-15 假定占有 m 块（初始为空）的进程有一个页访问串，这个页访问串的长度为 p，其中涉及到 n 个不同的页号。对于任何页面替换算法，求出：

(1) 缺页中断次数的下界是多少？

(2) 缺页中断次数的上界是多少？

解 (1) 缺页中断次数的下界是 n；

(2) 缺页中断次数的上界是 p。

例 5-16 假定我们希望使用需要一个访问位的分页算法（类似 Second Chance 或工作集）。但硬件并未提供这种访问位。你能否模拟这种访问位？若能，简述如何模拟。

解 可以利用硬件支持的“ 有效/无效 ”位来模拟这种访问位。最初，置该位为“ 无效 ”。在第一次访问时设置一个陷阱而进入操作系统。操作系统将置一个软件位为 1，并重置该“ 有效/无效 ”位为“ 有效 ”。

例 5-17 类似于分页式方法中的请求式分页那样，分段式方法中也可以采用请求式分段策略。这就需要一个段替换算法（类似于页面替换算法）。试提出一个合理的段替换算法，并估计在段替换过程中会出现哪些在页面替换过程中不出

现的问题？

解　可使用 FIFO 替换算法。它查找满足要求的第一个段，为避免内部产生存储碎片，可把该段的未被占用的部分并入空闲空间表中。若找不到满足要求的段，则可通过查找两个或多个连续的段来满足要求。在段替换过程中必须要考虑到段的大小变化，但在页面替换中页的大小是固定不变的。

例 5-18　考虑下面的页访问串：

1，2，3，4，2，1，5，6，2，1，2，3，7，6，3，2，1，2，3，6

假定分别有 1，2，3，4，5，6，7 个页块，应用下面的页面替换算法时，各会出现多少次缺页中断？

(1) LRU；

(2) FIFO；

(3) Optimal。

解　应用上面的算法时，各会出现的缺页中断次数如表 5.4 所示。

注意：所给定的页块初始均为空，因此，首次访问一页时就会发生缺页中断。

表 5.4

页块号	LRU	FIFO	Optimal
1	20	20	20
2	17	18	15
3	15	16	11
4	10	14	8
5	8	12	7
6	7	9	7
7	7	7	7

例 5-19　分段与分页很类似，只是段的大小是可变的。试基于 FIFO 和 LRU 页面替换算法提出两个段替换算法。

解　FIFO 查找第一个这样的段，该段能容纳即将进入的段，若不允许重定位，且没有一个段能容纳即将进入的段，则选择几个内存区域连续的段，它们“最靠近空闲空间表的首部”，合并它们后能成为所需要的段。若允许重定位，则重新安排内存，使得能够容纳即将进入段的开始的 N 个段在内存中是连续的。在这两种情况下，都应将剩余空间并入空闲空间表中。

LRU 选择一个未被使用的、时间周期最长的、能容纳即将进入段的段，并将剩余空间加到空闲空间表中。若没有满足要求的段，则选择一组内存连续的且合并起来能满足要求的“最老”的段（不能重定位的情况）。若可以重定位，则重新安排这组最老的段，使其在内存中是连续的，它们合并后可以满足即将进入段

的要求。

注意：由于段的大小可以不同，因此，所选定被替换的段可能小于要替换它的那个段，需要分别考虑对能够重定位和不能重定位的情况。

例 5-20 一个好的页面替换算法应使缺页中断次数最少。一种方法是将正使用的页均匀地分散在整个存储区中。可以给每一页附加一个计数器，用它记录与该页相关的页的个数。当进行页面替换时，选择其计数器之值最小的那个页。

(1) 利用上述思想，提出一个页面替换算法，并回答下面的问题：

① 该计数器的初值是多少？

② 该计数器何时增值？

③ 该计数器何时减值？

④ 如何选择被替换的页？

(2) 若有 4 个页，给定下面的页访问串，则使用你的算法将会出现多少次缺页中断？

1，2，3，4，5，3，4，1，6，7，8，9，7，8，9，5，4，5，4，2

(3) 给定(2)问中同样的条件和访问串，若采用最佳页面替换算法，其缺页中断次数的最小值是多少？

解 (1) 回答问题如下：

① 该计数器的初值为 0。

② 每当一个新页与该计数器对应的页块相关时，计数器增值。

③ 每当与该计数器对应页块相关的那些页之一不再使用时，计数器减值。

④ 查找一个其计数器之值最小的页块，选择被替换的页。

(2) 14 次缺页中断。

(3) 11 次缺页中断。

例 5-21 引起系统抖动的原因是什么？系统如何检测抖动？一旦检测出抖动后，系统怎样消除它？

解 由于分配给进程的页面数少于进程所需的最低页面数，因此，会出现接连不断的缺页中断，从而引起系统的抖动。系统可以利用将 CPU 的利用率与多道程序的度数进行比较的方法来检测系统抖动，一旦发生抖动，则可通过减少多道程序的度数的办法来消除它。

例 5-22 假定有一个使用“基址/界限寄存器”的操作系统，为了也能提供页表，曾对机器作了修改。那么，能否用页来模拟“基址/界限寄存器”？若能，则说明如何模拟；若不能，则说明为什么。

解 只要内存按固定大小的段分配，就可用页表模拟“基址/界限寄存器”。

在这种方法中，页表可记录段的基址，而其“有效/无效”位用于指明该段在内存中的部分。此外，还要注意解决可能出现的内部碎片问题。

例 5-23　现有一请求式分页系统，页表保存在寄存器中。若有一个可用的空页或被替换的页未被修改，则它处理一个缺页中断需要 8ms。若被替换页已被修改，则处理一个缺页中断需要 20ms。内存存取时间为 1μs。

假定 70%被替换的页被修改过，为保证有效存取时间不超过 2μs，可接受的最大缺页中断率是多少？

解　设最大缺页中断率为 p，则

$$(1-p)\times 1\mu s + 0.3p \times 8ms + 0.7p \times 20ms = 2\mu s$$

$$-p + 2400p + 14000p = 1$$

$$p \approx 0.00006$$

例 5-24　考虑一个存储器。现从它的一端起连续存放若干个段 S_1，S_2，…如果已经将 n 个段存放于存储器内，则下一个要存放的段 S_{n+1} 必须存放于 S_n 之后（即使 S_1，…，S_n 中某些段已经被删除）。如果 S_{n+1} 存放不下，就需要对正在使用的段进行压缩合并（这种存储分配方法称为“压缩-顺序存放”的存储分配。如 S_1，S_2，S_3，S_4，S_5，S_6，S_7 七个段中，若 S_1，S_4，S_6 被删除，则需分配内存给新段，而内存不足时，就要对段区进行压缩，即将正在使用的段合并为连续的存储区，压缩后的段为 S_2，S_3，S_5，S_7。

设 t 为一个段的平均寿命，以存储访问次数为单位；s 为一个段的平均长度，以字为单位；f 为平衡状态下，不用部分占总存储量的百分比；F 表示用于压缩的系统开销占存储分配总开销的百分比。试证明：

$$F \geq (1-f)/(1+Kf) \qquad \text{其中，} K=\frac{t}{2s}-1$$

解　考虑一个有 c 个字的存储器。上次压缩后，空域部分 H 有 f·c 个字。边界以每次存储访问有 s/t 个字的速度移动。结果，在 f·c·t/s 次访问后，边界越过空余部分而到达存储器的另一端。此时$(1-f)c$ 个字必须被压缩。这就需要至少 $2(1-f)c$ 次存储访问。所以用于压缩的时间开销满足如下关系：

$$F \geq 2(1-f)\ c/(2(1-f)c+f\cdot c\cdot t/s)$$

上式化简后即为所要证明的关系式。

例 5-25　在页式管理中，设主存大小为三页，已知页面走向为：4，3，2，1，4，3，5，4，3，2，1，5。调页时分别采用先进先出（FIFO）和最佳（OPT）算法，问缺页率各是多少？

解　先进先出算法的页面调度情况如表 5.5 所示。其中，缺页次数为 9 次；缺页中断率 f= 9/12 = 75%。

表 5.5

次序	1	2	3	4	5	6	7	8	9	10	11	12
页面分配情况	4	3	2	1	4	3	5	5	5	2	1	1
		4	3	2	1	4	3	3	3	5	2	2
			4	3	2	1	4	4	4	3	5	5
是否缺页	是	是	是	是	是	是	是	否	否	是	是	否

最优算法的页面调度情况如表 5.6 所示。其中，缺页次数为 7 次；缺页中断率 f=7/12=58.3%。

表 5.6

次序	1	2	3	4	5	6	7	8	9	10	11	12
页面分配情况	4	3	2	1	1	1	5	5	5	5	5	5
		4	3	3	3	3	3	3	3	3	1	1
			4	4	4	4	4	4	4	2	2	2
是否缺页	是	是	是	是	否	否	是	否	否	是	是	否

例 5-26 考虑一个请求式分页的计算机系统，它使用一个分页盘，利用全局 LRU 替换算法和一种把页平均分给进程的分配策略（即若有 m 个页和 n 个进程，则每一进程分得 m/n 个页）。多道程序设计的度数固定为 4，测得系统的 CPU 和分页盘的利用率为：

(1) CPU 的利用率为 13%，盘利用率为 97%；

(2) CPU 的利用率为 87%，盘利用率为 3%；

(3) CPU 的利用率为 13%，盘利用率为 3%。

上述每一种情形可能会出现什么问题？能否用增加多道程序的度数来增加 CPU 的利用率？

解 (1) 会出现抖动现象，不能靠增加多道程序的度数来增加 CPU 的利用率。

(2) CPU 的利用率相当高，可以增加多道程序度数。

(3) 增加多道程序的度数。

例 5-27 试说明多级存储方法与虚拟方法的主要区别。

解 所谓多级存储方法是指在主存以外再配上后援存储器来补充主存的方法，它先把用户作业放在后援存储器上，然后等到要真正执行它时再设法把这部分作业调入主存，它们之间的调度由用户自己负责，这造成用户使用上的困难。由于系统不同，其主存大小也会随之不同，用户往往不得不修改程序。在批量分时系统中多个用户同时共享主存，因此，用户就不能直接参与主存和后援存储器

之间的调度了。这就需要由操作系统来对它进行统一的、自动的管理，并采用虚拟存储技术，有了虚拟存储器之后，用户就不感到主存和后援存储器之间的区别，而在统一的逻辑存储器上安排程序和数据。由于虚拟存储器比实际存储器大得多，因而可以同时装入许多作业。

例 5-28　设一作业共有 5 页（第 0~4 页），其中程序占 3 页（第 0~2 页）、常数占 1 页（第 3 页）、工作单元占 1 页（第 4 页）。它们依次放在外存的 45、46 页和 98、99、100 页上。现已将程序段分配在内存的 7、10、19 页中；而常数区和工作区尚未获得内存。请回答下述问题：

(1) 页表应包括哪些项目？填写此页表。若工作区分配到内存的第 9 页，则页表如何变化？

(2) 在运行中，因需使用常数而发生中断，假定此时内存无空闲页面，需要把第 9 页淘汰，操作系统应如何处理？页表又发生什么变化？

解　(1) 页表所包括的项目应有：作业页号、在内存标志、存取方式、所在外存页号、所在内存页号、修改标志等（见表 5.7）。若工作区分配到内存的第 9 页，则在页表中作业页号为 4 的那一行，“ 在内存标志 ”一栏改为 1，“ 内存页号 ”一栏填入 9。

在表 5.7 中，存取方式 E 表示可执行、R 表示可读、W 表示可写。

表 5.7

作业页号	在内存标志	存取方式	外存页号	内存页号	修改标志
0	1	E	45	7	
1	1	E	46	10	
2	1	E	98	19	
3	0	R	99		
4	0	RW	100		

(2) 在把第 9 页淘汰之前，先检查其修改标志，若此页内存已发生过写操作，则说明与外存对应的页面副本内存不一致，必须重新送往外存，然后才能分配给常数区。页表中作业页号为 4 的那一行，“ 在内存标志 ”一栏改为 0；作业页号为 3 的那一行，“ 在内存标志 ” 一栏改为 1，“ 内存页号 ” 一栏填入 9。

例 5-29　某虚拟存储器的用户编程空间共 32 个页面，每页为 1KB，内存为 16KB，假定某时刻一用户页表中已调入内存页面的页号和物理块号的对照表如右表所示：

页　号	物理块号
0	7
1	10
2	4
3	5

则逻辑地址 OA5C(H)所对应的物理地址是多少？

解　逻辑地址 OA5C(H)分成页号和页内偏移，页号是 2，对应的物理页是 4 号，再把物理页号 4 和页内偏移组合成物理地址，即得 125C（H)。

例 5-30　考虑下述页面走向：

4，3，2，1，4，3，5，4，3，2，1，5

当内存块数量分别为 3 和 4 时，试问 LRU、FIFO、OPT 这 3 种置换算法的缺页次数各是多少(假设所有内存块起初为空)？

解　其置换次数如表 5.8 所示。

表 5.8

	LRU	FIFO	OPT
3 块	10	9	7
4 块	8	10	6

例 5-31　为什么与分页技术相比分段技术更容易实现程序和数据的共享和保护？

解　分页技术有利于内存的利用率，它能更好地面向内存分配的物理实现，而分段技术则有利于满足用户的需求，段是一组逻辑信息的集合，它的组织结构能很好地与用户作业的逻辑结构相吻合。含有很多的描述信息及字段可以用于保护和共享。

5.3　自 测 练 习

一、填空题

1. 内存分配算法主要有_____、_____和_____算法。
2. 页表中包含每个帧的_____和_____。
3. 决定缺页中断时间的主要因素有_____、_____和_____。
4. 常用的解决外部碎片问题的方法是_____。
5. _____页面调度，简称为_____，是最常用的虚拟存储器系统。
6. 在某些页面替换算法中，缺页率可能随着可使用的块数量的增加而增长，这种情况称为_____。
7. 分页环境下的存储保护是由与每页相连的_____来完成的。

二、判断题（正确的在括号中记√，错误的记×）

1. 为了减少内部碎片，页应偏小为好。 ( )
2. 为了减少缺页中断率，页应该小一些。 ( )
3. 用户程序中出错处理部分不必常驻内存。 ( )
4. 使用预分页的原因是每个进程在最初运行时需要一定数量的页面。( )
5. Cache 是作为主存和后援存储器之间缓冲区的一种速度较高的存储区。 ( )
6. 页保护位是与每一页相关的标识位，用以表明“读/写”和“只读”特性。 ( )
7. 可变分区法可以比较有效地消除外部碎片，但不能消除内部碎片。 ( )

三、选择题

1. 下面哪些程序设计技术和数据结构“适合于”请式调页环境？ ( )
   a) 栈；　　b) 杂凑符号表；
   c) 顺序查找；　　d) 折半查找；
   e) 纯代码；　　f) 向量操作。
2. 假定有一个请式调页系统，现测得相关成分的利用率为：
   CPU 的利用率 20%；
   分页磁盘 99.7%；
   其他 I/O 设备 5%。
   下面哪些措施将（有可能）改进 CPU 的利用率？ ( )
   a) 增加一个更快速的 CPU；　　b) 增添一个更大的分页盘；
   c) 增加多道程序的度数；　　d) 减少多道程序的度数；
   e) 增加其他更快速的 I/O 设备。
3. 下面对计算机存储器体系中的各个部分按速度从快到慢排列，其中正确的是哪些？ ( )
   a) 寄存器、Cache、主存储器、后援存储器、磁盘设备、磁带设备；
   b) Cache、寄存器、后援存储器、主存储器、磁盘设备、磁带设备；
   c) 主存储器、Cache、寄存器、后援存储器、磁盘设备、磁带设备；
   d) 磁盘设备、主存储器、寄存器、Cache、后援存储器、磁带设备。
4. 下列存储器哪些可用来存储页表？ ( )
   a) Cache；　　b) 主存；
   c) 后援存储器；　　d) 高速磁盘。
5. 下面哪种内存管理方法有利于程序的动态链接？ ( )

a) 分段存储管理；　　b) 分页存储管理；

c) 可变区分割分配；　　d) 固定区分割分配。

四、问答题

1. 假定有一个用联想寄存器记录最活跃页表项目的分页系统。还假定页表通常放在内存，且内存存取时间为 1μs。

(1) 若所有存储访问的 85%都可以在联想寄存器中找到相应的项目，那么，有效存取时间是多少？

(2) 若联想寄存器中的命中率只有 50%，那么有效存取时间又是多少？

2. IBM/360 系统是利用键（key）来实现存储保护的。一个键有四位，每 2048 个字节的内存块有一个与其相关的键（存储键）。CPU 也有一个与其相关的键（保护键）。仅当这两个键相等或其中之一为 0 时才允许存储操作。试问，下面哪些存储管理方案可成功地与这种硬件设置一起使用？

(1) 裸机。

(2) 常驻监控程序。

(3) 具有固定进程数的多道程序系统。

(4) 具有可变进程数的多道程序系统。

(5) 分页。

(6) 分段。

3. 有一个 460 字的程序的下述内存访问序列：

10，11，104，170，73，309，185，245，246，434，458，364

(1) 假定页面大小为 100 字，试给出页访问串。

(2) 假定内存中有 200 个字可供程序使用，且采用 FIFO 算法，那么有关该访问串的缺页中断率是多少？

(3) 若使用 LRU 替换算法，那么有关该访问串的缺页中断率是多少？

(4) 若使用 Optimal 替换算法，其缺页中断率又是多少？

4. 设有二维数组为

var A:array[1..100] of array[1..100] of integer；

其中，数组元素 A[1][1]存放在页面大小为 200 的分页存储系统中的地址 200 处。使用该数组的一个较小的程序存放在第 0 页中（地址为 0～199），这样只会从第 0 页中取指令。

假定现有三个页面，第一个页面存放程序，其余两个页面初始为空，使用 LRU 替换算法，下面的数组初始化循环时，将会产生多少次缺页中断？

(1) for j := 1 to 100 do

　　for i := 1 to 100 do

```
            A[i][j] := 0;
(2) for i := 1 to 100 do
        for j := 1 to 100 do
            A[i][j] := 0;
```

5. **列出处理一次缺页中断的步骤。**

6. **图** 5.1 **所示的是一种段页式管理配置方案。**

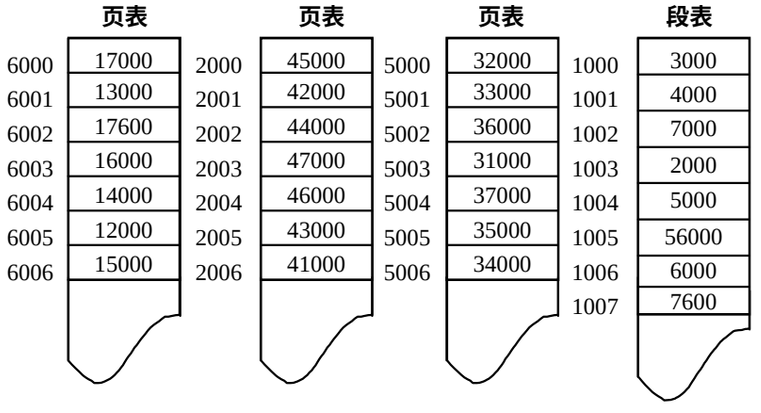


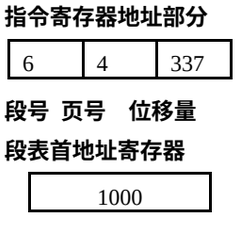
图 5.1

(1) **根据给出的虚地址写出实地址。**

(2) **描述地址映像过程。**

(3) **从虚地址到实地址要经过几次访问主存？哪几次？**

(4) **一般采用什么设施提高映像速度？**



5.4 自测练习答案

一、填空题

1. **内存分配算法主要有首次适配、最佳适配和最差适配算法。**

2. 页表中包含每个帧的基址和页号。

3. 决定缺页中断时间的主要因素有中断服务时间、交换页面的时间和重启进程的时间。

4. 常用的解决外部碎片问题的方法是压缩。

5. 请求式页面调度，简称为请式调页，是最常用的虚拟存储器系统。

6. 在某些页面替换算法中，缺页率可能随着可使用的块数量的增加而增长。这种情况称为 Belady 异态。

7. 分页环境下的存储保护是由与每页相连的保护位来完成的。

二、判断题

1. √　　2. ×　　3. √　　4. √

5. ×　　6. √　　7. ×

三、选择题

1. a)、c)、e)、f)

2. d)

在存储分配过程中，该系统显然在分页方面花费大量时间，若减少多道程序设计的度数，则驻留进程的缺页中断率减小，且改进了 CPU 的利用率，改进性能的另一方法是获得更多的物理存储器或更快速的分页盘。

3. a)

4. a)、b)

5. a)

四、问答题

1. (1) 有效存取时间 = 85%×1μs+15%×2μs = 1.15μs

   (2) 有效存取时间 = 50%×1μs+50%×2μs = 1.5μs

2. (1) 不必保护，置系统键为 0；

   (2) 当处于监控态时置系统键为 0；

   (3) 大小必须固定为 2KB 的倍数，根据存储块来分配键；

   (4) 同(3)；

   (5) 块的大小必须是 2KB 的倍数，根据页面来分配键；

   (6) 段的大小必须是 2KB 的倍数，根据段来分配键。

3. (1) 此时，页访问串为 0，0，1，1，0，3，1，2，2，4，4，3。

   (2) 6 次缺页中断。

   (3) 7 次缺页中断。

(4) 5 次缺页中断。

4. 假定数组按行存储，即依次先存放第一行的元素，接着依次存放第二行的元素等等。

(1) 页访问串是

0，1，2，…，49，0，1，2，…，49，…

因此，将发生 5000 次缺页中断；

(2) 页访问串是 0，1，2，…，49

因此，将只发生 50 次缺页中断。

5. (1) 查看 PCB 中的页/帧表（page-frame）。若地址无效，则夭折该程序。

(2) 若地址有效，但其所在页不在内存中，则调入它。

(3) 查找空闲帧。

(4) 请求 I/O，装入所需的页面。

(5) 更新 PCB 中的页/帧表。

(6) 重新启动该指令。

6. (1) 主存实地址为 14337。

(2) 地址映像过程为：段号 6 映像到段表为 1006，在段表 1006 项查得页表地址为 6000。进而查得页表项 6004 内容为 14000，加上位移量 337 即得实地址为 14337。

(3) 共三次访问主存：第一次访问段表，第二次访问页表，第三次访问实地址。

(4) 因访问一个实地址需三次访问主存，故速度较慢。若段表、页表采用快表则可提高速度，快表是用联想存储器构成的（由高速寄存器组成）。因联想存储器的容量不能太大，故快表的大小有限。因此只存放常用的页面，故会产生调页问题。

第 6 章

处理机调度

6.1 重点与难点

6.1.1 概述

处理机调度分为三个级别：高级(作业)调度、中级调度、低级(进程)调度。一进程在 CPU 上的一次连续执行过程称为该进程的一个 CPU 周期。一个 CPU 周期是由进程自我终止的。一个进程通常有若干个长短不等的 CPU 周期。

进程调度方式包括抢占方式和非抢占方式两种。在抢占方式下，系统可强行夺走现行进程占用的 CPU，即强行分割现行进程的当前 CPU 周期。在非抢占方式下，系统不能分割当前的 CPU 周期，只在一个 CPU 周期执行完后才能重新调度。

基本的调度原则是：尽量提高系统吞吐量，均衡利用资源，对所有作业给予公平服务，对高优先级作业或进程给予优先服务。

如何选择和设计调度算法是实现调度的关键。对调度算法进行评价的常用量度标准是：平均周转时间、平均带权周转时间及平均等待时间。

通过本章的学习，了解处理机调度类型，记熟相关概念；清晰地理解各种调度算法机制，以及它们的优缺点，熟练地运用常用调度算法解决问题；记熟多处理机调度和实时调度的相关概念，理解这两种调度的独特之处以及调度算法的相似和不同。处理机调度也是操作系统中很重要的一部分。

6.1.2 知识要点

1. 调度类别

(1) 高级调度(长程调度)

高级调度即作业调度，也叫长程调度。它决定哪些作业可参与竞争 CPU 和

其他系统资源，从状态观点出发，就是将一个或一批作业从后备状态变为运行状态。一个作业一旦被高级调度选中，便可获得所需要的基本内存和设备资源，并被装入内存，此后就以进程形式参与并发运行，与其他进程竞争 CPU。换言之，高级调度决定给哪个作业分配一台虚拟处理机，获得虚拟处理机的作业将在该虚拟处理机上顺序执行。从这个意义上说，高级调度进行的是虚拟处理机的分配，即 CPU 的宏观调度。需要指出的是，不同类型的操作系统关于作业调度的功能、调度时机以及工作形式是有差异的。

当一个作业完成后，或当系统中作业数目还未达到规定的多道程序的度数时，引用高级调度。

(2) 中级调度(中程调度)

中级调度决定哪些进程可参与竞争 CPU，从状态观点出发，就是将进程从活动态变为静止的挂起态，或者将进程从挂起态变为就绪态或等待态。这主要是为了短期调整系统负荷，以缓和内存紧张的矛盾。中级调度的实质是执行“挂起”和“激活”操作。挂起一个进程是把该进程的实体(程序和数据)从内存迁移到外存的专门区域（称为交换区），并释放该进程占用的用户内存区，这称为“换出”；反之，激活一个进程是把该进程的实体从外存交换区迁移到内存，这称为“换入”。因此，中级调度也常称为进程交换，通常仅用于分时系统。

(3) 低级调度(短程调度)

低级调度即进程调度。它决定哪个进程可获得物理 CPU，从状态观点出发，就是将某个进程从就绪态变为执行态。被低级调度选中的进程将实际获得 CPU，并可立即在物理 CPU 上执行它的程序。因此，低级调度是处理机三级调度中的终结调度。

2. 调度算法

低级调度的主要目标是以使其性能得到优化的方法来分配处理机时间。通常使用的标准分为两类：面向用户的和面向系统的。也可根据是否与性能相关来区分。面向用户的标准有 4 个，与性能相关的是：响应时间、轮转时间和期限；与性能无关的是：预测性。面向系统的标准有 5 个，与性能相关的是：吞吐量和处理机利用率；与性能无关的是：公平性、优先权和资源平衡。详细信息可以参考课本[1]中的表 6.2。

衡量和比较系统性能优劣的基本因素有：

① CPU 利用率。

② 吞吐量，即每单位时间所完成的作业数目。

③ 轮转时间，即从作业提交直至完成这一作业的时间间隔。

④ 等待时间，即作业在就绪队列中等待所花的时间。

⑤ 响应时间，即从作业提交到首次产生回答信息之间的时间。

主要的调度算法有如下几种。

(1) FCFS 调度算法

FCFS(First Come First Served)即先来先服务，故它本质上是非抢占式的。它简单易行，但调度性能较差，有可能使短的、重要的或紧迫的作业及进程长期等待。其实现过程容易，可采用 FIFO 队列管理。

(2) SJF 调度算法

SJF(Shortest Job First)即最短作业优先，该算法对短作业或短进程最为有利，它可获得最短的平均周转时间。但它忽略等待时间的长短，对长作业不利，特别是在抢占方式下，可能会使长作业无限延迟。对于抢占式 SJF 进程调度，还需要考虑是按最短原则还是按剩余最短原则抢占。理论上该方法在等待时间方面是最优的，但实际上无法预测下一个 CPU 瞬时段的长度。

(3) SRTF 调度算法

SRTF(Shortest Remaining Time First)调度算法是这样一种抢占性调度算法，即它将较高的优先级给予具有最短剩余 CPU 瞬时段的作业。

(4) HRRN 调度算法

HRRN(Highest Response Ratio Next)调度算法，即最高响应比优先调度算法。该算法的好处是考虑了进程的等待时间，既优待了短作业，又不忽略先来者。它是非抢占式的调度算法。

(5) RR 调度算法

RR(Round Robin) 调度算法，即轮转调度算法，是指对每个作业给予一个运行时间片，若一个作业在规定的时间片内未运行完，则挂起该作业并调度另一作业(继续)运行。当所有的作业都运行完分配的一个时间片后，第一个作业才再次得到运行的机会。RR 算法性能依赖于时间片的大小，时间片过大则退化为 FCFS 算法，时间片过小时则称为“处理机共享”。该算法是抢占性算法。

(6) HPF 调度算法

HPF（Highest Privilege First）调度算法，即最高优先级算法，是让具有高优先级的作业或进程获得优先服务。优先级通常用一个整型的优先数表示。优先级的设置可采用静态的或动态的两种方式。动态设置优先级可使调度更为灵活，使调度性能得到改善。HPF 调度可以是抢占式的或非抢占式的。

(7) Feedback 调度算法

Feedback(多级反馈队列)调度算法是指进程依赖于其条件的变化，从一个队列移到另一队列。它综合了 FCFS、RR 及 HPF 三种算法，根据进程运行情况的反馈信息而对进程就绪队列进行组织并实施调度。Feedback 调度是一种抢占式调度算法。

这些算法的特性都在课本[1]中的表 6.3 中一一列出。

3. 多处理机调度

多处理机调度中有三个相关联的要点：

① 分配进程给处理机；

② 在单个处理机上运行多道程序；

③ 实际分派进程。

使用了线程这个概念之后，执行的概念就与进程分开了。一个应用程序可用一系列线程的形式完成，它们之间相互配合，并在相同的地址空间同时运行。

在多处理机上调度线程和分配处理机时，较多使用以下四种方法。

(1) 负载共享

线程并不分配给某一特定的处理机。系统中有一个全局就绪队列，每个处理机空闲时就从该队列中选择一个线程。这里的负载共享要与负载平衡相区别。

(2) 群调度

相关联的线程集被一个处理机集一对一调用。

(3) 专用处理机分配

与自我调度相对应，其调度是隐式的，在程序执行期间，每个程序都被分配给与其线程数相等的处理机，程序结束时，将所有的处理机归还，以便其他程序使用。

(4) 动态调度

程序的线程数可随程序的执行情况而改变。

4. 实时调度

实时操作系统在决定性、响应性、用户控制、可靠性、弱失效操作等五个方面有独特的要求。为满足这些要求，实时系统应包含如下特性：

① 快速现场切换；

② 尺寸小；

③ 迅速响应外部中断；

④ 多任务并存，并有如信号量、信号、事件等进程间通信工具；

⑤ 使用专门的线性文件来收集数据；

⑥ 基于优先权的抢占调度；

⑦ 最小化禁止中断的时间间隔；

⑧ 简单地延迟任务一段时间或停止／重新开始任务；

⑨ 特殊的警告和超时。

常见的实时调度算法可分为四类：

① 静态表驱动算法。
② 静态优先权驱动抢占算法。
③ 动态计划算法。
④ 动态尽力算法。

5. 其他概念

(1) 多道程序度数

多道程序的度数(degree of multiprogramming)是指系统中当前已有的正在运行／等待的混合作业群中的作业个数。

(2) 进程交换

进程交换是指进程的调入／调出，即把一进程从内存复制到磁盘上，以腾出空间供另一个(些)活跃进程使用；当内存充裕时，或轮到运行该复制的进程时，再把它调入内存中(可能不在原来的空间)。

(3) 算法评估

对调度算法的评估方法主要有分析评估法、模拟方法和实现法等几种。分析评估法占主导地位，它通过算法和系统负荷产生相应的公式和数据来评估该算法相对于负荷的性能，通常采用确定性模型和队列模型来进行评估。

(4) Gantt 图

Gantt 图是描述进程／作业执行情况的一种直观形式，它展示了作业／进程执行过程中瞬时段或时间片的变化分布情况。

(5) 确定模拟

确定模拟是指草拟给定的一组作业的 Gantt 图，并确定指定的平均数。确定模拟的优点是计算简单，缺点是其结论(结果)仅适用给定的一组作业。

6.1.3 系统实例

有关 Unix System V、Windows NT 和 IBM MVS 这三个操作系统在处理机调度方面的信息可以参考课本[1]中 6.5 节的内容。在此主要介绍一下 Linux 操作系统的一些知识。

Linux 使用一个基于简单优先权的调度算法来从现有进程中选择合适的进程运行。在进程的 task_struct 数据结构中包含以下有关调度的信息。

• policy：将被应用于本进程的调度策略。一般有时间片轮转(RR)和先进先出(FCFS)两种策略。

• priority：给予进程的优先级。

• rt_priority：实时进程的相对优先级。

• counter：随运行时间递减的时间计数器。

Linux 系统支持对称多处理器(SMP)。在 SMP 系统中，每个进程的 task_struct 包含进程正在运行的处理器号和其上次运行的处理器号。一进程被选中后可在不同的 CPU 上运行，但可以用 processor_mask 来把一个进程限制在指定的处理器上运行。

6.2 例 题 解 答

例 6-1 叙述抢占调度策略和非抢占调度策略之间的区别，解释为什么在分时系统中不可能使用严格的非抢占调度策略。

解 抢占调度策略允许中断一个正在执行的进程，并抢占它所占有的 CPU，把 CPU 分配给另一进程。非抢占调度策略确保一进程仅当它完成了自己当前所占用的 CPU 时间片后才释放 CPU。

非抢占调度策略不适用于分时系统，因为它不能保证每个用户在固定的时间间隔内共享 CPU。非抢占调度策略允许程序无限期地运行下去，这样就延误了其他已提交作业的轮转时间(响应时间)。

例 6-2 简述短程调度策略、中程调度策略和长程调度策略间的差别。

解 长程调度又称为作业调度，它决定哪些作业进入内存以便处理；短程调度也称为 CPU 调度，它从位于内存的就绪作业中选择一个作业，并将 CPU 分配给它运行。中程调度特别适用于分时系统中作为中间级的调度程序使用。它利用交换方案从内存移出部分运行的程序，并在以后某个时候又将其调入内存且从其断点恢复运行。

三者的主要差别在于它们的执行频率不同。短程调度必须频繁地为 CPU 选择新的进程。长程调度执行将作业调入内存的任务，而且在允许另一作业进入内存之前，它可能不得不等待某个作业的完成，因此，其执行的频繁程度比短程调度要低得多。

例 6-3 叙述多级调度的作用。

解 多级调度通常根据作业的某些特性将就绪队列划分为若干队列，每一队列有其自己的调度算法，根据一定规则将这些队列设计成具有不同的优先级。高优先级的进程会得到一定的优惠(较快地获得较多的 CPU 时间)。

多级反馈队列根据作业具有不同 CPU 瞬时段的特性对作业进行分类，并允许作业在队列之间移动。在这种环境中，开始时所有作业位于同一队列，若在给定的时间间隔之后某作业还没完成，则将它置入另一队列，它在该队列中将等待

另一 CPU 时间。继续该过程直至该作业完成或它进入基于 FCFS 的队列为止。

例 6-4　一个交互系统用时间片轮转调度时，试图给那些小进程足够的响应时间。当结束所有就绪进程一轮后，系统用最大响应时间除以进程数作为下一轮的时间片，这可行吗？

解　这种调度算法不可行，因为在时间片轮转调度中，无法知道一个进程的响应时间，因而无法知道最大响应时间。

例 6-5　哪种进程——偏重处理机型或偏重 I/O 型，适合于多级反馈队列调度？简述之。

解　偏重 I/O 的进程最适合多级反馈队列调度，因为偏重 I/O 的进程的总体服务时间中，需要 CPU 的时间相对要少许多，它到达不了深层的队列就已经执行完毕了。而偏重处理机的进程则不然，由于需要处理机的时间很长，所以要到达很深层的队列后才能逐步执行完毕。

例 6-6　考虑采用 RR 调度策略的环境，其中就绪队列中的项目指向相应进程的进程控制块。问：

(1) 在就绪队列中，若将两个指针指向同一进程会产生什么结果？

(2) 这种方案的优缺点是什么？

(3) 如何修改基本的 RR 调度算法，使其不使用重复指针也能实现相同的效果？

解　(1) 事实上，由两个指针所指向的进程将提高它的优先级，因为它会得到更多的 CPU 时间从而受到优先处理。

(2) 这种方案的优点是越重要的作业可得到越多的 CPU 时间，即这类作业会优先处理。但这显然对短作业不利。

(3) 为值得优先处理的进程分配较多的 CPU 时间片，换言之，在这种 RR 调度算法中将有两个或多个时间片。

例 6-7　在基于优先级的调度中，仅当没有优先级更高的进程时，才有一个进程获得调度。假设没有其他调度信息，且优先级在创建进程时产生，以后不变。在此系统中，使用 Dekker 的互斥方法很危险，为什么？考虑会有什么不希望的事情发生，如何发生？

解　例如，当一个低优先级的进程正在临界段时，处理机被调度给另一个与之互斥执行的但优先级比前者高的进程，此时就会发生死锁。低优先级的进程无法获得处理机，因而无法执行完临界段释放互斥资源，高优先级的进程也因无法进入临界段而忙碌-等待。这种情况产生的原因是，Dekker 算法是用忙碌-等待策略解决互斥，等待进程仍在执行代码，而不是被阻塞，因此高优先级进程无法让出处理机供低优先级进程使用。用信号量解决互斥不会产生此问题。

例 6-8　多级队列系统中，对不同级别的作业分配不同时间片的优点是什

么？

解　在多级队列系统中，若短作业所需的时间比初始时间片还少，则它们将有最高的优先级。这样，CPU 就能很快地运行完短作业再去集中处理长作业。对于被推迟到下一级的作业，可给它们分配比其初始时间片更长的时间片，因为我们总希望尽可能快地运行尽可能多的程序，同时保持对其他程序的延迟时间最小。因此，根据多级队列系统中的级别来增加时间片，可以使短作业获得较高的优先级，长作业也可同时运行且具有最少的延迟时间。

例 6-9　许多 CPU 调度算法都有参数。例如，RR 算法需要指明时间片的参数；多级反馈队列需要一些参数来定义队列的个数、每个队列的调度算法以及在各队列间移动作业所用的准则等等。这意味着这些算法实际上是一些算法的集合(例如，有关所有时间片的 RR 算法，等等)；一个算法的集合可以包含另一个算法的集合(例如，FCFS 是具有无穷时间片的 RR 算法)。请指出下列每对算法间存在什么关系(如果存在某种关系的话)：

(1) 最高优先级，SJF；

(2) MLFQ(多级反馈队列)，FCFS；

(3) 最高优先级，FCFS；

(4) RR，SJF。

解　(1) 最短作业具有最高的优先级。

(2) MLFQ 的最低级是 FCFS。

(3) FCFS 有一种优先级，这种优先级定义为：将最高优先级给予到达时间最长的作业。

(4) 没有关系。

例 6-10　假定要在一台处理机上执行下表所列作业：

作业	执行时间	优先级
1	10	3
2	1	1
3	2	3
4	1	4
5	5	2

且假定这些作业到达的次序是 1，2，3，4，5。

(1) 给出 Gantt 图来说明分别使用 FCFS，RR(时间片=1)，SJF 以及非抢占优先调度算法时这些作业的执行情况；

(2) 针对上述每一调度算法，给出每个作业相应的轮转时间；

(3) 就上述每一调度算法，求出每个作业相应的等待时间；

(4) 对所有作业而言，具有最小平均等待时间的调度算法是哪一个？

解　(1) 执行情况的 Gantt 图如图 6.1 所示。

FCFS

1(10)	2(1)	3(2)	4(1)	5(5)

RR

1	2	3	4	5	1	3	5	1	5	1	5	1	5	1

SJF

2	4	3	5	1

Priority

2	5	1	3	4

图 6.1

(2) 与这些作业相对应的轮转时间如表 6.1 所示。

表 6.1

作业	FCFS	RR	SJF	Priority
1	10	19	19	16
2	11	2	1	1
3	13	7	4	18
4	14	4	2	19
5	19	14	9	6

(3) 与这些作业相应的等待时间如表 6.2 所示。

表 6.2

作业	FCFS	RR	SJF	Priority
1	0	9	9	6
2	10	1	0	0
3	11	6	2	16
4	13	3	1	18
5	14	9	4	2

(4) SJF 调度算法具有最小平均等待时间。

例 6-11　证明：在非抢占式调度算法中，SPN（最短进程优先）具有最小平均等待时间。

解　假设在某一时刻，有 n 个作业到达，其服务时间是：$t_1 \le t_2 \le \cdots \le t_n$，所以响应时间 $T(t_1)=t_1$，$T(t_2)=t_1+t_2$，… 因此平均响应时间为：

$$T_{av}=\frac{1}{n}\sum_{i=1}^{n}(n-i+1)\,t_i$$

再假设作业 a 和作业 b 交换执行顺序，且 $a<b$，则新响应时间为：

$$T'_{av}=\frac{1}{n}\left[nt_1+(n-1)t_2+\Lambda+(n-a+1)t_b+\Lambda+(n-b+1)t_a+\Lambda+t_n\right]$$

$$T'_{av}-T_{av}=\frac{1}{n}\left[(n-a+1)t_b-(n-a+1)t_a+(n-b+1)t_a-(n-b+1)t_b\right]$$

$$=\frac{1}{n}(t_b-t_a)(b-a)$$

因为 $a<b$，所以 $t_a<t_b$，上式大于 0，即：按 SPN 法调度平均响应时间最小。

例 6-12　假定有一个调度算法(属于短程调度级别)比较偏爱那些在“最近的过去”很少使用过处理机的程序，试叙述为什么该算法有利于偏重 I/O 的程序但又不会永久性地“饿死”偏重 CPU 的程序？

解　该算法有利于偏重 I/O 的程序，是因为它们只需要相当短的 CPU 时间。此外，由于偏重 I/O 的程序会相当频繁地释放 CPU 而去执行它们的 I/O，因此，偏重 CPU 的程序也不会饿死。

例 6-13　在一个非抢占式单处理机系统中，就绪队列在时间 t_0 时有 3 个进程，它们分别在 t_1，t_2，t_3 时刻到达，估计运行时间为 t_{r_1}，t_{r_2}，t_{r_3}。图 6.2 表明了它们的响应比随时间线性变化的情况。利用这个例子找出一个响应比调度的方法，叫做最大响应比最小化调度算法，它使一组给定的进程的最大响应比最小，不考虑后续到达。

提示：首先决定哪一个进程最后调度。

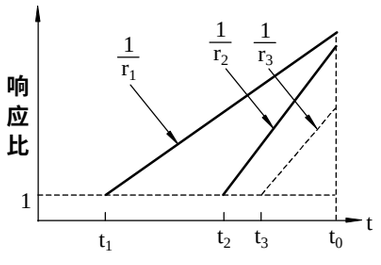


图 6.2

解　考虑 t_1 时刻到达的进程，假设它最后一个被调度，那么计算出它被调度时的响应比 r_1，以同样的方式计算出另两个进程的响应比 r_2，r_3，比较这三个值，

最小值对应的进程应该被最后调度。同样的方法再决定剩下两个进程中，哪一个最后被调度，如此下去即可得到一个调度序列。该调度序列即可使这一组进程的最大响应比最小。

例 6-14　叙述下列调度算法在支持短作业方面的差别：

(1) FCFS；

(2) RR；

(3) Feedback。

解　(1) FCFS——不怎么有利于短作业，因为在长作业之后到达的短作业将会等待较长的时间。

(2) RR——同等对待所有的作业(给它们分配相等的 CPU 时间)，由于短作业将首先完成，所以短作业可以较快地离开系统。

(3) Feedback——比较有利于短作业，因为其工作方式类似于 RR 算法。

例 6-15　什么叫无限期封锁(indefinite blocking)? 它在什么情况下出现？

解　无限期封锁也称为饥饿，即一个具有优先级的进程决无运行的机会，当 CPU 接连不断地被较高优先级的作业占用时就可能出现这种情况。

例 6-16　在一个排队系统中，新作业在服务前必须等待。等待时，它的优先级从 0 开始随时间以α斜角线性增加，作业一直等待到优先级等于正在被服务作业的优先级，然后，加入被服务作业共享处理机，同时优先级以β斜角线性增加。该算法称为自私时间片轮转算法，因为被服务进程优先级不断增加来垄断处理机，试根据图 6.3 所示，证明：作业服务时间为 t_x，平均响应时间为 t_{Rx}，且

$$t_{Rx}=\frac{t_s}{1-\rho}+\frac{t_x-t_s}{1-\rho'}$$

$$\rho=\lambda t \quad \rho'=\rho\left(1-\frac{\beta}{a}\right) \quad 0\le\beta\le\alpha$$

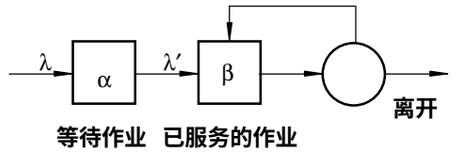
假设到达和服务时间分别服从$\frac{1}{\lambda}$和 t_s 的指数分布。

提示：考虑整个系统及两个子系统。

解　根据图 6.3 所示的几何关系，作业以$1/\lambda$ 的平均时间间隔到达α，从α出来后，则是以$1/\lambda'$的时间间隔到达β，并且$\lambda'=\lambda\left(1-\frac{a}{\beta}\right)$，根据排队论公式$\rho=\lambda t_s$有 $\rho'=\rho\left(1-\frac{\beta}{\alpha}\right)$。一个作业的平均响应时间由两部分组成：在$\alpha$中的等待时间，在$\beta$中的服务时间。其中：

α中的等待时间＝整个系统的平均响应时间－β的平均响应时间时间

$$=\frac{t_s}{1-\rho}-\frac{t_s}{1-\rho'}$$



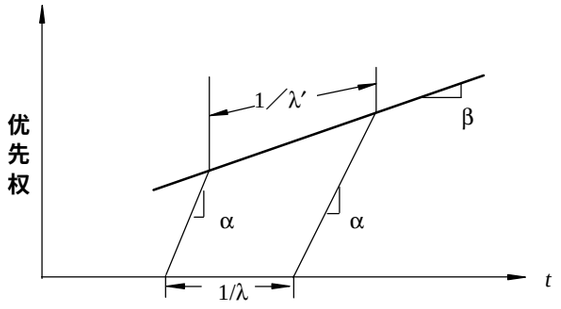


图 6.3

其中，$$\rho' = \rho\left(1-\frac{\beta}{\alpha}\right)$$

服务时间为 t_x 的作业的响应时间为（由上一题的结论）：$\dfrac{t_x}{1-\rho'}$，

所以，平均响应时间 $t_{Rx} = \dfrac{t_s}{1-\rho} - \dfrac{t_s}{1-\rho'} + \dfrac{t_x}{1-\rho'} = \dfrac{t_s}{1-\rho} + \dfrac{t_x - t_s}{1-\rho'}$

例 6-17　一个处理机被所有的就绪队列中的进程所复用，进程以无限的速度运行(这是一个理想的时间片轮转调度，它使用的时间片和要求服务的平均时间相比非常小)。证明：对于一个无穷的泊松分布输入，一个进程的平均响应时间 t_{Rx} 和服务时间 t_x，有如下关系：

$$t_{Rx} = \frac{t_x}{1-\rho}$$

提示：考虑系统在给定输入下的平均队列长度。

解　在理想的时间片轮转调度的情况下，一个服务时间为 t_x 的进程在创建后立即被加入就绪队列，并且系统中共有 n 个进程，则此进程在系统中的时间为 nt_x，因为进程只分得 $1/n$ 的处理器时间。在此服务模型中，根据排队论的分析理论，进程到达系统时，系统中平均有 $\rho/1-\rho$ 个进程，加上此进程，共有 $1/1-\rho$ 个进程分享处理器，所以此进程的平均响应时间为

$$t_{Rx} = \left(\frac{1}{1-\rho}\right)t_x = \frac{t_x}{1-\rho}$$

6.3　自 测 练 习

一、填空题

1. 在 CPU 调度层之上必须作出的调度决策有：_____和_____。
2. 处理机调度分为三类：_____、_____和_____，本章主要讨论的是_____。
3. 最简单的调度算法是_____，最优的调度算法是_____。
4. 当 RR 算法的时间片过小 (如小于 1ms) 时，它又被叫做_____。
5. 对调度算法的评估方法通常有_____、_____和_____。
6. 多处理机调度线程主要有_____、_____、_____和_____四种方法。
7. 实时操作系统在_____、_____、_____、_____和_____五个方面有独特的要求。

二、判断题（正确的在括号中记√，错误的记×）

1. 吞吐量是每单位时间所完成的作业个数。 (　　)
2. 轮转时间＝CPU 忙碌时间＋等待时间。 (　　)
3. 时间片稍大于进程切换的时间。 (　　)
4. 一个系统中通常有一到两个设备队列。 (　　)
5. 实时系统通常采用抢占式调度。 (　　)
6. 抢占是指引起一个进程暂时停止，以便运行另一个进程。 (　　)

三、选择题

1. 下面哪些是合理的长程调度算法？ (　　)
   a) RR(轮转法)；　　b) SJF(最短作业优先)；
   c) LJF(最长作业优先)；　　d) LCFS(后来先服务)。
2. 下面哪些是合理的短程调度算法？ (　　)
   a) FCFS(先来先服务)；　　b) RR(轮转法)；
   c) HPF(最高优先级优先)；　　d) LJF(最长作业优先)。
3. 下面哪些是衡量和比较系统性能优劣的指标？ (　　)
   a) CPU 利用率；　　b) 吞吐量；
   c) 轮转时间；　　d) 等待时间。
4. 影响优先级的内部因素有哪些： (　　)

a) 查找策略；　　b) 时间限制；
c) 内存使用情况；　　d) 打开的文件个数。

5. RR 算法是专门为下面哪个系统设计的算法？　　(　　)
a) 批处理系统；　　b) 分时系统；
c) 实时系统；　　d) 多处理机系统。

6. Unix 系统采用的是哪种调度算法？　　(　　)
a) RR；　　b) FCFS；
c) Feedback；　　d) HRRN。

7. 下面哪些方法是实时系统使用的调度方法？　　(　　)
a) 动态尽力方法；　　b) 群调度方法；
c) 静态表驱动方法；　　d) 动态计划方法。

四、问答题

1. 假定待处理的三个作业的到达时间和运行时间如下表所示：

作业	到达时间/s	执行时间/s
1	0.0	8
2	0.4	4
3	1.0	1

若采用下列调度算法，则这些作业的平均轮转时间是多少？

(1) FCFS。

(2) SJF。

(3) 假定要改善 SJF 的性能，由于事先并不知道其中的两个较短的作业将很快到达，所以在时刻 0，选择作业 1 来运行。如果让 CPU 在第一个时间单位空闲，然后使用 SJF 调度策略，那么，这些作业的平均轮转时间是多少？

注意：由于作业 1 和作业 2 在空闲时间中处于等待状态，因此，它们的等待时间可能会增加，该算法称之为“未来知识调度算法(FKS)”。

2. Kleinrock 提出了一个基于动态修改优先级的抢占性优先级调度算法，当作业等待 CPU(在就绪队列，但并未运行)时，其优先级以速率 a 变化；当作业运行时，其优先级以速率 b 变化。当进程进入就绪队列时，其优先级为 0。给参数 a 和 b 赋以不同之值可能得到不同的调度算法。

(1) b>a>0 导出的调度算法是什么？

(2) a<b<0 导出的调度算法是什么?

3. CPU 调度算法决定了其被调度的作业的执行次序。假定现有一台处理机和 n 个被调度的作业，那么，存在多少种不同的调度方案呢？试根据 n 给出相应的

公式。

4. 叙述多级反馈调度算法和多级多队列(前／后台)调度算法之间的区别。这里，对于前台作业使用 RR 算法，对于后台作业使用抢占性优先算法。

5. 简单描述 Unix 系统的调度策略。

6.4　自测练习答案

一、填空题

1. 在 CPU 调度层之上必须作出的调度决策有：作业调度和交换。

2. 处理机调度分为三类：高级调度(或长程调度)、中级调度(或中程调度)和低级调度(或短程调度)，本章主要讨论的是低级调度(短程调度)。

3. 最简单的调度算法是 FCFS 算法，最优的调度算法是 SJF 算法。

4. 当 RR 算法的时间片过小（如小于 1ms）时，它又被叫做处理机共享。

5. 对调度算法的评估方法通常有分析评估法、模拟法和实现法。

6. 多处理机调度线程主要有负载共享、群调度、专用处理机分配和动态调度四种方法。

7. 实时操作系统在决定性、响应性、用户控制、可靠性和弱失效操作五个方面有独特的要求。

二、判断题

1. √　　2. √　　3. ×　　4. ×　　5. √　　6. √

三、选择题

1. b)、c)　　2. b)、c)　　3. a)、b)、c)、d)　　4. b)、c)、d)
5. b)　　6. c)　　7. a)、c)、d)

四、问答题

1. (1) FCFS　10.53s
   (2) SJF　9.53s
   (3) FKS　6.86s

记住：轮转时间是完成时间与到达时间之差，所以，计算轮转时间应减去到达时间。若忘记减去到达时间，则采用 FCFS 调度策略时，这些作业的平均轮转时间为 11s。

2. (1) FCFS

(2) LIFO

3. n！

4. **多级反馈队列调度算法和多级多队列调度算法的主要区别是，在多级反馈队列中，若作业未完成，则它们将从一队列移到另一队列。用于多级多队列前台作业的** RR **算法类似于多级反馈队列中的高层队列，只是前台的每一作业分配相等的时间片，而且当其执行完给定的时间片后也不会丧失其优先级。**

5. **参见系统实例。**

第 7 章

I/O 设备管理

7.1 重点与难点

7.1.1 概述

计算机设备是计算机系统的重要组成部分。由于设备种类繁多，所以管理很复杂。根据设备的物理特性和为了管理上的方便、有效，常采用的设备管理技术有三种：独占、共享和虚拟。

设备管理程序应具有如下功能：对缓冲区进行管理，进行 I/O 调度，把虚拟设备地址转换成物理地址并驱动 I/O 设备执行 I/O 操作，管理设备的中断等。其基本目标是：向用户提供方便的设备使用接口以及充分发挥设备的利用率。

大、中型计算机系统常采用通道来实现 CPU 计算和 I/O 操作的并行。小型和微型计算机系统则通常采用总线结构进行设备的数据交换。

缓冲技术是得到广泛采用的用以平滑数据 I/O 速率的技术。利用缓冲技术可增强系统的处理能力和提高资源的利用率。按照组织方式的不同，缓冲可分为单缓冲、多缓冲和缓冲池三类。对缓冲区的使用是通过对缓冲控制块的申请、释放等操作来实现的，系统设立不同的队列来管理缓冲控制块。

磁盘系统是大多数计算机中最主要的 I/O 设备。可移动头磁盘的调度算法总是试图减少磁头移动的总量。可用的调度算法有 FCFS、SSTF、SCAN 和 C-SCAN 等。扇区排队是适用于固定头磁盘和磁鼓的调度算法。

通过本章的学习要对整个 I/O 系统有一个整体的了解，包括软件系统和硬件系统；通晓各个知识点和概念，应用的重点在磁盘调度，要会运用这些算法。

7.1.2 知识要点

1. I/O 系统硬件

计算机所管理的 I/O 外部设备可以分为三类：用户可读设备；机器可读设备和通信设备这些设备之间差异很大，表现在一些不同方面：传输速度不同，应用方式和范围不同，控制的复杂性不同，传输单元不同，数据描述不同，错误条件不同，报错的方法、后果和影响也不同。

(1) 设备控制器

I/O 设备一般由机械和电子两部分组成，将它分为两部分是为了便于模块化和通用的设计。电子部分称为设备控制器或适配器(Device Controller 或 Adapter)。设备控制器是一个可编址器件，很多控制器可以连接多个相同的设备，有与控制设备数一样多的设备地址，使每一个地址对应一台设备。每个控制器有 n 个寄存器通过寻址机制与 CPU 通信，用于 I/O 功能的地址集合称为 I/O 空间。

(2) I/O 技术

1) I/O 技术概述

操作系统中所采用的 I/O 技术一般有三种 :程序控制 I/O——由一段 I/O 控制程序进行 I/O，它不断查询 I/O 控制器，直到 I/O 动作完成为止；中断驱动 I/O——在这种方式下，进程只需发出 I/O 请求，如果进程不需等待 I/O 完成，则继续执行后续指令序列，否则，该进程被挂起，直到中断到来才解除挂起状态，然后继续其他工作 ;直接存储器存储(DMA) I/O——由一个 DMA 模块控制主存和 I/O 设备块之间的数据交换，传输数据块过程无需 CPU 干预。DMA 出现以后，I/O 部件有了自己的处理器和存储器，于是产生了 I/O 模块的概念，由 I/O 模块独立控制 I/O 任务的完成。

2) DMA 技术

在 I/O 技术中，DMA 是一个比较重要且应用广泛的技术，它优于中断技术，其特点是传输单位是数据块，所传数据直接送到内存，仅在传输开始和结束的时候需要 CPU 做少量的干预工作。DMA 可以有很多配置方案，各个方案有不同的特性和优缺点，适用于不同的系统要求。

3) 通道技术

通道也称为 I/O 处理机，是在 DMA 的基础上发展起来的，它也是一种 DMA 技术。只是通道的控制芯片更为复杂，有了自己专用于 I/O 的指令集和存储器。这时的 I/O 模块（即通道）本身也就构成了一个小型的专用计算机。通道的功能

比 CPU 弱、速度较慢，但价格便宜。通道由 CPU 启动后独立于 CPU 工作，并在 I/O 任务完成后向 CPU 发出中断信号报告指定任务的完成情况。

通道结构中 I/O 控制的工作过程主要有下面三步：

① CPU 需进行 I/O 数据交换时，先组织通道程序并将程序起始地址放入通道地址字（CAW）中，然后执行启动 I/O 指令。

② 通道启动后，根据 CAW 访问通道程序，执行通道指令，向控制器发出 I/O 操作命令完成实际操作。

③ 控制器执行实际 I/O 操作，控制设备进行数据传输。

一般 I/O 通道有如下三种类型：

① 字节多路通道。

② 数组选择通道。

③ 数组多路通道。

通道技术一般仅用于大型计算机系统之中。

2. I/O 软件

(1) I/O 软件的层次及其设计

I/O 软件的总体目标是，按分层的思想构造软件，较低层的软件要使较高层的软件独立于硬件，较高层的软件则要向用户提供一个友好、规范、清晰的界面。I/O 软件设计的具体目标是：设备独立性、统一命名、同步/异步传输、出错处理、设备共享与独占。

根据 I/O 软件的设计目标，将 I/O 软件组织成以下 4 个层次：中断处理程序、设备驱动程序、与设备无关的 I/O 软件和用户空间的 I/O 软件。

1) 中断处理程序

在采用中断驱动方式管理 I/O 设备时，当设备完成任务后，会向 CPU 发出中断信号，CPU 分析中断原因，并调用对应的中断处理程序进行处理。中断处理程序进行相应的检查并取走数据，然后从中断处理程序返回至原来的执行点，继续执行。

2) 设备驱动程序

设备驱动程序中存放着所有与设备相关的代码，每一类设备配置一种驱动程序。设备驱动程序的功能有如下几点：

① 将接收到的来自它上一层的与设备无关的抽象请求转为具体请求。

② 检查用户 I/O 请求的合法性，了解 I/O 设备的状态，传递有关参数、设置设备的工作方式。

③ 发出 I/O 命令，启动分配到的 I/O 设备，完成指定的 I/O 操作。

④ 及时响应控制器或通道发来的中断请求，并调用相应的中断处理程序进

行处理。

⑤ 对于有通道的计算机系统，驱动程序还应能根据用户的 I/O 请求构成通道程序。

3) 与设备无关的 I/O 软件

与设备无关的 I/O 软件和设备驱动程序之间的确切界限依赖于具体系统，某些系统出于效率的考虑，让设备驱动程序来实现本层软件功能。与设备无关的软件的基本任务是实现一般设备都需要的 I/O 功能，并且向用户层软件提供一个统一的逻辑接口。与设备无关的 I/O 软件系统称为 I/O 子系统。I/O 子系统执行着与设备无关的操作，同时还为用户应用程序提供一个统一的接口。

4) 用户空间的 I/O 软件

用户空间的 I/O 软件有两类，一类是与用户程序相连的库过程，这些库过程是 I/O 软件的一部分。另一类是 SPOOLing（外部设备联机并行操作）系统，也称假脱机。它是针对慢速独占设备提出的一种设备管理技术，其核心思想是利用一台可共享的、高速大容量的块设备来模拟独占设备的操作，使一台独占设备变为多台可并行的虚拟设备，即把独占设备变成逻辑上的共享设备。

SPOOLing 系统具有下列特点：

① 对于用户进程是透明的，用户进程仍使用统一的系统调用命令访问字符设备。

② 用户进程实际上使用的是虚拟设备，而不是直接使用字符设备。

③ 字符设备与各虚拟设备之间的数据交换由 SPOOLing 进程统一调度实施，而且这种交换是以并行方式进行的。

(2) 缓冲

为缓和 CPU 与 I/O 设备间速度不匹配的矛盾，减少 CPU 的中断频率，放宽对中断响应的限制，提高 CPU 和 I/O 设备之间的并行性，操作系统引入了缓冲技术。I/O 缓冲区是缓冲技术用到的重要数据结构，缓冲区管理是逻辑 I/O 系统的基本功能之一。

从使用方式上分，缓冲区可分为专用缓冲区和公用缓冲区两部分。前者是为某台设备专门设置的缓冲区，占用固定的内存空间。后者是为所有设备设置的缓冲区，为各设备共享。

缓冲区的组织方式可分为：单缓冲区、多缓冲区和缓冲池。单缓冲区和多缓冲区的组织方式为专用缓冲区方式采用。对于单缓冲区，生产者和消费者只能串行访问缓冲区。对于多缓冲区（如双缓冲区），生产者和消费者可并行交替地访问各缓冲区，从而加快了传输速度，提高了系统效率。缓冲池为公用缓冲区方式采用。它由若干大小相等的缓冲区组成，各缓冲区既可用于输入，也可用于输出。

3. 磁盘调度

(1) **调度算法简介**

完成一个磁盘服务请求的总时间由查找时间、等待时间和传输时间组成。查找时间是（在读/写时）磁头查找所需柱面的时间。等待时间是磁盘旋转到所需扇区开始处所花的时间。传输时间是磁盘和主存之间的实际传输时间。

磁盘服务请求队列非空时，需采用适当的调度算法确定服务次序，以缩短总的服务时间。磁盘调度算法有很多，课本[1]**中的表** 7.1 **列出了这些算法。**

常用的算法有 FCFS（**先来先服务**）、SSTF（**最短查找时间优先**）、SCAN、C-SCAN **等算法。**

① FCFS **算法是最简单的调度算法，其优点是容易编程且较清晰，缺点是性能较差，不能提供最好的（平均）服务。**

② SSTF **算法先处理位于磁道上最接近于读/写头当前位置上的** I/O **请求。其实质是一种** SJF **调度形式，其不足之处是如果绝大多数请求都聚集在少数磁道上，而少数请求位于远离这些磁道的其他磁道上，就可能造成某些请求“饥饿”。**

③ SCAN **算法先从具有** I/O **请求的最低磁道号上开始处理** I/O **请求，然后按磁道号的次序处理** I/O **请求，直至到达最高磁道号，再按相反的次序，重复这一过程。**

④ C-SCAN **算法在到达最高磁道号并处理完其上的** I/O **请求（如果有的话）后，就马上返回到最低磁道号去处理** I/O **请求，且不处理返回途中所遇到的任何请求。**

实际上，实现 SCAN **或** C-SCAN **算法时，通常是让磁头在每个方向上仅仅移动到最后请求位置，若当前方向上没有请求，则磁头反向。这种实现方式称为** LOOK **或** C-LOOK**。**

(2) **各算法特点比较**

在各算法中，SSTF **算法比较通用，**SCAN **和** C-SCAN **算法更适合大负荷的磁盘系统。当请求队列中只有一个请求时，采用** FCFS **算法较合适。**

对于固定头设备（如磁鼓），需采用不同的调度算法。常见的算法是扇区排队算法。其基本思想是，把磁道划分成固定数量的称为扇区的块，为磁鼓的每个扇区定义一个单独队列，当一个对扇区 I **的请求到达后，就被置入扇区** I **的队列中。扇区** I **移动到读/写头下时，就为相应队列中的第一个请求服务。如果在一个特定磁道或柱面上存在多个服务请求，则可将扇区排队算法用于可移动头设备。**

(3) **磁盘缓冲区**

为了提高磁盘访问的平均速度，在主存中为磁盘扇区开辟的一个缓冲区，称为磁盘高速缓存。在使用磁盘高速缓存时要考虑两个问题：首先，Cache **中的数**

据传送，是传送地址指针还是传送数据值要依情况不同而定，前者效率较高；其次是 Cache 中数据的替换策略，经常使用的算法是最近最少使用算法(Least Rencently Used, LRU)和最少使用算法(Least Frequently Used, LFU)。

7.1.3 系统实例

有关 Unix System V 和 Windows NT 这两个操作系统在设备管理方面的信息可以参考课本[1]中 7.4 节的内容。在此主要介绍一下 IBM MVS 操作系统有关设备管理的一些知识。

MVS 采用了分层的 I/O 设备管理方法。MVS 的 I/O 逻辑结构从下往上可以分为通道子系统、I/O 检查者（IOS)、EXCP 处理器、访问方法和用户程序等五层。在 MVS 中一个典型的 I/O 执行序列包括以下步骤：

① 用户程序用宏指令对目标 I/O 设备进行 I/O 操作。操作一般都以 OPEN 宏指令开始。

② 一条 I/O 宏指令调用一个称为存取方法的操作系统服务。MVS 存取方法分成三类：传统的存取方法、远程通信存取方法和静态存取方法。这些存取方法可使程序和 I/O 实现细节分开。

③ 为了请求数据的移动，存取方法或用户程序把操作信息传给 EXCP（执行通道程序）处理器，EXCP 把信息转换成通道子系统可理解的形式，并调用 I/O 检查者。

④ IOS 把队列中的 I/O 定位到选好的 I/O 设备，并初始化通道子系统。

⑤ 通道子系统选择主存和设备中最好的数据传输的通道路径，并控制数据的移动。I/O 完成时，子系统发出一个中断给 CPU。

⑥ IOS 评估中断，返还控制权给 EXCP。

⑦ EXCP 更新信息，表明 I/O 操作的结果，把控制交给分配者。

⑧ 分配者重新激活存取方法来响应 I/O 请求的完成。

⑨ 存取机制将控制信息和任何需要的状态信息返回给用户程序。

7.2 例 题 解 答

例 7-1　试画出微型机的主机中常采用的 I/O 系统组织结构图。

解　微型机的主机中常采用的 I/O 系统组织结构如图 7.1 所示。

例 7-2　假定有一个具有 200 个磁道（编号为 0~199）的移动头磁盘，在完

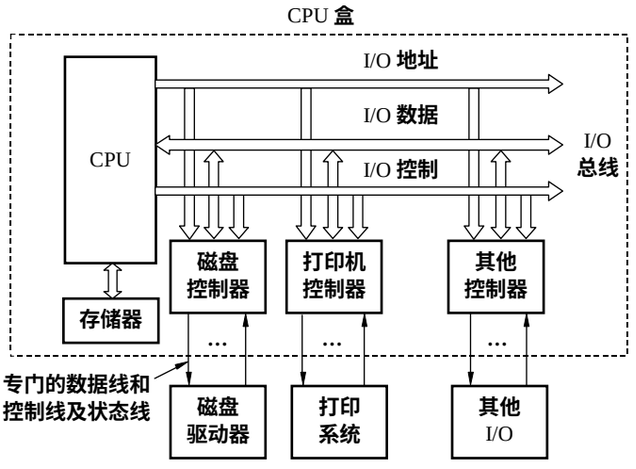


图 7.1

成了磁道 125 的请求后，当前正在磁道 143 处为一个请求服务。若请求队列以 FIFO 次序存放：

86，147，91，177，94，150，102，175，130

对下列每一个磁盘调度算法，若要满足这些请求，则总的磁头移动次数是多少？

(1) FCFS　　(2) SSTF　　(3) SCAN

(4) LOOK　　(5) C-SCAN

解　各种调度算法的寻道次序如表 7.1 所示。

表 7.1

次　序	FCFS	SSTF	SCAN	LOOK	C-SCAN
1	143	143	143	143	143
2	86	127	147	147	147
3	147	150	150	150	150
4	91	130	175	175	175
5	177	102	177	177	199
6	94	94	199	130	0
7	150	91	130	102	86
8	102	86	102	94	91
9	175	175	94	91	94
10	130	177	91	86	102
11			86		130

通过计算可得各算法的磁头移动次数如下。

(1) FCFS：565　　(2) SSTF：162

(3) SCAN：169　　(4) LOOK：125

(5) C-SCAN：386

例 7-3　列出操作系统在处理一个 I/O 中断时所采取的步骤。

解　操作系统在处理一个 I/O 中断时所采取的步骤如下：

(1) 保留寄存器；　　(2) 确定中断原因；

(3) 检查相应的设备表；　　(4) 如果有必要就等待；

(5) 处理中断请求；　　(6) 修改有关表格（如果必要）；

(7) 恢复寄存器；　　(8) 返回到用户程序。

例 7-4　I/O 软件设计的目标是什么？以下各项工作是在 4 个 I/O 软件层的哪一层完成的？

(1) 为一个磁盘读操作计算磁盘扇区、寻道和定位磁头的时间。

(2) 维护一个最近使用的块的高速缓存。

(3) 向设备寄存器写命令。

(4) 检查用户是否有权使用设备。

(5) 将二进制整数转换成 ASCII 码以便打印。

解　I/O 软件的总体目标是，按分层的思想构造软件，较低层的软件要使较高层的软件独立于硬件，较高层的软件则要向用户提供一个友好、规范、清晰的界面。I/O 软件设计的具体目标是：设备独立性、统一命名、 同步/异步传输、出错处理、设备共享与独占。

(1) 由驱动程序完成。　　(2) 由与设备无关的软件完成。

(3) 由中断处理程序完成。　　(4) 由与设备无关的软件完成。

(5) 由用户程序完成。

例 7-5　通过“再紧缩”信息可消除存储设备上的碎片，但磁盘和磁鼓一般没有定位或基址寄存器，那么怎样才能对文件进行重定位呢？为什么通常要避免对文件的再紧缩和重定位？

解　辅存上没有定位或基址寄存器，所以文件的重定位需要相当高的开销，必须用将数据块读入主存再将它们写回到辅存的新地址处的方法进行重定位。此外，重定位寄存器仅适合于顺序文件，而许多磁盘文件都不是顺序文件。基于同样的原因，许多新的磁盘文件并不要求连续的磁盘空间。若磁盘系统保存着逻辑连续块之间的链接，那么顺序文件也可以分配在非连续的块上。因此，要尽量避免文件的重定位。

例 7-6　当平均队列长度较小时，所有的磁盘调度算法都将退化为 FCFS 调度算法，请说明原因。

解　当队列长度 L 为 1 时，算法 FCFS、SSTF、LOOK 和 C-LOOK 功能都是等同的。当 L 为 2 时，通过把 FCFS 和 SSTF、LOOK 和 C-LOOK 作比较，就会发现它们功能等同的情况大约是 50%，而且访问的局部性也可以使调度算法之间差异变得极小。

例 7-7　除 FCFS 外的所有磁盘调度算法都不是真正公平的（例如，会出现饥饿现象）。

(1) 说明为什么？

(2) 提出一个确保公平性的方案。

(3) 为什么公平性在分时系统中是一个很重要的指标？

解　(1) 对于位于当前磁头所在的磁道上的新请求，从理论上讲，只要它们一到达就可得到服务，而对于位于其他磁道上的请求则不然。

(2) 预定一时间限额，把所有在这期间内尚未服务的请求“强行”移到队列的顶部，并置其相关的位以指明任何新的请求都不得移到这些请求之前。对 SSTF 而言，必须相对于这些“老”请求的最后一个，重新组织队列的剩余部分。

(3) 将公平性作为分时系统中的重要指标的目的是，避免响应时间过长而使某些进程等待太久。

例 7-8　假定请求是均匀分布的，试比较 C-SCAN 和 SCAN 的吞吐量。

解　二者的主要差异在于 C-SCAN 浪费了从磁盘的内边沿到外边沿的搜索时间。另外，下面的事实也会引起二者在吞吐量方面的差异，对内磁道的两次搜索中，SCAN 只搜索一次边沿磁道，而且 SCAN 对一给定磁道的连续访问之间的时间间隔的变化比 C-SCAN 要大。

例 7-9　为什么磁盘调度中通常没使用等待时间（latency）优化技术？

解　这会发生定时问题：进行等待时间定序的程序需要知道在 I/O 服务时隶属于磁头的扇区数。该信息必须在服务该 I/O 请求之前尽快给出，以确保准确性。这就需要与外设进行额外的通信，从而浪费了时间，以致失掉了某些收益。

例 7-10　随着半导体存储器价格的下降，一些厂家已开始制造半导体“盘”。这些新设备使用存储芯片而不是传统的磁盘作为存储器。这样的设备没有移动部件，因此，它们比传统盘的存取速度更快，工作更可靠。这些新设备常常设计成与现在的盘在插件级兼容，因此，其程序设计和寻址方式与传统的盘完全相同，但它们的存取速度却快得多。这种情况将对磁盘调度算法的选择有何影响？

解　由于没有移动部件，所以可能有一个不依赖于被访问地址的固定的查寻或等待时间。在这种情况下，这些新设备更适合选用磁鼓调度算法而不是磁盘调度算法。而且，假如这些“半导体”盘足够快，则磁盘队列长度很可能至多为 1，

因此，调度算法总是相同的。这样，FCFS 就是最合适的调度算法了。

例 7-11　SSTF 算法比较有利于中间柱面的磁道，说明这是为什么？

解　磁盘的中心是到盘中所有其他磁道的平均距离最短的位置，这样在服务了第一个请求后，磁头将很可能更靠近中心磁道。因此，使用 SSTF 算法将有更多的机会首先到达中心磁道。

一旦到达某个特定的磁道上，SSTF 往往把磁头保持在该磁道附近，这样就可以调整最初的趋势而使磁头向中间移动。

例 7-12　假定有一台可动头的磁鼓设备。磁头可读／写数据的位置有 3 个（道 0，1，2）。每道有 8 个记录（0~7），磁鼓转一周需 8ms。在 8ms 中磁头也可在相邻道之间运动，如图 7.2 所示。

假设处理这台磁鼓的程序接受下面的读地址表（见图 7.2）。

道号	记录号
0	2
1	2
1	3
1	5
2	4
2	3

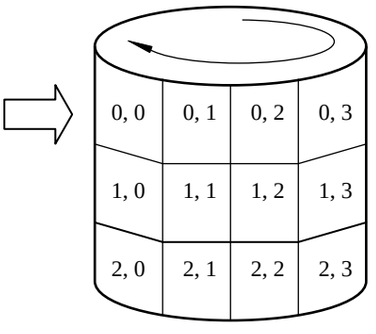


图 7.2

(1) 按给定的次序（即 FCFS）执行以上的 I/O 请求将占多少时间？设开始时磁头在位置（0，0）上。

(2) 为这些请求服务的最佳次序是什么？

解　(1) 由于磁鼓是单向旋转，所以磁头在读取记录号为 N 的记录后需旋转一周方可读/写另一道上的 N 号记录，因此，按照给定次序执行 I/O 请求所需时间为：(2 + 8 + 1 + 2 + 7 + 7) ms = 27ms

(2) 显然，请求服务的最佳次序应使磁鼓每旋转一周后都能响应尽可能多的服务请求，按照扇区排队的方法可得出最佳次序为：

(0，2)，(1，3)，(1，5)，(1，2)，(2，3)，(2，4)

例 7-13　实现 SPOOLing 技术系统需付出哪些代价？

解　(1) 占用大量内存作为外设间传输数据用的缓冲区，系统所用的表格页占用不少内存空间。

(2) 占用大量磁盘空间用作输入和输出。

(3) 增加了系统的复杂性。

例 7-14　某系统主机与 I/O 设备（读卡机，打印机）具有并行工作能力，现希望在单道环境下完成一叠卡片的输入和打印工作，假设读卡机在读完一张卡片、读不到卡片或设备出错时会产生中断，打印机在打完一张卡片的内容或设备出错时会产生中断。若用一主控程序、两个中断处理程序来完成任务，请绘出这些程序的框图。

要求：(1) 采用双缓冲技术，充分利用并行能力。

(2) 假设执行前，卡片已装上卡片机。

(3) 一叠卡片读完并打印完，转向结束。

(4) 输入时出错，转向结束。

(5) 打印时出错，重新打印，重复出错三次，转向结束。

解　读卡机中断程序和打印机中断程序如图 7.3 和 7. 4 所示。主控程序如图 7.5 所示。

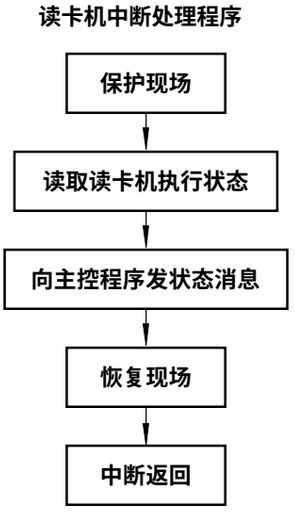


图 7.3

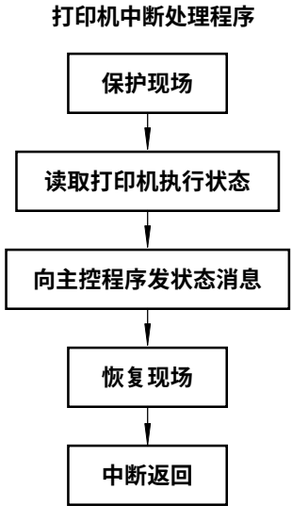


图 7.4

例 7-15　为什么打印机的输出文件在打印前通常都假脱机输出到磁盘上？

解　打印机是低速独占设备，在多道程序的系统中如果有多个程序同时要求打印，就会出现错行打印的问题，假脱机技术用磁盘存储器模拟独占设备的操作，使一台独占设备变为多台可并行的虚拟设备，即把独占设备变成逻辑上的共享设备。使得多个程序能同时完成打印操作，而不必彼此等待。

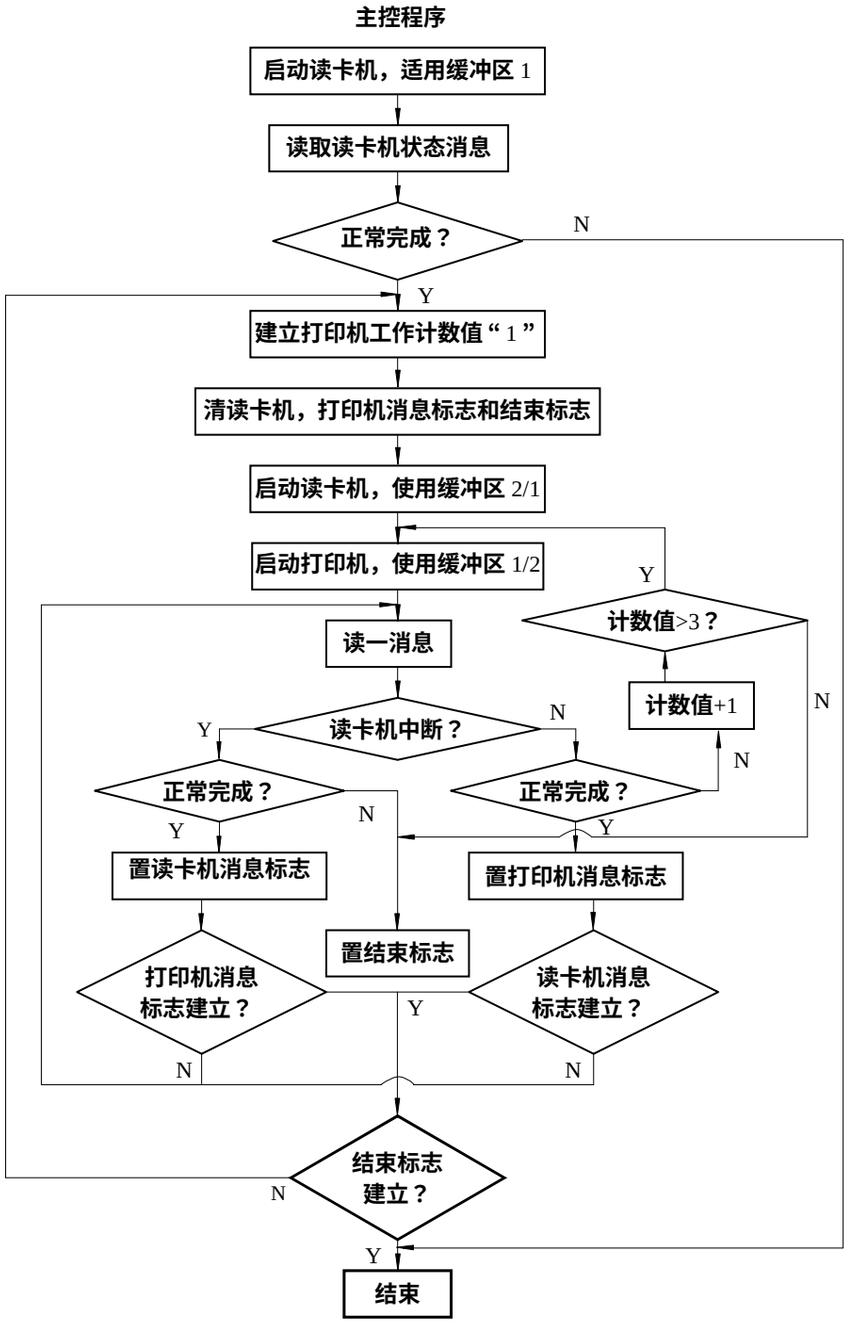

主控程序
启动读卡机，适用缓冲区 1
读取读卡机状态消息
正常完成？
N
Y
建立打印机工作计数值“ 1 ”
清读卡机，打印机消息标志和结束标志
启动读卡机，使用缓冲区 2/1
启动打印机，使用缓冲区 1/2
Y
计数值>3？
读一消息
计数值+1
N
读卡机中断？
Y
N
N
正常完成？
正常完成？
N
Y
Y
置读卡机消息标志
置打印机消息标志
置结束标志
打印机消息
标志建立？
读卡机消息
标志建立？
Y
N
N
结束标志
建立？
N
Y
结束

图 7.5

7.3　自 测 练 习

一、填空题

1. 通道按信息交换的方式可分为_____、_____和_____三种。

2. 总线结构中通常采用的 I/O 方式有_____和_____两种。其中，前者又叫_____，后者又叫_____。

3. I/O 系统大致分为三个层次：底层的_____、中层的_____和高层的_____。

4. 管理和分配设备时，常用的技术有_____、_____和_____三种。

5. 缓冲区的组织方式分为_____、_____和_____三种。

6. 完成一个磁盘服务请求的总时间由_____、_____和_____组成。

7. SCAN 算法在实现时通常改进为磁头在当前方向上，无请求时便反向，此时该算法称为_____算法。

二、判断题（正确的在括号中记√，错误的记×）

1. 磁盘比磁鼓更适合用作分页设备。（　　）

2. 当平均队列较小时，所有磁盘调度算法都退化为 FCFS 算法。（　　）

3. 所有磁盘调度算法都不是真正公平的。（　　）

4. 与 CPU 相比，通道处理 I/O 的功能较强，但价格较高。（　　）

5. 在 SPOOLing 系统中，对于用户进程的设备申请，系统将物理字符设备按时间片方式分配给用户进程使用。（　　）

6. 一个盘块的物理地址是由三个参数唯一确定的，它们是柱面号、盘面号和扇区号。（　　）

7. 公用缓冲区方式通常采用缓冲池。（　　）

8. 扇区排队算法只能用于固定头设备。（　　）

9. 最短寻道时间优先法是最优的磁盘寻道算法。（　　）

10. C-SCAN 算法有时也被称为“电梯算法”。（　　）

三、选择题

1. SPOOLing 系统在工作过程中会和下面哪些操作系统的组成部分发生联系？（　　）

a) 内存管理；　　b) 处理机管理；

c) 文件管理；　　d) 设备管理。

2. 下面哪些信息是磁盘请求所需的？　　(　　)

a) 盘地址；　　b) 内存地址；

c) 传输长度；　　d) 调度算法。

四、问答题

1. 设备管理的主要目标和所实现的功能各是什么？

2. 简述 SPOOLing 技术及其特点。

3. 通道的作用是什么？按信息交换方式它分为几类？

4. 假定有一个具有如图 7.6 所示结构的程序，其中循环几千次，该程序为打印机输出使用了缓冲方案，一次打印一行。

图 7.6

如果这样一个程序在具有 SPOOLing 功能的操作系统下运行，则操作系统必须完成 I/O 程序设计，问：I/O 必须做什么？它们如何影响程序的运行速度？如果若干这样的程序同时运行，会增加系统的吞吐量吗？

7.4　自测练习答案

一、填空题

1. 通道按信息交换的方式可分为字节多路通道、选择通道和成组多路通道三种。

2. 总线结构中通常采用的 I/O 方式有中断处理方式和 DMA 方式两种。其中，前者又叫字符传输方式，后者又叫块传输方式。

3. I/O 系统大致分为三个层次：底层的 I/O 中断管理程序、中层的设备驱动程序和高层的服务软件。

4. 管理和分配设备时，常用的技术有独占、共享和虚拟三种。

5. 缓冲区的组织方式分为：单缓冲区、多缓冲区和缓冲池三种。

6. 完成一个磁盘服务请求的总时间由查找时间、等待时间和传输时间组成。

7. SCAN 算法在实现时通常改进为，磁头在当前方向上无请求时便反向，此时该算法称为 LOOK 算法。

二、判断题

1. ×　　2. √　　3. ×　　4. ×　　5. ×
6. √　　7. √　　8. ×　　9. ×　　10. ×

三、选择题

1．a)、b)、c)、d)　　2．a)、b)、c)

四、问答题

1. **见知识要点。**
2. **见知识要点。**
3. **见知识要点。**
4. **程序每次寻求打印一个缓冲区，**I/O **必须被假脱机到某个二级存储器设备上，在程序执行结束时，必须打印出假脱机的输出。由于缓冲区可能很快地清空，因此，运行速度有可能提高。当若干道程序同时运行时，系统吞吐量一般也不会恶化。如果缓冲区不总是为输出做好准备，或者如果更有效地使用二级存储器使其产生并行请求而不是串行请求，则系统吞吐量可能会更好。**

第 8 章

文 件 管 理

8.1 重点与难点

8.1.1 概述

文件是由操作系统定义并实现的一种抽象数据类型，它是逻辑记录的序列。操作系统可支持不同的记录类型，当然它们也可由应用程序来定义。文件系统的主要功能是将逻辑文件概念映射到物理存储设备中，并采取合适的方式管理这些文件。由于设备的物理记录大小不同于逻辑记录大小，所以有必要将逻辑记录块存入物理记录中。此工作既可由操作系统来做，也可由应用程序来做。

多数文件系统都是建立在磁盘上的。磁带常用来在机器或后援存储器之间进行数据传输。在磁盘系统中，文件既可顺序存取也可直接存取，它在磁盘中的分配方法有三种：连续、链接和索引。

文件系统通常采用创建目录的方法来组织文件。

文件系统还可以对文件进行保护。存取文件时可以对不同的存取类型分别进行控制：读、写、执行、添加、列目录等等。文件保护可通过提供口令、存取表及其他一些特别技术来实现。

文件系统通常都是用层次模块结构来实现的。低层涉及到存储设备的物理特性，高层涉及符号文件名及文件的逻辑特性，中间层将逻辑文件概念变换成物理设备特性。

通过本章的学习，了解文件和文件系统的知识，掌握各种文件的组织结构和存储结构，理解并会应用文件管理策略，掌握一些实际系统的知识。

8.1.2 知识要点

1. 文件与文件系统

(1) 文件的定义

文件是由创建者定义的相关数据的集合，通常存放在外存上，可以作为一个独立单位来实施相应的操作(如打开、关闭、读、写等)。为便于管理和控制文件，常将文件分成若干类型，由操作系统支持。

(2) 文件系统的功能

操作系统中负责管理和存取文件信息的软件机构称为文件管理系统，简称文件系统。它为用户提供了存取简便、格式统一、安全可靠的管理信息的方法，提供了用户与外存的接口。一般来说，文件系统具有下列主要功能：

① 实现按文件名存取文件信息的操作，完成从文件名到文件存储物理地址的映射。

② 文件存储空间的分配与回收。

③ 对文件及文件目录的管理。

④ 提供(创建)操作系统与用户的接口。

⑤ 提供有关文件自身的服务，如文件的安全性、文件的共享机制等。

2. 文件结构及存取方法

文件的结构是指文件的组织形式，文件的结构有两种：一种是逻辑结构，是从用户观察和使用文件的角度出发定义的文件组织形式；另一种是物理结构，是从系统的角度考察文件的物理存放形式。

按文件的逻辑结构可将文件分为：

① 字符流式文件。

② 记录式文件。

其存取方式有：

① 顺序存取。

② 随机存取。

③ 按键存取。

按文件的物理结构可将文件分为：

① 顺序文件。

② 链接文件。

③ 索引文件。

其存储设备有：

① 顺序存取设备。

② 直接存取设备。

文件的物理结构，必须适应文件的存储设备，而不同的存储设备的特性，又决定了其上的文件的存取方式。

3. 文件管理

(1) 文件控制块及文件目录

文件系统提供了文件的按名存取机制，将文件名转换为存储地址及对文件实施控制管理就需通过文件目录来实现。

在文件系统内部，给每个文件设置了一个文件控制块，文件控制块中通常包括如下信息：文件名、文件类型、文件地址、文件大小、保护位、使用次数、时间、日期、拥有者等等。为加快文件检索速度，文件系统将控制块集中管理，这种文件控制块的有序集合就是文件目录。目录上的主要操作有：查找文件，创建文件，删除文件，列文件名，复制文件等。

(2) 文件目录所使用的数据结构

在组织目录时，可以采用的数据结构及其优缺点如下：

① 线性表。优点是查找简单，缺点是存取文件的速度较慢。

② 链接表。优点是易于实现增、删操作，缺点是需要为指针分配存储空间，而且当指针指错位置时，其后的文件信息或者会丢失或者会发生混乱。

③ 排序表。优点是查找快速，缺点是排序表总是要求所需的表是排过序的，这意味着在创建和删除文件时需要做一些额外的工作。

④ 链接二叉树。优点是查找速度较快，缺点与链接表相同。

⑤ 杂凑表。优点是查找速度最快，缺点是要事先考虑到文件的最大个数，而且还得设法解决冲突问题。

(3) 目录的组织形式

目录的组织方式直接关系到用户对文件的存取是否方便。通常采用的文件目录结构有：单级目录、二级目录、多级(树形结构)目录和非循环图目录。

单级目录结构简单，设备目录就是单级目录。其缺点是，用户必须谨慎地选用文件名，以免发生名字冲突；用户没有保密措施，文件可能会被他人破坏。

二级目录为解决不同用户的文件名字冲突问题，为每一用户创建了一个文件目录，称为用户文件目录(UFD)。每一用户目录有相似的结构，但只能列出单用户文件。用户作业开始或注册时，查询主文件目录(MFD)，主文件目录用用户名或账号索引，指向各用户的 UFD。采用二级目录的优点是：用户是彼此隔离的，

它们在选用文件名时有较大的自由度。采用二级目录的主要缺点是：如果没有其他规定或措施，则会使两个或多个用户由于难以互访相关的文件而不能相互合作，而且不可存取系统文件。对于前者，可用链结构把用户目录链接起来，并用创建路径名的办法来克服。对于后者，可让命令解释程序首先查寻目录，若所需文件不在其中，则再用查寻系统目录的办法来解决。

在采用二级目录时，可通过下列方法访问公共文件和程序：

① 在每个用户的注册名中，复制每个公共文件。

② 在一个特定的系统文件中汇集公共文件。由系统把对这些公共文件的直接访问命令，转换成对现在这个特定系统文件中的文件进行访问的路径名访问的形式。

③ 允许使用从一个目录到另一个目录的路径名形式。

多级目录是由二级目录推广而来，多级目录常采用树形结构来组织。树形结构目录允许用户在目录下定义自己的子目录。使用子目录的原因是子目录有利于分类组织、管理文件，从而达到方便使用和保护文件的目的。目录中文件由文件路径名来识别。文件路径名是一个目录和(或)子目录(或文件名)的序列，根据这个序列可从主目录到达指定的文件。

非循环图目录比树形结构目录便于文件和目录的共享。通常创建一个被称为链的新目录条目，它是有效地指向另一文件及子目录的指针，但这种方式查询及删除都较麻烦。另一种实现共享文件的方法是将共享文件的所有信息复制到两个共享目录中。目录结构一般不会采用普通图形形式组织，否则将会出现一些问题，如：对某一特定文件的查寻可能导致对同一目录的多次查找；删去一个文件可能导致访问计数为非 0，甚至没有包含此文件的目录也是如此。

为回收可用空间，文件系统常进行“无用单元收集”(garbage collection)，即确定哪些文件空间是可用的，调整它们并做上可供用户使用的标记。

文件的目录多是以目录文件的形式存放的，存取一个文件往往需要访问多级文件目录，系统通过把当前正在使用的那些文件的目录表复制到内存中，来提供良好的目录管理，提高文件的访问速度。

4. 空间分配与管理

文件系统管理外存时通常会出现空间浪费问题，其表示形式一般分为内部碎片和外部碎片两种。内部碎片是在以块为单位分配数据区时所浪费掉的字节。这主要是由大多数文件末端的未用字节所引起的。外部碎片是一些独立的空闲空间，但它们中任何一个都无法满足一个新的分配要求。外部碎片问题可以通过使用不会导致碎片的分配技术，或者把存储空间中已有的文件尽可能移到一起，以获得较大的空闲空间的方法来解决。

分配磁盘空间的三种主要方法是：

(1) 连续分配

连续分配是为一个文件分配一片连续的区域。其主要困难是如何为一个新文件找到合适的连续空间。在连续分配方法中，块中未使用的片段称为“空穴”，分配问题就在于如何从空穴表中找出满足需要的适配方法。

连续文件分配空间中常用的适配方法有：首次适配(first fit)，最佳适配(best fit)和最差适配(worst fit)方法。

为了避免文件无法扩展的问题，连续分配方法使用称为预分配的方法，即在创建一个文件之前就为它分配空间，以允许其扩张。这样就为一特定文件保留了空间，其他文件就不能占用它。新文件最初可能只使用这一保留区的一小部分(以后再扩张)。

(2) 链接分配

链接分配是指用这样的方式来进行存储分配：文件的每一块(最后块除外)都含有指向下一块的指针，并用一目录结构存放指向每一文件第一块和最末块的指针及该文件的名字。

(3) 索引分配

在索引分配中每个文件有它自己的指针块，以指向相关文件的对应区段。

上述三种分配方法各有其优点，均可用于一般的存储分配。连续分配易于实现，且可随机存取文件，但会出现外部碎片；链接分配不会产生外部碎片，文件的伸缩也易于实现，但可能产生内部碎片；索引分配支持无外部碎片的直接存取。

从速度上来看，连续分配最快；链接分配较慢，因为磁头可能不得不在存取文件之间移动；索引分配最慢，除非在任何时刻整个索引都放在内存中，否则必须花一定时间去存取文件索引的下一块。

文件的磁盘存储空间的管理包括磁盘空间块的分配和回收。磁盘空间的有效管理有助于提高整个操作系统的效率，它涉及到盘块大小的划分及管理的方式。

盘块可分为逻辑块和盘区，对盘块的管理方法有：

① 盘图法。也称为字位映像图，是一种常用的方法，它用位(bit)的值来表示磁盘上相应物理块是否被分配。

② 链接法。首先是选择若干空闲物理块建立索引表块，而后将这些含有空闲块号的索引块之间用链接方式链接起来。

5. 文件保护

系统保护用户文件主要有下面几种方法：

① 允许用户在命名文件时使用不可打印的字符，因此，别人就难以猜出完整的文件名。

② 给每一文件指定口令，在允许访问文件前得先给出口令。

③ 构造一访问表，表中列出允许使用的每一文件(或相应文件)的用户名。

④ 根据用户的情况(如系统、拥有者、拥有团体、地址、身份等)分类，再给每一文件指定相应的保护位。

8.1.3 系统实例

1. Windows NT

有关 Windows NT 操作系统在文件管理方面的信息可以参考课本[1]中 8.5 节的内容。

2. Unix System V

Unix 内核把所有文件看成字节流。文件的内在逻辑结构是随应用而定的。Unix 注重文件的物理结构，把文件分成如下四类：

① 普通文件(Ordinary)。它是包含来自用户、应用程序和系统设备程序的所有信息的文件。

② 目录文件(Directory)。它是文件名列表和简短的与信息结点相关的指针。目录是分层组织的，实际上是有特别的写保护的文件，以保证只有文件系统可以写文件，用户程序有读权限。

③ 特殊文件(Special)。它用于访问外部设备，如终端和打印机。每个 I/O 设备都与一个文件相联系。

④ 命名文件(Named)。它是命名管道。

Unix 系统中文件由操作系统通过索引结点(又称为 I 结点)管理。索引结点是含有关键信息的一个控制结构，文件的属性、权限和其他控制信息都保存在索引结点中。

文件基于块进行分配。分配根据需要动态进行，并不进行预先分配。磁盘上的文件块不一定需要连续。系统使用索引技术跟踪每个文件，部分索引保存在文件的索引结点中。在 Unix 系统中，块的长度为 1KB，每个块可容纳 256 个块的地址。因此，一个文件最大可达 16GB。

Unix 文件系统的优点在于：

① 索引结点定长，相对较小，这样可以长期保存在主存中。

② 对较小的文件可直接访问，减小了处理和访问磁盘时间。

③ 文件理论上的最大尺寸足以满足所有应用。

3. Linux

Linux 支持许多不同的文件系统，系统可用的独立文件系统不是通过设备标识来访问的，而是将它们链接到一个树形结构目录中。

当前 Linux 主流版本中的文件系统是 EXT2 文件系统。与 Unix System V 相似，EXT2 文件系统也采用了 inode 数据结构来表示系统中的每个文件。inode 结构包括文件占用数据块、文件访问权限、文件的更改时间、文件类型等信息。文件系统的所有 inode 结点记录于 inode 表中。EXT2 文件系统将其逻辑分区划分为块组，各块组除了复制文件系统的重要信息外，还包括以数据块形式存放的物理文件、目录等信息。各块组有一描述符，包括块位图、inode 位图、inode 表等信息。

Linux 系统用虚拟文件系统 VFS 来管理装载各个不同的文件系统。文件系统初始化时向 VFS 注册。当包含文件系统的块设备被装载时，VFS 读入它的超级块，各文件系统的超级块读例程确定整个文件系统的拓扑结构，并将这些信息映射到 VFS 超级块数据结构内。

此外，Linux 系统还使用缓冲区和目录查找缓存机制，以加速对文件系统的访问。

8.2 例 题 解 答

例 8-1 Windows NT 可执行文件的结构是怎样的？

解 Windows NT 的可执行文件格式是可移动文件格式，简称 PE。由 5 部分构成：MS-DOS 首部、PE 首部、信息块表、信息块和辅助信息。其中 MS-DOS 首部在最前面，标明了文件必须具备的运行环境；信息块表的表项描述了特定的信息块；最后是一些辅助信息。

例 8-2 假定有一个当前含有 100 块的文件，如果

(1) 在开头加入一块；

(2) 在中间加入一块；

(3) 在末端加入一块；

(4) 在开头删除一块；

(5) 在中间删除一块；

(6) 在末端删除一块。

那么，使用连续、链接和索引分配策略时各涉及到多少次磁盘 I/O 操作？

解 它们所涉及的操作次数如表 8.1 所示。

表 8.1

序号	连续	链接	索引
(1)	201	1	1
(2)	101	51	1
(3)	1	2	1
(4)	0	1	1
(5)	98	52	1
(6)	0	100	1

例 8-3　能否用一个可使用任意长度文件名的单级目录结构来模拟多级目录结构？若能，则说明如何模拟。如果文件名限制为 7 个字符，你的方案应作哪些修改？

解　如果允许使用任意长的名字，则可模拟多级目录结构。例如，可用符号“.”指明子目录的结束。这样，名字 h.lisp.F1 指明 F1 是子目录 lisp 中的文件，而 lisp 是根目录 h 的文件。

若名字限制为 7 个字符，那么上面的方案可能无用。一个变通的方案是使用一个特定的文件作为符号表(目录)。它将随意长的文件名(如 h.lisp.F1)映射为较短的文件名(如 XX00101)中，然后将后者作为实际的文件名来使用。

例 8-4　有些系统通过保留文件的单一副本来提供文件共享，有些系统则通过对每一共享文件的用户各保留一个副本的方式来提供文件共享，试叙述它们各自的优缺点。

解　保留单一副本，对于一个文件的多次并发更新，可能导致用户得到不正确的信息，并使结果文件处于不正确的状态。保留多个副本则浪费了一些存储空间而且各副本彼此可能会不一致。

例 8-5 考虑这样一个文件系统，其中文件可被删除，并且在指向它的链路仍然存在的情况下可重新使用其磁盘空间。若在同一磁盘空间建立一新文件，将会出现什么问题？如何避免该问题？

解　令 F_1 是老文件，F_2 是新文件，那么，希望通过现有链路访问文件 F_1 的用户实际上访问的是文件 F_2。这里使用的是对文件 F_1 的存取保护而不是与文件 F_2 相关的存储保护。

采用删除指向一个已删除文件的所有链路的方法可避免该问题。这可用如下几种方式实现：

(1) **保留一张记录指向一个文件的所有链路的表，当删除该文件时，就删除这些链路。**

(2) **保留这些链路，当试图访问一个已删除的文件时，就删去相应的链路。**

(3) 保留一个文件访问表，仅当对某文件的访问被删除后或所有链路都被删除之后才删除文件访问表中的相应文件。

例 8-6　假定一系统中，自由空间通过自由空间表管理：

(1) 如果指向该自由空间表的指针丢失了，那么，系统能否重构该自由空间表？

(2) 提出一种不因内存故障而丢失该指针的方案。

解　(1) 为了重构自由空间表，必须执行“无用单元收集程序”。这需要查寻整个目录结构以判定哪些页面已分配给作业，而将那些未被分配的页面重新链接为一个自由空间表。

(2) 可将自由空间表指针存放在磁盘上。

例 8-7　在许多系统中，一个授权的用户可像对通常的文件那样读、写子目录。

(1) 这样做会出现什么保护问题？

(2) 试提出一个处理这些问题的方案。

解　(1) 保存在目录项目中的信息之一是文件位置。如果用户可以修改这个位置信息，那么，他也会访问其他文件，从而违犯了存取保护的规定。

(2) 不允许用户直接向子目录写入信息，而应提供系统操作来完成这些事情。

例 8-8　考虑一个支持连续、链接和索引分配策略的系统，对于一给定文件，决定采用每种策略时的条件是什么？

解　采用连续分配策略的条件是，对文件的访问通常是顺序的，而且文件比较小。

采用链接分配策略的条件是，文件较大且通常是顺序访问的。

采用索引分配策略的条件是，文件较大并通常是随机访问的。

例 8-9　考虑一个存于盘上的文件系统，其中的文件由大小为 512 字的块组成。假定每一文件有一个文件目录项，该目录项包含该文件的名字、文件长度以及第一块(或第一索引块)和最后一块的位置，而且，该目录项位于内存。对于采用索引分配策略的情况，该目录项指明第一索引块，该索引块又依次指向 511 个文件块，且有一指向下一索引块的指针。对连续、链接、索引分配策略中的每一种：

(1) 说明在这个系统中是如何实现逻辑地址到物理地址的映射的；

(2) 如果当前位于逻辑块 10(即最后一次访问的块是逻辑块 10)且希望访问逻辑块 4，那么，必须从盘上读多少物理块？

解　令 m 是开始逻辑地址，用 512 去除逻辑地址，令 x 和 y 分别是除得的商和余数。

若为连续分配，则

(1) 将 x 加上 m 可获得相应物理块的位移；

(2) 1。

若为链接分配，则

(1) 查链接表直至找到所需块号为止。

```
m1：＝m
for i：＝1 to x
    begin
        取位于 m1 的物理块；
        用下一块地址替代 m1 之值；
    end
```

(2) 4。

若为索引分配，则

(1) 将第一索引块读入内存，用 511 去除逻辑块号(x)，设除后所得的商和余数分别为 r 和 s。

```
for i：= 1 to r
    begin
        从当前索引块的地址 512 处获得块地址 q；
        将块地址 q 处的索引块送内存；
    end
利用 s 作为该索引块的位移可获得相应物理块的地址。
```

(2) 1。

例 8-10　当用户自愿撤离或终止一个作业时，某些系统自动删除所有相关的文件，除非用户明显地请求保留它们；而另一些系统则保留所有文件，除非用户明显的请求删除它们。试叙述这两种途径的特点。

解　对于前一种途径，由于不必保留不(再)需要的文件，每一用户所需的文件空间可变成最小。后一种途径，则对用户更为安全，因为用户不会由于疏忽没有保存文件而丢失文件。

例 8-11　什么是文件的存取控制？请列举出三种以上实施存取控制的方案。

解　文件存取控制是指限制文件共享，保护文件的方法。实施文件存取控制的方案有存取控制矩阵、存取控制表、用户权限表、口令密码等。

8.3　自 测 练 习

一、填空题

1. 文件按用途可分为_____、_____和_____等三种；按属性可分为_____、_____、_____和_____等四种。

2. 组织目录时可采取的数据结构有_____、_____、_____、_____

和_____等五种。

3. 分配磁盘空间的三种主要方法是_____、_____和_____。

4. 连续文件分配空间中常用的适配方法是_____、_____和_____等三种方法。

5. 目录上的主要操作有_____、_____、_____、_____和_____等五种操作。

6. 文件的逻辑形式有_____和_____两种。

7. 基于磁盘文件模式，将文件视为编上号的块的文件存取方法称为_____。

8. 二级目录结构由_____目录和各用户自己的_____目录组成。

9. 在 Unix 系统中，文件名/usr/m1/prog/f1.c 称为该文件的_____，若当前目录为/usr/m1，则 prog/f1.c 称为该文件的_____。

二、判断题（正确的在括号中记√，错误的记×）

1. 如果用户极其频繁地访问其当前目录中的文件，那么应将该目录放在内存。 (　　)

2. 连续文件的缺点之一是不便于扩充。 (　　)

3. 树形结构目录的层次和隶属关系清晰，有利于文件和目录的共享。 (　　)

4. 多重索引结构适合于有大量大文件的系统。 (　　)

三、选择题

1. 下面哪种结构不是文件物理结构的一种 (　　)

a) 顺序结构；　　b) 链接结构；

c) 字符流结构；　　d) 索引结构。

2. 下面哪种磁盘空间分配方法会产生内部碎片，但是，不会产生外部碎片？ (　　)

a) 连续分配；　　b) 链接分配；

c) 索引分配；　　d) 首次适配。

3. 下面哪种目录结构允许目录共享子目录和文件？ (　　)

a) 单级目录；　　b) 二级目录；

c) 树形结构目录；　　d) 非循环图目录。

四、问答题

1. 什么是文件、文件系统？文件系统的功能是什么？

2. 考虑一个有 5000 个用户的系统。假定只允许这些用户中的 4990 个用户能

存取一个文件。请问：

(1) 如何实现？

(2) 试提出能更有效地用于此目的的另一种保护方案。

3. 与连续分配相关的问题之一是用户必须为每一文件预分配足够的空间。若使用过程中文件变得大于预分配给它的空间，则须采取相应的措施。解决此问题的办法之一是先定义一个仅由一个初始连续区组成的文件结构，若该初始区域满，则操作系统自动定义一个溢出区并把该溢出区链接到初始区域；若新定义的溢出区也满了，则再分配另一溢出区，等等。试将这种方案与标准的连续分配和链接分配方案作一比较。

4. 在一个具有树形结构的文件系统（见图 8.1）中，其叶子表示文件，中间结点表示文件的目录。问是否允许进行下述操作？为什么？

(1) 在目录 E 中建立新的文件，取名为 A。

(2) 把文件 B 改名为 A。

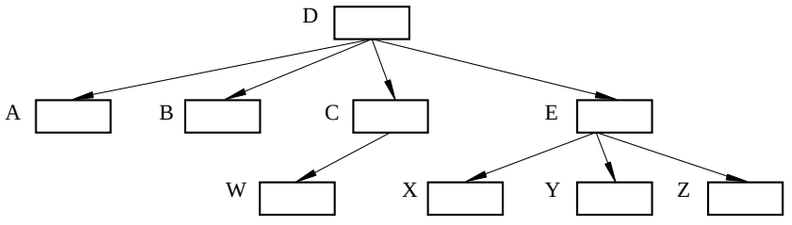


图 8.1

如果 C 和 E 是两个用户各自的目录，再回答下述问题：

(3) 若 C 目录的用户欲共享 E 目录中的文件 X，则 C 目录应做什么处理？

(4) 若每个用户都欲实现自己文件的保密安全要求，则该文件系统应做怎样的控制？

8.4　自测练习答案

一、填空题

1. 文件按用途可分为用户文件、库文件和系统文件等三种；按属性可分为可执行文件、只读文件、可读写文件和非保护文件等四种。

2. 组织目录时可采取的数据结构有线性表、链接表、排序表、链接二叉树和杂凑表等五种。

3. 分配磁盘空间的三种主要方法是连续分配、链接分配和索引分配。

4. 连续文件分配空间中常用的适配方法是首次适配、最佳适配和最差适配等三种方法。

5. 目录上的主要操作有查找文件、创建文件、删除文件、列文件名和复制文件等五种操作。

6. 文件的逻辑形式有字符流式文件和记录式文件两种。

7. 基于磁盘文件模式，将文件视为编上号的块的文件存取方法称为随机存取方法。

8. 二级目录结构由主文件目录和各用户自己的用户文件目录组成。

9. 在 Unix 系统中，文件名/usr/m1/prog/f1.c 称为该文件的全路径名，若当前目录为/usr/m1，则 prog/f1.c 称为该文件的相对路径名。

二、判断题

1. √　　2. √　　3. ×　　4. ×

三、选择题

1. c)　　2. b)　　3. d)

四、问答题

1. (略，见前面知识要点)

2. (1) 有两种方法可实现：

① 建立一个包含所有 4990 个用户名字的存取控制表。

② 将这 4990 个用户合为一组，并对该组设置相应的存取权。由于这组用户是受系统限制的，所以这种方案不一定总能实现。

(2) “everyone else”信息适用于不在某存取控制表中的所有用户。据此，只要把余下的 10 个用户的名字放入该存取控制表但不给予任何存取权即可。

3. 这种方案所需的开销高于标准的连续分配方案，但低于标准的链接分配方案。

4. (1) E 目录允许建立名为 A 的文件。因为同名的另一个文件在目录 D 中，可由不同的目录名加以区分。

(2) 文件 B 不能改名为 A，因为 D 目录下将发生文件重名的错误。

(3) C 目录中应增加一个文件项，使其指向目录 E 中的文件 X，达到共享的目的。

(4) 目录中应该对每个文件项的数据结构增加文件存取控制权的内容。例如，按照文件所有者和非所有者，设置不同的存取控制权：R 表示可读、W 表示可写、E 表示可扩展、D 表示可删除。格式为[所有者权限，非所有者权限]，则[RWED，R]就表示文件所有者可以对该文件进行读、写、扩展和删除，而其他用户只有读的权利。

第 9 章

分 布 计 算

9.1 重点与难点

9.1.1 概述

分布数据计算(Distributed Data Processing，简称 DDP)使处理器、数据和数据计算系统的其他功能分散到一些结构中。分散可以使系统更好地满足用户需要，提供更短的响应时间，而且与集中式方法相比减小了通信的开销。

分布式系统是一个不共享公共存储器的处理机集合，每个处理机都有自己的局部存储器，并通过各种通信线路连接成一个可相互通信的网络，网络中的各站点在规模和功能上可以各不相同。

客户/服务器计算模型是用于信息系统的典型模型，其自身的特点，使之得以迅速地占据计算模型的主导地位。在此模型的基础上，添加一个中间件层，就构成了三层模式的客户/服务器模型，从而为运行在不同平台上的客户与服务器间的通信奠定基础。

在分布式系统中，计算机并不共享存储器，站点（Node）之间是通过消息的传递进行通信。

远程过程调用使得调用远端过程像调用本地过程一样简便易行。

本章的重点主要是对客户/服务器计算模型的理解，掌握中间件技术及其优势，理解分布式消息传递的机制和远程过程调用的执行流程。

9.1.2 知识要点

1. 分布式系统简介

(1) **分布式系统概述**

分布式系统是由多个分散的计算机经互连网络连接而成的计算机系统，其中，各个资源单元(物理或逻辑的)既相互协同又高度自治，在全系统范围内实现资源管理，动态地进行任务分配或功能分配，并能并行地运行分布式程序，分布式系统是松耦合系统。

分布式操作系统是被一个计算机网络共享的操作系统。对于用户，像一个普通的集中式操作系统，但它为用户提供了对若干机器资源的透明访问。一个分布式操作系统要依赖于具有基本通信功能的通信结构。

(2) **分布式系统的拓扑结构**

在分布式系统中，多个站点以某种方式从物理上连接起来，这种连接方式称为网络拓扑。几种常用的网络拓扑结构有如下几种：

① 星形结构：有且仅有一个中央站点与其他所有站点直接相连，除此中央站点外，其他站点都不直接互连，星形结构采用集中式的控制方式，实现简单。

② 环形结构：网中的站点构成一个闭环，每一个站点恰好与两个站点相连，可以通过双向环来提高环结构的可靠性。

③ 总线结构：系统中所有站点都直接与总线连接，任意两个站点都可以通过总线直接通信，总线可以是线状的，也可以是环状的，总线可能成为系统的瓶颈。

④ 树形结构：树形结构是一种层次网络，所有站点组织成树形结构。

(3) **分布式文件系统**

与传统文件系统不同，分布式文件系统的设计主要要考虑如下两个方面：

① 透明性。用户能使用相同的方式存取系统内所有文件而不管其驻留位置。

② 局部性。文件驻留在分布系统中的具体位置。

目前主要有三种在分布式环境中组织文件系统的方法：

① Arpanet FTP**。每个站点管理自己的文件系统，通过** FTP**（文件传输协议）在站点间复制文件。**

② 集中式途径。每个站点管理自己的局部文件系统，其上的文件被该站点的用户所存取，所有共享的文件都驻留在文件服务器上。

③ 分散式途径。每个站点管理自己的局部文件系统，并且局部文件可以由

系统中任何站点的用户所存取。文件系统可以采用文件位置对用户不透明和透明两种方案来实现。

(4) 分布式系统中的资源共享

在资源共享方面，分布式系统实现的方法主要有：

① 数据迁移。当站点 A 的用户希望存取站点 B 的数据时，可以将站点 B 的所有数据传输给站点 A，或者将实际需要的数据部分传输给站点 A。

② 计算迁移。

③ 作业迁移。当一个作业提交给系统后，系统可以在一特定的站点上执行这整个作业，也可以将作业分散到不同的站点上去执行。

在分布式系统中发生故障时，把故障站点隔离出去，使得系统在比原来小的情况下继续运行，称之为分布系统的重构。

2. 客户/服务器计算

(1) 客户/服务器模型

在客户/服务器模型中，动作的请求者在一个系统中，动作的提供者可能在另一个系统中，这种计算方式称为客户/服务器计算。应用的处理被分散到（非必须的）客户和服务器上，由客户初始化，而且部分由客户控制，客户与服务器协作执行一个应用。

客户/服务器模型的计算方式和其他分布计算方式不同之处在于：第一，用户依赖于自己系统中的应用。用户要对应用的执行时间和使用方式进行控制；第二，应用是分散的，但是却要集中共享数据库和一些网络管理的功能；第三，网络管理和网络安全在组织、设计和使用信息系统中有着十分重要的地位。

客户/服务器体系结构的特征是应用级任务可在客户和服务器之间进行分配，客户/服务器数据库是该结构的主要应用之一，并根据客户与服务器分担任务的不同，将客户/服务器应用分为以下四类：

① 基于主机的处理。

② 基于服务器的处理。

③ 基于客户的处理。

④ 协同处理。

(2) 中间件

中间件是为了满足更加统一和方便地使用客户/服务器体系结构的要求而产生的，是一组在应用层之上、通信软件和操作系统层之下，使用标准的编程接口和协议的软件，它们都掩藏了不同网络协议和操作系统之间的复杂细节和差异。中间件产品有许多种，但这些产品都是基于两种机制：消息传递和远程过程调用而产生的。

3. 分布式消息传递

分布式计算系统中计算机并不共享存储器，因此，在这类系统中使用消息传递机制。在最简单的形式中，只需要两个函数：发送(Send)函数和接收(Receive)函数。Send函数说明发送的目标和消息内容；Receive函数说明消息的来源，为消息的存储提供一个缓冲区。在实现中，进程使用操作系统提供的消息传递模块来传递消息。可以通过使用可靠的传输协议和给发送进程一个确认的方法来保证消息的可靠。

4. 远程过程调用

远程过程调用(Remote Procedure Call，即RPC)允许不同机器上的程序通过简单的过程调用/返回算法来交互通讯，就好像这两个程序在同一台机器上一样。它的优点是：已经被广泛应用于各种分布式环境；标准化和精确定义的界面利于程序开发和检错；提供了客户端与服务器端的独立性，使之能各自被独立地开发出来。

远程过程调用中的参数传递和参数表示是两个需要考虑的问题，参数传递一般是使用传值方式传递，因为传地址的开销太大；参数表示一般要参照一个数据表示的标准格式来进行，以消除系统间的差异。

9.2 例 题 解 答

例 9-1　计算机网络和分布式系统的区别是什么？

解　计算机网络和分布式系统的区别如表 9.1 所示。

表 9.1

序号	网络系统	分布式系统
1	各站点均有自己的操作系统，资源归局部所有，并被局部控制，进程调度主要是通过迁移进程和数据实现	每个站点运行全局操作系统的一部分，多机之间相互协同，平衡系统的负载
2	每个用户使用自己的机器	用户任务（进程）在系统内各计算机间动态调度
3	用户知道资源存放在何处	资源由操作系统管理，其位置对用户透明
4	容错能力差	有一定容错能力，可适度降级处理
5		比网络有更高级别和更高程度的透明性

例 9-2　试叙述构造分布式系统的主要原因。

解　构造分布式系统的主要原因有四点，这四点也是分布式系统的优点。

(1) **资源共享。**

(2) **提高计算速度。**

(3) **提高可靠性。**

(4) **便于通信。**

例 9-3　比较集中式文件系统和分散式文件系统的特点。

解　在集中式文件系统中，所有的共享文件都驻留在单一的中央站点上，管理比较容易，但是，文件服务器会成为瓶颈；在分散式文件系统中，共享文件可驻留在任何站点上，管理局部表和存取过程需要花费更多的开销。

例 9-4　试叙述 Bully 算法。

解　当进程 P_i 给协调者发送一条请求消息时，在一定的时间间隔 T 内没收到回复消息，P_i 便假定该协调者发生故障，试图选择它自己作为新的协调者。

进程 P_i 给每个优先级高于 i 的进程发送一条选择消息，然后，在时间间隔 T 内等待来自这些进程中的任何一个的回复消息。

如果在时间间隔 T 内没收到任何回复消息，那么，P_i 便假定优先级大于它的进程都发生了故障，于是 P_i 选择自己作为新的协调者，并向优先级小于自己的进程发送消息，通知它们自己是新的协调者。

如果 P_i 收到一条回复消息，那么它就等待另一个时间间隔 T，准备接收新协调者的通知。如果在 T 内没有收到消息，那么，就假定这个优先级大于它的进程发生了故障，重新开始选择算法。

例 9-5　试叙述远程过程调用机制的结构以及实现方法。

解　远程过程调用机制由以下四部分构成：

(1) stub。client **和** server **各一个。**

(2) **约束（**binding**）。使** client **能定位到相应的** server **中。**

(3) **控制。为追踪远程过程调用状态所设。**

(4) **传输。确定如何将信息从一个站点传送到另一个站点。**

实现远程过程调用的步骤如图 9.1 **所示。**stub **包含了一组远程过程调用机制**

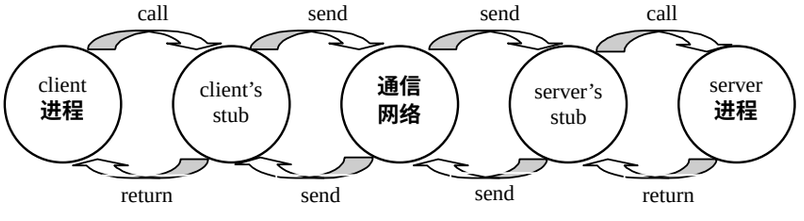


图 9.1

的操作原语，这些原语构成了远程过程调用的实现细节，可独立于 client 和 server 进行编程，在编译后再连接起来。

client's stub 与一个 client 连接，它对于该 client 就像一个“server”。在调用时，它在截取 client 的远程过程调用命令后，利用通信网络向 server 发送“请求服务”的信息。在返回时，它获取返回消息，并带返回结果返回到 client，然后，client 继续执行。

server's stub 与一个 server 连接，它对于该 server 就像一个“client”。在调用时，它收到远程调用请求后，产生一个本地调用，去执行被请求的远程过程。在返回时，它截取远程过程的返回结果，并形成返回消息发送出去。

例 9-6　中间件的主要作用是什么？

解　为了更广泛、便利地使用客户/服务器体系结构，消除在不同系统中客户和服务器之间通信的不便，开发了三层客户/服务器结构，即客户通过一个中间层，也就是中间件和服务器进行通信。中间件提供了对所有平台都一致的资源访问接口，掩盖了不同网络协议及操作系统间的复杂细节和差异，为更好地使用客户服务器结构提供了良好的基础。

例 9-7　传统的 PRC 模型缺乏通用性，如果要多次用到同一个远程过程，则需多次调用。试设计一个一次调用多次接收返回结果的 PRC 模型。

解　可以利用带有 Cache 的 Agent 来满足该要求。client 提出的请求不直接送给 server，而是首先送给 Agent，然后由 Agent 在 Cache 中查找，如果有历史记录，则直接送回结果，否则，根据实际情况送到相应的 server，server 的返回结果送给 Agent，Agent 把结果送给相应的 client，同时，在 Cache 里保存结果。其结构如图 9.2 所示。

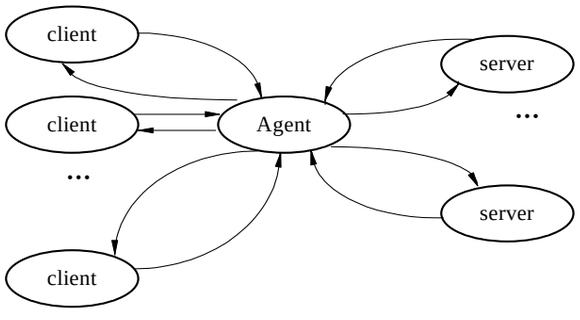


图 9.2

9.3　自 测 练 习

一、填空题

1. 分布式操作系统主要的优点是：_____、_____、_____和_____。

2. ISO 定义的 OSI 参考模型的七层由高到低分别为_____、_____、_____、_____、_____、_____和_____。

3. 分布式系统是由若干_____的计算机通过网络连接起来，并配上分布式系统软件的系统。

4. 在分布式系统中，当协调者发生故障时，使用_____算法来生成一个新协调者。

5. 远程过程调用将实现网络七层协议中_____层的功能，在两个试图进行通信的站点之间建立一条_____信道。

二、判断题（正确的在括号中记√，错误的记×）

1. 松散耦合系统和紧密耦合系统均为分布式系统。（　　）

2. 在集中式的文件系统中，用户可以以 client/server 的方式访问共享文件。（　　）

3. 在星形结构中，如果作为协调者的中央站点发生故障，可通过 Bully 算法来选择新协调者。（　　）

三、选择题

1. 如图 9.3 所示的网络拓扑结构是（　　）。

a) 环形；　b) 总线形；　c) 星形；　d) 混合形。

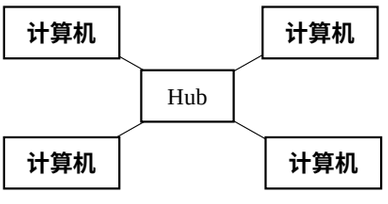


图 9.3

2. 在分布式系统中，主机 1 的用户要修改主机 2 上的文件，将该文件从主机

2 传送到主机 1 供用户使用，此种文件的共享方式为（　　）。

a) 数据迁移；　b) 计算机迁移；　c) 进程迁移；　d) 以上都是。

四、问答题

1. 分布式系统的优点有哪些？
2. 试述远程过程调用的功能以及通信模型。

9.4　自测练习答案

一、填空题

1. 分布式操作系统主要的优点是：资源共享、提高计算机速度、提高可靠性和方便通信。

2. ISO 定义的 OSI 参考模型的七层由高到低分别为应用层、表示层、会话层、传输层、网络层、数据链路层和物理层。

3. 分布式系统是由若干 自治 的计算机通过网络连接起来，再配上分布式系统软件的系统。

4. 在分布式系统中，当协调者发生故障时，使用 选择 算法来生成一个新协调者。

5. 远程过程调用将实现网络七层协议中 会话 层的功能，在两个试图进行通信的站点之间建立一条 逻辑 信道。

二、判断题

1. ×　　2. √　　3. ×

三、选择题

1. b)　　2. a)

四、问答题

1. 见例题解答。

2. 远程过程调用已经成为计算机、终端用户、进程、网络之间进行信息交换的一种标准规程。其功能类似于本地过程调用，因此，它可向应用层和用户提供良好的接口。远程过程调用的服务程序具有可独立编程的特点，有利于系统扩充其服务功能。从应用角度看，远程过程调用基于过程的远程通信特点，为网络和分布式系统的应用提供了灵活方便的通信功能。

远程过程调用的通信模型是基于 client/server 进程之间相互通信模型的一种同步通信形式，它对 client 提供了远程服务的过程抽象，其底层消息传递操作对 client 是透明的。在远程过程调用中，client 是请求服务的调用者，server 是执行 client 请求而被调用的程序，它们之间存在这样的通信模型：

对于 client 方：

① 发送“请求服务”的消息给远方的 server。

② 等待，直至 server 对该请求发送一个回复为止。

③ 一旦 client 接收到来自 server 的回复消息，就获取相应的调用结果，然后继续执行。

对于 server 方：

① 等待 client 的调用信息而处于倾听状态。

② 一旦调用消息到达，server 就截取调用参数，并执行指定的服务程序。

③ 将执行后的返回结果，以回复消息形式传递给 client。

第 10 章

分布式进程管理

10.1 重点与难点

10.1.1 概述

分布式进程管理是分布式操作系统的关键核心部分，它涉及到很多内容。其中，进程迁移是当前分布式操作系统中的热门话题，还有不同系统上的进程间的协调，以及分布进程管理的主要问题：互斥和死锁等也是很重要的内容。

分布式操作系统与集中式操作系统不同，在分布式操作系统中，由于没有全局状态，同步问题的设计策略将更为复杂。一般用一个分布式快照算法来解决这个问题，该算法可用记录了一致全局状态的分布式快照，来为同步问题提供一致的全局状态。

由于分布式操作系统中没有共享存储器和时钟，故用时间戳来确定系统中事件发生的顺序。事件发生顺序用偏序先于关系来表示。并以此为基础提出三个算法：Lamport 算法、Ricart 算法和令牌算法来解决分布式操作系统中的互斥问题。

在分布式操作系统中，死锁问题也变得更加复杂，源于没有一个站点能精确地知道系统的全局状态。分布式操作系统有两种类型的死锁：资源分配引起的死锁和消息通信引起的死锁。与集中式操作系统一样，分布式操作系统用死锁预防、死锁检测、死锁避免等方法去解决死锁问题。

本章的学习要求是，了解分布式操作系统中进程管理的独特之处以及相关知识点；理解掌握分布式快照算法、分布式互斥算法，会用这些算法解决和判断问题；与集中式操作系统相比较地学习掌握分布式死锁问题。

10.1.2　知识要点

1. 进程迁移

进程迁移就是将一个进程的状态，从一台机器（源机）转移到另一台机器(目标机)上，从而使该进程能在目标机上执行。在分布式系统中主要是以下原因要对进程进行迁移：

① 负载共享。

② 减少通信开销。

③ 可获得性。

④ 利用特定资源。

进程的迁移要有迁移机制保障其正确运行，在设计迁移机制时要考虑如下三个问题：

① 由谁激发迁移。

② 迁移什么。

③ 未完成的消息、信号的处理。

为了保证目标机的用户响应时间，需要目标系统参与决定进程的迁移，就需要协商机制以提供更好的整体平衡性。协商进程还允许一个系统将迁移到其上的进程驱逐出去，以提供良好响应特性。

根据被迁移进程的运行状态，进程迁移可分为：抢占进程的迁移和迁移非抢占进程两种。二者的区别在于是否允许迁移部分运行的进程。非抢占进程的迁移在负载平衡中很有用。其优点是可以避免频繁的进程迁移，其不足在于对负载分配的突变反应不灵敏。

2. 分布式全局状态

要获得远程进程的状态，只能通过消息原语接收状态信息，但这些状态信息只表示远程进程某一时刻的状态，并不能记录一个系统的全局状态。分布式快照能记录一个一致的全局状态。如果对于每个记录，接收消息的进程状态都有该消息的发送记录在发送消息进程的进程状态中，那么全局状态是一致的。

在分布式快照算法中，假设消息按序发送，并且没有丢失，则算法用到一个专门的控制消息，叫做marker。一个进程通过向所有输出通道发送marker消息来启动快照算法，当所有的输入通道都收到marker消息时，算法结束。其思想是，在按序发送并且没有丢失的消息队列中插入marker消息，这个消息起到一个界标

的作用，marker消息在进程间传递，把属于快照描述的状态的消息和快照所描述的状态以后发送的消息分割开来。这样，就可以得到一个全局一致的状态。

3. 分步进程互斥

(1) 分布式系统中事件的定序

大多数互斥和死锁的分布式算法的基本操作是对事件进行定序。用一个先于关系“→”来确定事件的先后次序，X→Y 表示 X 先于 Y 发生。先于关系定义如下：

① 事件X、Y发生在同一进程，如果X在Y之前发生，则有X→Y；

② 如果存在消息m，则有send（m）→receive（m）；

③ 如果有X、Y、Z，且X→Y、Y→Z，那么有X→Z。

在分布式操作系统中用时间戳来实现先于关系。时间戳把系统中的所有事件映射到一个整数集合上，且满足：如果 X→Y，则 T（X）< T（Y），其中 T（X）表示事件 X 的时间戳。

(2) 互斥算法

互斥方案有 Lamport 算法、Ricart 算法和令牌算法，Ricart 算法是在 Lamport 算法基础上提出的一个分布式算法，它们是通过把进程进入临界段的顺序，按时戳的先后排序的方式来解决互斥问题。令牌算法则是通过在进程中传递一个令牌，持有令牌者才可进入临界段的方式来解决互斥问题。

许多分布式操作系统都利用一个协调者进程来执行系统中其他进程所需的功能。如果协调者所处的处理机发生故障使得协调者无法继续工作，那么需要用选择算法来产生新的协调者。在本章的“例题解答”中有选择算法的介绍。

4. 分布式死锁

与集中式操作系统一样，分布式操作系统也是只有满足死锁的四个条件，才会出现资源分配死锁。但是，分布式死锁处理遇到的困难在于“死锁幻象”(phantom deadlock)。死锁幻象是一个错误的检测，并不是真正死锁，是由缺少全局状态而引起的。在集中式操作系统中不存在这样的情况。

在分布式操作系统中同样可以通过死锁预防、死锁避免、死锁检测来解决死锁问题，但是，其中死锁避免的开销过于庞大，因此并不实用。

(1) 死锁预防

在分布式环境中，死锁预防方法中有如下两种可以用：

① 对资源类定义一个线性序列可破坏循环等待条件。

② 一个进程一次请求它所需要的全部资源，通过阻塞该进程直到所有请求同时被满足为止，这可破坏占用并等待的条件。

然而，进程并不总是可能事先就知道它所需要的资源，例如，在数据库的应用中，往往会动态加入一些新项。wait-die方法和wound-wait方法在数据库的应用中解决了这个问题，课本[1]中的图10.13列出了这两种方法。

(2) 死锁检测

死锁检测允许进程先获得它想要的资源，在其后再确定是否死锁。若检测到死锁，则可选一个进程，释放其资源，从而破坏死锁。其检测策略有集中式策略、分布式策略和分层策略三种，课本[1]中的表10.1详细列出了这三种策略的优缺点。

课本中还介绍了一个用于分布式数据库的分布式死锁检测算法。该算法的思想是，通过一种“等待”(wait-for）关系把事务都连接成一个图结构，事务是图中的结点，若发现一个等待环路，就破坏其中一个事务并释放其占用的所有资源，使其他事务能获得资源并继续执行下去。课本[1]中的图 10.14 列出了该算法的代码。

(3) 消息通信死锁

分布式操作系统中，除了资源分配死锁外，还有消息通信死锁。互相等待消息和消息缓冲分配不合理都会引起死锁。

① 等待消息死锁：一组进程中的每一个成员都在等待组内另一个成员的消息，若没有任何消息在传送，就会发生消息通信死锁。

② 缓冲分配不合理引起死锁，分为两种。课本[1]中的图10.17列出了这两种死锁的图解。

直接的存储转发死锁：一个包交换结点用公共缓冲池来缓冲包，若结点A的全部缓冲区都被要发给B的包占据，B处缓冲区都被要发给A的包占据，则A和B都不能接受包，因而发生死锁。将缓冲区分成大小固定的块，每块对应一条链，就可以预防死锁。

间接的存储转发死锁：每个结点，在一个方向上的队列都是满的，并且还有不是传给下一结点的包。可以通过结构化的缓冲池来预防这种死锁，缓冲池分层为：第0层缓冲池没有限制，任何输入包都可以存放进去；第1层到第N层缓冲池(N是网络路径的最大跳数)，按下述方法预留——第k层缓冲池留给跳数至少为k的包。这样在负载较重时，缓冲池按从第0层到第N层的顺序逐渐装满。如果从第0层到第k层的缓冲池都被装满了，则跳数小于或等于k的包被丢掉。这种方法同样可以预防直接的存储转发死锁。

10.2　例 题 解 答

例 10-1　通过以下两点证明分布式快照算法的正确性。

(1) 算法不会记录不一致的全局状态(提示：反证法)。

(2) 算法正确记录了每个通道的状态，也就是说，对于每个通道，如从进程 r 到进程 p 的通道，通道状态(r，p)就是由 r 发送的消息序列，它可达到 Sr，但 p 不一定接收到 Sp (提示：假设消息按其发送的顺序接收)。

解 (1)反证法：假设此算法记录了一个不一致的全局状态，即至少存在一个消息 m；假设是进程 p 发给进程 q 的，既没有被接收又没有在通道中；又假设消息按其发送的顺序接收，那么一个进程在收到 marker 消息时，在 marker 前被发送的消息都应该被接收，除非消息丢失。自 q 应该从（p→q）通道收 m 消息的时刻起，q 从 p 收到 marker 消息分为两种不同的情况，第一种情况是，q 第一次收到 marker 消息，在这种情况下，消息 m 应该是被 q 接收，记录在 q 的状态中，若 q 没有收到消息 m，则表示该消息丢失，这与算法的假设相矛盾。第二种情况是，q 已经开始记录状态，那么消息 m 应该被记录在通道中，消息 m 不在通道中表示消息丢失，矛盾。所以，此算法不可能记录一个不一致的全局状态。

(2) 从第一个问题的讨论中，可以清楚地看到，对于每个通道都有上述的两种情况，只要消息没有丢失，每个通道的状态都会被正确地记录。

例 10-2 在一个分布式操作系统中，如果事件 X、Y 的时间戳满足 $T(X)<T(Y)$，则 X→Y，根据该原则，所有的事件都有了确定的次序。这种说法对吗？

解 该种说法是错的。比如有两个进程 P_1 和 P_2，由于它们开始的事件 X、Y 均为非接收消息事件，所以这两个事件的时间戳一样，但是，我们无法确定这两个事件的次序。

例 10-3 在一个分布式操作系统中，进程 P_1 和 P_2 的事件 e_1 和 e_2 均有相同的时间戳，且希望进入临界段，试设计一种策略，使系统按一定次序来满足这两个事件的请求。

解 我们可以定义，对于系统中任何两个事件 X 和 Y，设它们分别是进程 P_1 和 P_2 的事件，

$$X \rightarrow Y$$

当且仅当 $(T(X)< T(Y)) \vee ((T(X)= T(Y)) \wedge (P_1 << P_2))$。其中，“ << ”是人为确定的一个进程顺序。

例 10-4 试叙述解决同步问题的 Lampot 算法。

解 设系统中有 n 个进程，

P_i 是希望进入临界段的进程，

request（T_i，P_i）表示进程 P_i 请求进入临界段的事件，其时间戳为 T_i，

reply（T_j，P_j）表示进程 P_j 的回复消息事件，其时间戳为 T_j，

release（T_i，P_i）表示进程 P_i 释放临界段事件，其时间戳为 T_i，

RQ_i 表示进程 P_i 的消息队列。

其算法描述如下。

(1) 进程 P_i 将 request（T_i，P_i）发送给其他进程，同时将 request（T_i，P_i）存入自己的 RQ_i。

(2) 进程 P_j 收到请求消息后，发送 reply（T_i，P_i）给 P_i，同时将 request（T_i, P_i）存入自己的 RQ_j。

(3) P_i 等待，直到满足以下两个条件才进入临界段：

- request（T_i，P_i）位于 RQ_i 的队首；
- P_i 已经收到 n–1 条时间戳大于 T_i 的 reply 消息。

(4) 当 P_i 退出临界段时，将 release（T_i'，P_i）发送给其他进程，同时将 request（T_i，P_i）从自己的 RQ_i 中去掉。

(5) 进程 P_j 收到 release（T_i'，P_i）后，将 request（T_i，P_i）从 RQ_j 中去掉。

例 10-5　如图 10.1 所示，考虑这样一个时间点：站点 2 收到消息 a 但未收到消息 q，站点 3 收到消息 q 但未收到消息 a，于是各站点不一致。讨论 Lamport 算法是如何解决这个问题的，它是否会引起本章讨论的其他互斥算法的困难？

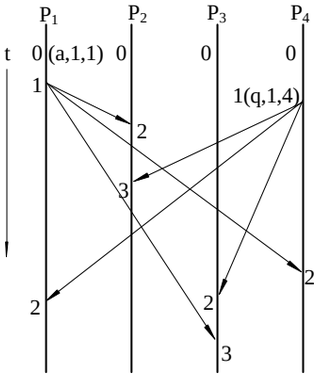


图 10.1　时间戳算法运行举例

解　Lamport 算法为了解决这个问题，就增加了一个规则：如果一个进程要根据它的队列进行分配，就需要从所有其他站点收到一个消息，以确保没有比它自己队列头更早的消息还在传送中。Lamport 算法的第二步实现了这个规则：进程 P_j 收到请求消息后，发送 reply（T_i，P_i）给 P_i，同时将 request（T_i, P_i）存入自己的 RQ_j。一旦 P_i 要进入临界段，系统中就不可能有其他的请求在其前传送，因为那时 P_i 已经从其他所有站点收到了一个消息，这些消息比它的请求消息要晚，reply 机制保证了这一点。

另外一个解决互斥的算法是：Ricart 算法，这个不一致问题不会对 Ricart 算法产生任何影响，因为 Ricart 算法不要求进程按消息的发送顺序接收，它是用发送请求给所有的进程，并等待接收所有的回应的方法解决互斥问题的，所以消息到达顺序的不一致不会使 Ricart 算法做出任何错误的判断。

例 10-6　试叙述解决同步问题的 Ricart 算法。

解　其算法如下。

该进程有一个数组q并遵循以下规则。

(1) 当P_i请求访问资源时，它发出一个请求(Request，T_i，i)。时戳为当前本地

时钟之值。它将这条消息放入自身数组q[i]中，然后，将消息发送给所有其他进程。

(2) 当P_j收到请求(Request，T_i，i)后，按如下规则处理：

·如果P_j正处于临界段中，就延迟发送Reply允许的消息(见规则4)；

·如果P_j并不等待进入临界段，就发送(Reply，T_j，j)给所有其他进程；

·如果P_j等待进入其临界段，并且到来的消息在P_j的Request后，则将到来的消息放入其数组的q[i]中，并延迟发送Reply消息；

·如果P_j等待进入其临界段，但是到来的消息在P_j的Request前，就将到来的消息放入其数组的q[i]中，并发送(Reply，T_j，j)给P_i。

(3) 如果P_i从所有其他进程都收到了Reply允许的消息，它就可以访问资源(进入临界段)。

(4) 当P_i离开临界段时，它给每个挂起的Request发送一个Reply允许的消息，从而释放资源。

例 10-7　对于 Ricart 算法，

(1) 证明其实现了互斥。

(2) 如果消息没按其发送顺序到达，则算法不保证按它们请求的顺序执行临界段。这可能引起饥饿吗？

解　(1) 假设某一个进程在临界段中，根据 Ricart 算法的描述，其他任何一个进程都不可能得到此进程的 reply 消息，因而也无法获得全部 n–1 个消息，所以任一时刻只能有一个进程进入临界段。

(2) 不可能，一旦请求的消息到达，由于这个消息的时间戳最早，它总会进入临界段执行，所以不会产生饥饿。

例 10-8　试叙述基于环结构的令牌解决互斥的方案。

解　令牌在环中逆时针循环。当进程收到左邻站点发来的令牌时，如果它不希望进入临界段，就直接将令牌传给右邻站点，否则就保存令牌，进入临界段，在退出临界段后，将令牌传给其右邻站点。程序实现如下：

```
if not token-present then begin clock:=clock+1;          [Prelude]
                                broadcast(request, clock, i);
                                wait(access, token);
                                token-present:=True
                           end;
end if;
token-held:=True;
<critical section>
token(i):=clock;                                         [Postlude]
```

```
token-held:=false;
for j:=i+1 to n, 1 to i–1 do
  if (request(j)＞token(j))∧token-present
    then begin
      token-present:=False;
      send(j, access, token)
    end
  end if;
```

(a)

```
When received(request, t, j) do
  request (j):=max (request(j), t);
  if token-present∧not token-held then
  <text of postlude>
  end if
end do;
```

(b)

其中，使用的记号说明如下：

send(j, access, token) 将access消息和令牌token传给进程j；

broadcast(request, clock, i) 将进程i的请求消息和时间戳clock传给所有其他进程；

received(request, t, j) 从进程j收到请求消息及时间戳t。

例 10-9　在令牌传递互斥算法中，时戳是否用来重设时钟和修正偏离？如果不是，时间戳的作用是什么？

解　时间戳的设置不是用来重设时钟和修正偏离的。时间戳是用来对提出的请求标一个先后的顺序，这个顺序并不是完全和客观时间顺序一致，但它是一个在所有站点都相同的顺序。

例 10-10　对于令牌传送互斥算法，证明：

(1) **保证了互斥；**

(2) **避免了死锁；**

(3) **公平。**

解　(1) 在此算法中，只有想进入临界段并且得到令牌的站点才能进入临界段，所以这个算法能保证互斥。

(2) **令牌是按照一个方向传送下去的，也就是站点进入临界段的顺序是一定的，是一个轮转顺序，所以它不会死锁。**

(3) **由于它采用轮转策略，严格按照轮转方向和顺序执行，所以每个站点的**

机会相等，即此算法是公平的。

例 10-11　在令牌方法解决互斥的程序实现中，解释为什么第 2 行不能简单地写成“request(j):=t”。

解　由于消息传递的延迟，可能出现晚发出的 request 消息先到达的情况，所以不能直接写成“request(j):=t”。

例 10-12　比较基于死锁预防的等死方法和因伤等死方法。

解　等死方法是基于非抢占技术的，当进程 P_i 申请当前已由 P_j 占用的资源时，如果 P_i 的时间戳小于 P_j 的时间戳（即 P_i 比 P_j 年长），那么 P_i 就等待，否则 P_i 撤离。

因伤等死方法是基于抢占技术的，当进程 P_i 申请当前已由 P_j 占用的资源时，如果 P_i 的时间戳大于 P_j 的时间戳（即 P_i 比 P_j 年轻），那么 P_i 就等待，否则 P_j 就撤离。

在等死方案中，年长的进程必须等待年轻的进程释放它占用的资源，因此，进程越老，等待的时间可能越长。在因伤等死方案中，年长的进程决不会等待年轻的进程。

在等死方案中，如果进程 P_i 由于申请已由 P_j 占用的资源而撤离，那么，当 P_i 重新开始时，它可能再次发出相同的申请。若该资源仍被 P_j 占用，则 P_i 将再次撤离，因此，在获得所需资源之前，P_i 可能撤离多次。在因伤等死方案中，如果进程 P_i 由于 P_j 申请一个已由它占用的资源而被撤离，那么，当 P_i 重新开始并申请现在由 P_j 占用的同一资源时，P_i 就等待，因此，在因伤等死方法中只存在较少的撤离。

这两个方法的主要不足是可能发生某些不必要的撤离。

例 10-13　分布式互斥的 6 个要求之一是进程在非临界段停止时不影响其他进程，假设有另外一个进程发送消息给已停止进程，并等待回答，问：这种死锁可避免吗？

解　这种死锁是消息死锁，和资源死锁一样，消息死锁也可以预防和检测。只要发送消息的进程不在临界段中等待其他停止进程的消息，就不会影响其他进程的互斥执行，因此当临界段中含有消息资源的使用时，就要注意消息死锁的问题。

例 10-14　在死锁检测中，如何消除假环路？

解　可以通过时间戳来消除假环路，给来自不同场地的请求消息附上唯一的时间戳。当站点 A 上的进程 P_i 申请站点 B 的进程 P_j 占用的资源时，发送一条带有时间戳的申请消息 insert（P_i，P_j，T）；当站点 A 上的进程 P_i 获得站点 B 的进程 P_j 占用的资源时，发送一条带有时间戳的删除消息 delete（P_i，P_j，T），在构造等待图时，根据时间戳丢弃过时消息。

10.3　自 测 练 习

一、填空题

1. 实现资源共享的方法主要有_____、_____和_____。

2. 假设分布式操作系统有一个逻辑时钟，如果 P_1 的时钟在它向 P_2 发送消息时为 12，进程 P_2 收到来自 P_1 的消息时的时钟为 8，则下一局部事件 P_2 的逻辑时钟为_____；如果接收时 P_2 逻辑时钟为 15，那么下一局部事件 P_2 的逻辑时钟为_____。

3. 设分布式操作系统中有 n 个进程，则 Lampot 方案的通信量为_____条消息，其中_____条 Request 消息，_____条 Reply 消息，_____条 Release 消息。

二、判断题（正确的在括号中记√，错误的记×）

1. 在分布式操作系统中，任一进程的第 1 个事件的时间戳为 1。　　(　　)

2. 先于关系具有传递性。　　(　　)

三、选择题

1. 在一个分布式操作系统中，有进程 P_1 和 P_2，P_1 的事件 e_2 的时间戳为 4，P_1 的事件 e_3 发送一个消息 m 给 P_2；P_2 的事件 e_2 的时间戳为 7，P_2 的事件 e_3 接收消息 m，e_3 的时间戳为多少？　　(　　)

a) 5；　　b) 6；　　c) 7；　　d) 8。

2. 在一个分布式操作系统中，有进程 P_1 和 P_2，它们的事件如图 10.2 所示。观察该图回答：以下说法正确的是哪些？　　(　　)

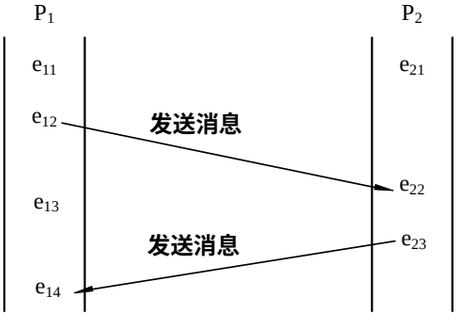


图 10.2

a) $e_{11}\rightarrow e_{21}$ ； b) $e_{21}\rightarrow e_{13}$ ； c) $e_{22}\rightarrow e_{14}$ ； d) $e_{11}\rightarrow e_{23}$ 。

3. **设分布式系统中有** n **个进程，问：**Ricart **方案的通信量为多少？** ( )

a) n–1； b) 2(n–1)； c) 3(n–1)； d) 4(n–1)。

4. Lamport **算法和** Ricart **算法的共性是什么？** ( )

a) **进程彼此需要知道系统中所有其他进程的标识符；**

b) **通信量为** 2(n–1)；

c) **如果系统中的某个进程故障，那么整个算法就无法工作；**

d) **无饥饿现象发生。**

5. **因伤等死方法破坏了死锁四个必要条件中的哪些条件？** ( )

a) **互斥；** b) **占有并等待；** c) **非抢占；** d) **循环等待。**

四、问答题

1. **试述分布式操作系统中，如何用时间戳来实现事件定序。**
2. **试叙述基于非环结构的用令牌解决互斥的策略。**
3. **在基于等待图的死锁检测中，为什么要使用时间戳？**
4. **试叙述层次式死锁检测方案。**

10.4 自测练习答案

一、填空题

1. **实现资源共享的方法主要有** 数据迁移 **、** 计算迁移 **和** 作业迁移 **。**

2. **假设分布式操作系统有一个逻辑时钟，如果进程** P_1 **的时钟在它向** P_2 **发送消息时为** 12**，进程** P_2 **收到来自** P_1 **的消息时时钟为** 8**，则下一局部事件** P_2 **的逻辑时钟为** 13 **；如果接收时** P_2 **逻辑时钟为** 15**，那么下一局部事件** P_2 **的逻辑时钟为** 16 **。**

3. **设分布式操作系统中有** n **个进程，则** Lampot **方案的通信量为** 3(n–1) **条消息，其中** n–1 **条** Request **消息，** n–1 **条** Reply **消息，** n–1 **条** Release **消息。**

二、判断题

1. ×　　2. √

三、选择题

1. d）　　2. c）、d）
3. b）　　4. a）、c）、d）
5. c）

四、问答题

1. **见知识要点。**

2. **假定有 n 个站点，每个站点有一个进程 P_1，P_2，…，P_n；有一个与令牌相关的向量 T=（T_1，T_2，…，T_n），T_i 记录进程 P_i 已经进入临界段的次数。其流程如图 10.3 所示。**

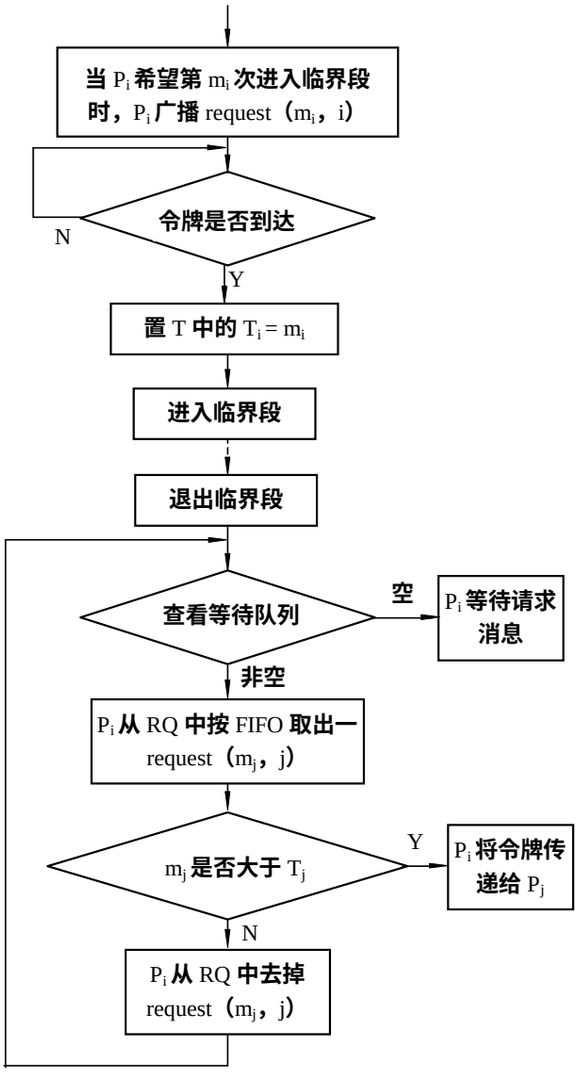


图 10.3

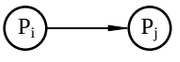
3. **在基于等待图的死锁检测中，可能会出现假环路。考虑如下情况，P_1 申**

请进程 P_2 在站点 A 上占用的资源，P_3 申请进程 P_1 在站点 B 上占用的资源。假定 P_2 释放了它在站点 A 上占用的资源后，导致要删除站点 A 中的边（P_1，P_2）。然后，P_2 申请进程 P_3 在站点 B 上占用的资源，导致在站点 B 上要增加边（P_2，P_3）。这两个更改要传给协调者。如果来自站点 B 的增加边的消息在来自站点 A 的删除边的消息之前到达协调者，那么当协调者执行了增加边操作而未执行删除边操作时，协调者将发现新增加的环路{P_1，P_2，P_3}，需要撤离其中一个进程。通过增加时间戳，丢弃过时的信息，协调者就不会发现新环路，从而避免不必要的进程撤离。

4. 层次式的死锁检测将全局进程等待图的管理分散，由位于不同层次的管理者管理。

(1) 最低层的管理者管理各自站点上的局部进程等待图。

(2) 如果某些管理者的局部进程等待中有公共站点，则在这些管理者之上构造新的一层管理者，将这些公共站点作为新一层管理者，并将这些公共站点作为新一层管理者管理的进程等待图中的站点。若站点 P_i、P_j 出现在新进程等待图中，且在下层之一的进程等待图中存在从 P_i 到 P_j 的路径，则将



插入到新的进程等待图中。

(3) 重复以上过程，直至无新层产生为止。

第 11 章

操作系统的安全性

11.1 重点与难点

11.1.1 概述

安全是一个比保护更宽广的课题，保护是采用某种机制以控制程序、进程或用户对被计算机系统所定义的资源的存取。安全是指被保护的系统及其数据的完整性、可用性、正确性的程度。

计算机系统包含许多对象，这些对象需要保护以防止滥用。对象可能是存储器、CPU 等硬件，或文件、程序等软件。存取权是在一个对象上执行一个操作的许可权。域是一组存取权。进程在域中执行，并且可以使用该域中的任何存取权来存取和操作对象。

存取矩阵是保护的基本模型。存取矩阵为保护提供了一种机制，而不用对系统或它的用户强迫实施特殊的保护策略。策略和机制的分离是一重要的设计特征。

存取矩阵是稀疏的，它通常的实现方法是，或者作为与每个对象相关联的存取表，或者作为与每个域相关联的权能表。动态保护可包括在存取矩阵模型中。

实际的系统提供的保护是非常有限的，并且趋向于只为文件提供保护。如 Unix，它分别对每个文件的拥有者、组用户和公共用户提供读、写、执行保护。Multics 系统将一个环结构加于文件存取上来实施保护。Hydra 系统是将保护延伸到用户定义软件对象上的权能系统。

保护是一个内部问题，安全性必须考虑计算机系统和系统应用的环境，口令通常用来解决鉴定问题。

通过本章学习，要求掌握安全性、系统安全、安全级别等相关概念；熟知系统的各种安全保护机制和各自的特点，能理解应用其中一些机制；了解病毒及其

防御的知识；了解两种重要的加密技术；理解、识记安全模型的概念，认识到安全模型的重要性；了解几种系统的安全实现。

11.1.2 知识要点

1. 安全性概述

(1) 安全性的内容

系统安全的内涵在一直不停地演变，一般来讲，系统的安全性包括以下几方面的内容：

① 保护系统内的各种资源免遭自然与人为的破坏；

② 估计到操作系统存在的各种不安全因素，以及它存在的特殊问题；

③ 开发与实施卓有成效的安全策略，尽可能减少系统所面临的各种风险；

④ 准备适当的应急措施，使系统在遭到破坏或攻击时能尽快恢复正常；

⑤ 定期检查各种安全管理措施的实施情况。

对系统安全的要求因系统不同而有所出入，但总的来讲系统应该有：

① 保密性。计算机系统的资源仅能由授权用户存取。

② 完整性。分为软件完整性和数据完整性，防止非法用户对软件和数据的蓄意修改或防止意外事件引起的破坏。

③ 可用性。系统内的各种资源应随时可供授权用户使用。

对计算机安全的威胁通常来自于：偶然无意的破坏、自然灾害和人为攻击等几个方面，操作系统安全是系统安全的关键，一个安全的操作系统应该具有用户身份鉴别、内存保护、文件及 I/O 设备存取控制、实体控制与保护、共享约束和用户平等问题等功能。

(2) 安全系统的设计原则和安全等级

安全操作系统的设计原则有：最小权限、经济性、开放式设计、策划完整、权限分离、最少通用机制等。在设计一个安全操作系统时要从物理分离、时间分离、逻辑分离和密码分离几个方面考虑。

美国国防部把计算机系统的安全从低到高分为 4 等(D 最低保护等级、C 自主保护等级、B 强制保护等级、A 验证保护等级)和 8 级(D1、C1、C3、B1、B2、B3、A1、A2)，从最低级(D1)开始，随着级别的提高，系统的可信度也随之增加，风险也逐渐减少，但是系统设计的复杂度也会随之迅速增长。

2. 安全保护机制

系统安全保护机制主要从操作系统内部的进程支持、内存及地址保护、存取

控制、文件保护和用户身份鉴别等方面进行实施。

(1) 进程支持

现代操作系统都支持一个进程代理一个用户的概念，系统将通过 PCB 而感知相应的进程，进程控制块 PCB 是进程存在的唯一标志，进程控制块 PCB 包含了进程的描述信息和控制信息。系统通过 PCB 隔离不同用户的进程，并借此使各个用户彼此隔离。

(2) 内存保护

多道程序中要防止一道程序在存储和运行时影响到其他程序。操作系统可以在硬件中有效使用硬保护机制来进行存储器的安全保护。现在最常用的是界址、界限寄存器、重定位、特征位、分段、分页和段页式机制，其中分段与段页式机制更有利于系统的安全保护。

(3) 存取控制

存取控制是安全机制的主要目标，它包含 3 个方面的内容：首先是授权，即确定可给予哪些主体存取实体的权力；其次是确定存取权限；最后是实施存取权限。存取控制机制实施的具体过程是检查每个存取动作，拒绝超越存取权限的行为，并防止撤权后对实体的再次存取；实施最小权限原则，不能对额外信息进行存取；存取验证除了检查是否存取外，还应检查在实体上所进行的活动是否适当。

(4) 文件保护

文件系统是操作系统的又一个重要部分，对文件系统的安全保护机制分为对文件系统本身的安全保护和对文件存储载体的安全保护。根据其实施的形式不同，分为基本保护、全部/无保护、组保护、单许可保护和单实体及单用户保护。

(5) 身份验证

操作系统的许多保护措施都是基于鉴别系统中的合法用户的，身份鉴别是操作系统中很重要的一个方面，也是用户获得权限的关键。大多数系统使用口令鉴定用户身份，也有采取其他方法如笔迹鉴定、声音识别、视网膜扫描等来鉴定用户身份的。

3. 病毒及其防御

(1) 病毒的概念

计算机病毒(简称病毒)是一种可传染其他程序的程序，它通过修改其他程序，使之成为含有病毒的版本或可能的演化版本、变种或其他繁衍体。病毒具有一些显著的特点：它是一段可执行程序，具有依附性，依附在宿主上；具有传染性、潜伏性和破坏性；病毒的发作还要一定的环境，一种病毒并不是在所有的系统中都能够传染，因此还具有针对性。

(2) 病毒的结构及分类

病毒大多由三部分组成：引导模块、传染模块和表现模块。其中引导模块负责将病毒引导到内存，对相应的存储空间实施保护，以防止被其他程序覆盖，并且修改一些必要的系统参数，为激活病毒做准备；传染模块负责将病毒传染给其他计算机程序；表现模块是病毒的具体表现部分。

根据病毒的特征，有很多种对病毒分类的方式，可以按照其攻击的对象（不同的软硬件系统）分类；可以按照攻击的机种分类；可以按链接方式分类，如操作系统病毒，源码病毒，入侵型病毒和外壳型病毒等；还可以按寄生方式分类，如引导区病毒和可执行文件病毒等。

(3) 病毒的预防

病毒具有针对性，不同的操作系统或不兼容的系统的病毒一般不会相互传染，但是，也有能感染不同系统的病毒，例如，有的病毒就可以同时感染 Linux 系统和 Windows 系统。病毒防御措施通常与系统的存取控制、实体保护等安全机制配合起来，再由专门的防御程序模块来完成。防御的重点在操作系统敏感的数据结构、文件系统、数据存储结构和 I/O 设备驱动结构上。针对病毒的各种攻击，病毒防御机制采用了存储映像、数据备份、修改许可、区域保护、动态检疫等方式。

在众多的病毒中，尤其值得一提的是特洛伊木马病毒，它是一段程序，表面上在进行合法操作，实际上却进行非法操作，受骗者是程序的用户，入侵者是这段程序的开发者。它攻击系统的一个关键标志是，通过一个合法的信息信道进行非法通信，信息就是通过这些隐蔽通道被泄漏出去的。

4. 加密技术

加密就是将可理解的消息变成不可理解的消息，从而隐藏其真正含义的过程。与加密技术针锋相对的技术是解密技术，破解密文的技术分为两种：一种是对加密算法所依赖的数学基础进行分析，然后对其攻击；另一种是用所有可能的密钥尝试，即所谓的穷举算法。

(1) 传统加密算法

传统的加密算法是将明文通过加密算法变成密文，在接受端则要通过解密密钥还原明文，这种加密过程不保密的加密算法，只对密钥进行保密。因此，传统加密算法的实现耗费较低，但是，密钥的安全管理成了传统加密的主要安全性问题。美国国家标准局(NBS)在 1977 年研制的数据加密标准 DES，就是一种传统加密算法。

(2) 公开密钥算法

在传统加密方法中，加密和解密所用的密钥是相同的，即对称加密算法；但是，这并非必要的，也可以设计出一个算法用一个密钥加密，用另一个不同的但

有联系的密钥来解密，即非对称加密算法。公开密钥算法就是其中的一种，在网中每个端系统都产生一对密钥，并且都将加密密钥放在公共寄存器或文件中(也就是公开密钥)，解密密钥设为私有。传输消息时用公开密钥加密，接收消息的一端用私有密钥解密。

(3) 密钥的管理

密钥的管理是加密系统中很重要的一环，加密系统的保密强度与密钥的分配有很大关系。传统加密算法中的两个实体具有相同的密钥，如何在他们之间传输密钥是关键，密钥分配方法有四种：由 A 来选择密钥，传输给 B；由第三者选择密钥，传输给 A 和 B；如果 A 和 B 先后用了同一个密钥，则由一个主体将新密钥传输给另一个；如果 A 和 B 都与 C 有加密链接，则 C 可在加密链上向 A 和 B 传输密钥。在最后一种方法中，A、B 间通信用一个暂时的密钥，通信结束后密钥作废；A、B 与 C 之间的加密链接则用的是永久密钥，通常使用公开密钥算法，但此算法耗费大不适合做数据传输加密。

5. 安全操作系统的设计

(1) 安全模型

安全模型用来描述计算机系统和用户的安全特性，是一种系统安全需求的抽象描述。其安全要求有 3 个：保密性、完整性和可用性。

在单层模型中，存取策略为每个用户和实体简单地指定权限，即用户将对实体的存取策略简单地设置为“允许”或“禁止”。此种模型又有两种不同的表达方法，第一种是监督程序模型，用户通过监督程序来访问实体；第二种是信息流模型，用一个类似智能筛的程序来过滤传输请求。

多层网络模型是另一种安全模型，它通过〈级别；隔离组〉结构来对实体分类，并引入关系“≤”，对于实体 O 和主体 S 有 O≤S 的关系，当且仅当级别 O≤级别 S，并且隔离组 O∈隔离组 S。此关系具有传递性和非对称性，仅当实体和用户满足此关系时，访问才是合法的。

(2) 安全操作系统的设计与实施

安全操作系统的设计开发要经历三个阶段：第一是模型化阶段，构造一个需保密的模型并研究达到安全性的不同方法；第二是设计阶段，选择一种实现该模型的方法，并保证设计者与用户确信设计准确地表达了模型的意图，而代码准确地表达了设计思想；第三是实现阶段，有两种方法，一种是专门针对安全性而设计的操作系统，另一种是将安全的特性加到目前的操作系统之中。

操作系统的设计是非常复杂的，安全因素的考虑增加了系统设计的难度。安全功能遍及整个操作系统的设计和结构中，一方面，必须在操作系统设计的各个方面考虑安全性，在设计了一个部分之后，必须检查它所强制或提供的安全性尺

度；另一方面，安全性必须成为操作系统初始设计的一部分。这也是进行操作系统安全设计时必须遵循的原则。

实施这些原则的方法有：隔离(最少通用机制的逻辑扩展)，内核化的设计(最少权限及机制经济性方面的考虑)，分层结构(开放式设计及完整的策划)等。

11.1.3 系统实例

有关 Windows 2000 操作系统在安全方面的信息可以参考课本[1]中 11.6 节的内容。

11.2 例 题 解 答

例 11-1 解释系统安全保护中的名词“域”和“保护域”。

解 域是存取权的一个集合，每一个存取权都是一个有序对（对象名，权集）。进程在保护域内操作，该域指明了进程能存取的资源，每个域定义了一组对象和每个对象可以涉及的操作类型。进程域之间的关联可以是静态的，也可以是动态的。如果关联是动态的，则可用某种机制来允许一个进程从一个域转到另一个域。

例 11-2 什么是存取矩阵，它是如何实现保护机制的？

解 存取矩阵由一个有序的三维表项<域，对象，权集>的集合组成。存取矩阵的行代表域，列代表对象，矩阵中的每一项含有一组存取权。存取矩阵主要通过全局表、存取表和权能表来实现。最简单的实现是全局表，在域 D_i 上对对象 O_j 执行操作 M 时，就查寻全局表，以搜索三维表项<D_i，O_j，R_k>，如果存在这样的三维表项且 $M \in R_k$，则操作允许执行。在存取表实现方法中，存取矩阵中的每列都作为一个对象的存取表来实现，每一对象的结果表就由有序对<域，权集>组成，这些有序对定义了有关该对象的具有一个非空存取权集的全部域。在权能表实现方法中，存取矩阵的每行都与域相关。关于一个域的权能表就是由一组对象及在这组对象上所允许的一组操作组成的表。一个对象常用它的物理名字或地址表示，叫做一个权能。

例 11-3 解释需知原理的含义。

解 需知原理就是指一用户可以在任何时候存取那些已被授权存取的资源，而且这些资源对于完成其任务是需要的。

例 11-4 一个操作系统可以提供哪几个层次的保护？

解 一个操作系统主要可以提供 6 个层次的保护：

(1) 无保护。适合于敏感过程运行于独立的时间的场合。

(2) 隔离。这种方法意味着并发运行的不同进程不会感觉到相互的存在，每个进程都有自己的地址空间、文件及其他对象。

(3) 全部共享或无共享。实体的属主将实体说明为“公用”或“私用”，公用实体对所有用户开放，而私用实体只可被属主使用。

(4) 借助于存取限制的共享。操作系统检查特定用户是否可以存取特定实体，因此，操作系统在用户和实体之间充当卫兵的角色，保证只有授权存取发生。

(5) 借助于权能的共享。它是限制存取共享的扩展，它允许为实体动态建立共享权限。

(6) 限制实体使用。这种形式的保护不仅限制对实体的存取而且限制存取后的使用。

上述这些保护是按其实现难度的递增次序排列的，同样，也是按其所能提供保护的性能的递增顺序排列的。一个特定的操作系统可能为不同的实体、用户或环境提供不同层次的保护措施。

例 11-5　在一个系统中，有 4 个保护域和 6 个资源对象，则全局表、存取表和权能表的个数分别最多应为多少个？

解　全局表是一个有序的三维表<域，对象，权集>，它能保存存取矩阵的所有信息，所以，有 1 张表就可以了。存取矩阵中的每列代表一个对象，它的每列都作为一个对象的存取表来实现，所以，存取表最多有 6 个。存取矩阵中的每行代表一个域，它的每行都作为一个域的权能表来实现，所以，权能表最多有 4 个。

例 11-6　目录是应控制存取的一种实体，为什么不允许用户直接修改他人的目录？

解　为了防止伪造存取文件，系统不允许任何用户写入文件目录。例如某个用户想非法读取他人的文件，就去读取该用户的目录表，把该文件的描述表项复制到自己的目录表中并设置相应的权限，以后就可以访问这个文件了。因此，操作系统必须在文件主命令的控制下维护所有的文件目录，禁止用户直接对目录存取，但用户可以通过系统进行合理的目录操作。

例 11-7　在环形保护中，可能存在两个不同的域 D_i 和 D_j，D_i 不是 D_j 的子集，D_j 也不是 D_i 的子集。这种说法正确吗？

解　这种说法是错误的。

在环形保护中，如果 $j<i$，则 D_i 是 D_j 的一个子集；反之，如果 $i<j$，则 D_j 是 D_i 的一个子集。所以对于一个系统中的任意两个环，其中一个必定是另一个的子集。

例 11-8　假定一个计算机系统中的“计算机游戏”的开放时间和对象是：每天 10:00PM~6:00AM 供学生占用；5:00PM~8:00AM 供教师占用；计算机中心的工作人员可在任何时候用机。试提出一个有效实现这种安排的方案。

解　建立一个动态保护结构，它根据分配给三类用户的时间来改变可用资源的集合。时间改变时，有权玩“计算机游戏”的用户的域随之改变。当用户有权玩“计算机游戏”的时间结束时，必须执行一个“撤销”过程，这个撤销可以是立即的、选择性的（计算机中心的工作人员可在任何时间存取它），绝对的或暂时的（以后将存取权归还）。

例 11-9　操作系统中保存有包含所有口令的一张表。若允许用户访问这张表，那么口令就难以得到保护。试提出一个避免该问题的方案。

解　对口令进行内部译码，使它们只能以密码的形式存取，只有系统操作员才有解码口令，才能存取。

例 11-10　假设从 26 个字母表中选 4 个字母形成一个口令，一个攻击者以每秒钟一个字母的速度试探每个口令。试回答：

(1) 直到试探结束才反馈给攻击者，找到正确口令的平均时间是多少?

(2) 只要输入了不正确字母就反馈给攻击者，找到正确口令的平均时间又是多少?

解　用穷举法搜索口令空间，其期望值是搜索一半就可以搜索到口令，

(1) 口令空间有 26^4 个，平均搜索一半即 $26^4/2$ 个，每个口令 4 个字母，所以平均要 2×26^4 秒的时间。

(2) 因为只要输入错误字符就反馈给攻击者，所以可以逐个测试，每个平均试 13 次，共 4×13 次，即要耗费 52 秒的时间。

例 11-11　假定授予一进程只能对一对象存取 n 次的权能，试提出一个实现该策略的方案。

解　增设一个有关权能的整型计数器，计数器的初始值为 n，每当该进程存取该对象一次，计数器减 1，当计数器的值为 0 时，该进程就不能访问该对象了。

例 11-12　若两个用户共同存取某一段，他们必须使用相同的名字才能存取。他们的保护权限也必须一样吗，为什么？

解　他们的保护权限不必一样。只要用户或进程有足够的存取权限就可以访问相应的实体，也就是说他们的权限不必一样，只要大于某个门限值就可以了。

例 11-13　试叙述 Unix 系统、Multics 系统和 Hydra 系统中的保护方案。

解　在 Unix 系统中，保护方案基本是以文件系统为中心的，相应于拥有者、组和普通类用户，三个保护域与每个文件相关。每个域含有三个位 r、w 和 x，位 r 控制读，w 控制写，x 控制执行。Unix 中的域同用户相关。域交换相当于临时改变用户标识，这种改变通过文件系统实现如下：一个拥有者标识和一个域位跟每个文件相关，当一个用户 A 开始执行被用户 B 拥有的文件时，相应域位被打开，用户标识被置为 B，当该执行过程退出这个文件时，用户标识被重置为 A。

在 Multics 系统中，保护方案以文件系统和环结构为中心。Multics 系统中的

保护域按分层形式组织成一个环形结构，每个环相当于一个单独的域，环从 0 到 7。令 D_i 和 D_j 是任意两个域环，如果 j <i，那么 D_i 是 D_j 的一个子集，即在域 D_j 中执行进程比在 D_i 中执行的进程更有特权。在域 D_0 中执行的进程是最有特权的。Multics 有分段式的地址空间，每个段是一个文件。每个段与一个环相关联，一个段说明包含一个标识该环号的项，另外，它还包括三个存取位，分别控制读、写和执行。当前环号计数器与每个进程关联，以标识进程当前正执行的那个环，当一个进程在环 i 上执行时，若 j<i，则不能存取与环 j 有关联的段；若 k≥i，则能存取与环 k 有关联的段。

Hydra 系统是一个基于权能的保护系统，它提供了可能存取权的一个固定集，该集为系统所知并由它解释。这些权包括如读、写和执行一个内存段这样一些存取的基本形式。另外，系统还为用户提供了说明附加权的手段。用户定义的权的解释由用户程序唯一执行，但系统为这些权的使用以及系统定义的权的使用提供了存取保护。当一个对象的定义被介绍给 Hydra 时，在那个类型上的操作名就变成辅助权，在权能中也能对类型的一个例示描述辅助权。为了让进程在类型对象上执行操作，对那个对象所握有的权能必须包含在辅助权之中所涉及的那个操作的名字。一个 Hydra 子系统被建在保护核的顶上且可以请求保护它自己的成分，子系统与内核的交互作用是通过内核定义的一组原语进行的，这些原语定义了由该子系统所定义的资源的存取权。用户进程使用这些资源的策略能由该子系统设计者定义，但可通过使用由权能系统提供的标准存取保护而强行实施。

例 11-14　给出一些流量分析可以破坏安全性的例子，描述端到端加密与链路加密结合时仍有可能进行流量分析的情形。

解　流量分析技术是通过观察网络中传输的消息流的格式、长短、目的地等属性来推测得到一些有用的信息技术，如目的机的所在地，连接主机的身份等。所以说即使信息被加密，攻击者即使不能直接看到信息的内容，也可以用这种方法得到一些有用的信息。例如，两个公司间在某一时间有商业联系，那么在这一段时间里，两个公司必然会交换大量的商业信息，攻击者可能无法解密这些加密信息，但可以通过分析得知这两个公司可能有商业联系。

11.3　自 测 练 习

一、填空题

1. Hydra 是一个基于_____表的保护系统。

2. 鉴认一个用户标识最常用的方法是使用用户_____。

3. 一个计算机系统的基本特征是_____、_____和_____。

4. 安全操作系统应该具有_____、_____、_____、_____、_____和_____等功能。

二、判断题（正确的在括号中记√，错误的记×）

1. 在一个系统中所定义的保护域是两两不相交的。 ( )

2. 存取表比全局表节省空间。 ( )

3. 在环形保护中，可能存在两个不同的域 D_i 和 D_j，D_i 和 D_j 的交集为空。 ( )

4. 安全性和保护是同一个意思。 ( )

三、选择题

1. 保护域和存取权的关系是什么？ ( )

a) 存取权是保护域的一个元素；　　b) 存取权是保护域的子集；

c) 保护域和存取域有交集；　　d) 保护域和存取域交集为空。

2. 在环形保护中，第 0 层对象有最大的存取权。如果把这种环形结构中某一特定层上的一个程序的存取权视为一个权能集，设 $j>i$，那么对一个对象而言，第 j 层上域的权能与第 i 层上域的权能之间的关系是什么？ ( )

a) D_i 是 D_j 的子集；　　b) D_j 是 D_i 的子集；

c) D_i 和 D_j 交集非空；　　d) D_i 和 D_j 交集为空。

3. 某个计算机系统定义了一棵进程树，该进程树的根是可做任何事情的操作系统，一个进程的所有后代是给定的资源和仅由其祖先可存取的存取权，这样的后代决不可能去做其祖先不能做的任何事情。假定用存取矩阵 A 表示存取权的集合，A（x,y）定义了进程 x 访问对象 y 的存取权，若 x 是 z 的后代，那么，对于任意对象 y，下列哪一个是正确的？ ( )

a) A（z,y）是 A（x,y）的子集；

b) A（x,y）是 A（z,y）的子集；

c) A（z,y）和 A（x,y）没有包含关系；

d) 以上三种关系都有可能。

四、问答题

1. 有三个保护域 D_1、D_2 和 D_3，域用矩形框表示，如图 11.1 所示，存取权用有序对<对象名，权集>表示，其存取矩阵是怎样的？

2. 存取表和权能表之间的主要区别是什么？

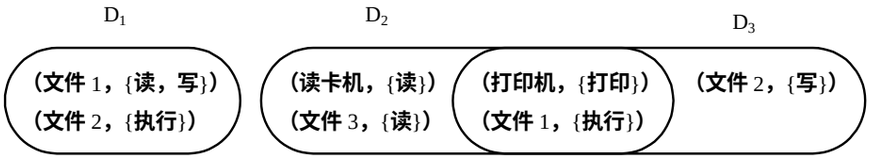


图 11.1

3. 阐述计算机病毒运行机理。

4. 试叙述计算机操作系统的安全模型。

5. 有三个基于动态保护结构的保护域 D_1、D_2 和 D_3，域用矩形框表示，如图 11.1 所示，存取权用有序对<对象名，权集>表示，允许保护域 D_1 转向保护域 D_3、保护域 D_2 转向保护域 D_1，其存取矩阵是怎样的？

11.4　自测练习答案

一、填空题

1. Hydra 是一个基于 权能 表的保护系统。

2. 鉴认一个用户标识最常用的方法是使用用户 口令 。

3. 一个计算机系统的基本特征是保密性，完整性和可用性。

4. 安全操作系统应该具有用户身份鉴别、内存保护、文件及 I/O 设备存取控制、实体控制与保护、共享约束和用户平等服务等功能。

二、判断题

1. ×　　2. √　　3. ×　　4. ×

三、选择题

1. a)　　2. b)、c)　　3. b)

四、问答题

1. 存取矩阵如表 11.1 所示。

表 11.1

域 \ 对象	文件 1	文件 2	文件 3	读卡机	打印机
D_1	读，写	执行			
D_2	执行		读	读	打印
D_3	执行	写			打印

2. **存取表将存取矩阵的每一列作为一个对象，每一个对象由具有该对象存取权的非空集的域组成；而权能表是关于每一域上的对象和在这对象上所允许的操作的表列。**

3. **病毒的运行机理首先是，检查是否要感染计算机，如要，就对计算机进行感染，然后，再去执行下面的命令。如不需要感染，就跳过感染过程去执行下面的指令，也就是去判断触发条件是否满足，如果不满足，就去执行正常的系统命令，如果满足，病毒的表现模块就会运行起来，产生一系列的症状。**

4. **略（见前面知识点）**

5. **存取矩阵如表** 11.2 **所示。**

表 11.2

对象 / 域	**文件** 1	**文件** 2	**文件** 3	**读卡机**	**打印机**	D_1	D_2	D_3
D_1	**读，写**	**执行**						switch
D_2	**执行**		**读**	**读**	**打印**	switch		
D_3	**执行**	**写**			**打印**			

第 12 章

操作系统设计原则

12.1 重点与难点

12.1.1 概述

在设计操作系统前，定义好系统的目标是非常重要的。所要求的系统类型是在不同算法和策略之间进行选择的基础。

一个操作系统相当大，因此，模块化是非常重要的。层次法是操作系统设计方法中的一个重要方法，而虚拟机是一种特殊的层次设计方法。

现代操作系统几乎总是用系统实现语言或高级语言来写的。这一特点改进了它们的实现、维护和可移植性。为了对一个特定机器配置构造一个操作系统，系统生成技术是必需的。

分布式操作系统工作于多机环境下，不同于单机操作系统，需要多个站点之间的协作， 它的设计有自己特殊的原则。

12.1.2 知识要点

1. 层次结构

所谓层次结构，就是把系统程序按照功能分成若干基本模块，再根据其作用和相互关联分别划归不同层次，各层之间具有单向依赖性，而不能构成循环。

层次结构分为全序层次结构和半序层次结构两种。在全序层次结构中，不仅各层之间是单向依赖的关系，而且每层中各模块之间也是相互独立的。在半序层

次结构中，各层次间单向依赖，不允许出现环路，而允许在同一层内模块间出现循环。

2. 虚拟机

在裸机上面添加一层软件来扩充该机器的功能，该软件层和裸机结合在一起，形成一台比原来机器性能更好、功能更强的机器，这种经软件改造过的机器称为虚拟机。

12.2 例 题 解 答

例 12-1　操作系统的设计为什么要采用分层的原则？

解　分层结构的主要优点是模块化。分层遵循的原则是每层仅使用其低层提供的服务设施和功能，这种结构使系统的调试和验证比较容易。在调试第一层时，不用关心系统的其他部分。因为，根据定义，它仅使用了基本硬件去实现其功能。一旦第一层调试通过，有关第二层的工作就可在它的基础上展开，然后逐层类推。如果在某层调试中发现错误，就可以知道错误的所在范围，因为在此层以下的功能均已调试通过。所以，操作系统的设计和实现通过分层而简单化了。

例 12-2　操作系统的层次划分有哪些参考原则？

解　根据人们大量的实践，可归纳出以下一些参考原则：

(1) 直接负责硬件资源驱动、分配和管理的模块（如 CPU 调度、I/O 设备驱动、中断处理等）要放在低层（最低层是硬件）。这样经过底层软件的扩充，硬件的特性就基本消失，而成为底层虚拟机。

(2) 直接与用户程序、库程序以及各种外部程序接口，提供系统服务的模块（如系统调用、作业控制等）要放在高层（最高层是用户程序）。这样容易改变操作系统的类型和服务。

(3) 把既非资源分配策略，又与机器特性无关的模块（如文件管理、I/O 格式加工等）放在中间层。因为在系统活动中，它们需要低层提供服务，而本身又是为高层提供服务的。

(4) 各层间遵循单向依赖关系。

这四点仅是参考原则，并非一定要遵循。因而在设计操作系统时，应根据具体情况来确定层次的多少以及各层的功能和组成。

例 12-3　设计分布式操作系统应考虑哪些问题?

解　设计分布式操作系统主要考虑以下三方面的问题。

(1) 分布式操作系统是为分布式多机系统设计的，因此，它不仅对于在各站

点上分别执行的任务及相关的资源有管理和控制的职责，而且还要负责协调各站点间的交互关系；不仅要保证在不同站点上执行的进程彼此互不干扰并严格同步，而且还必须保证避免或妥善地解决各处理机对某些资源的竞争所可能引起的死锁、饥饿及公平性等问题。

(2) 分布式操作系统的结构本身也应是分布式的，系统中各站点都可包含该操作系统或其中的一部分。一般的做法是，把分布式操作系统划分成若干不相交的并行模块，并把它们分别指派给系统中的各站点，且每个站点都有一个局部操作系统的核心模块副本。

(3) 分布式操作系统可划分为高层和局部，划分的一般原则是：

- 属于本机独立运行的基本管理功能归为本站点局部操作系统；
- 本站点与其他站点之间的通信、同步、消息传递等管理功能也归为本站点局部操作系统；
- 各站点协调合作完成的功能，如任务分配、负载平衡、死锁处理、错误检测、系统重构等归为高层操作系统。

(4) 分布式操作系统的基本调度单位不再是单机操作系统中的进程，而是一种任务队列，这种队列是由位于多个站点上的并发进程所组成的任何队列。

(5) 在分布式系统中，需要提供一个支持资源共享的环境，把任务进行分解并分配给相应的站点。若整个系统由不同类型的处理机组成，则由于每个处理机的处理能力及硬设备配置等方面都可能各具特点，因此，操作系统应根据这些特点给处理机分配任务。若系统由同类型的处理机组成，则在任何给定的时刻，任务可分配给任何一台处理机，并可随时进行任务的迁移，以平衡系统负载。

(6) 分布式操作系统的构成与分布式计算机系统的耦合方式关系很大。

(7) 分布式操作系统必须具有探测任一处理机停机或发生故障的能力，并采取适当的处理措施。

例 12-4　分布式操作系统分为哪几层？各层的功能是什么？

解　分布式操作系统大致分为四层，由低到高依次为执行层、进程通信层、服务支持层和用户接口层。

执行层主要是对分布式系统中分散的各类资源进行分配、回收、控制和管理。在该层上，各站点有自己的局部操作系统，它们一方面以自治的方式对本地资源进行管理，另一方面又相互合作，直至对全系统范围内的资源实现共同管理为止。

进程通信层主要以消息传递为基础，建立包括远程过程调用在内的进程间的同步和通信机制。

服务支持层主要提供若干基本的抽象逻辑结构的服务功能及大多数服务功能所需要的公共信息。

用户接口层是用户见到的分布式操作系统的外层，该层要为用户提供分布处

理的各种直接服务。

例 12-5　虚拟机实现的困难是什么？

解　虚拟机实现的困难在于如何提供下层机器的精确副本。例如，下层机有两种工作方式：用户方式和管理方式。由于虚拟机软件是操作系统，它可以在管态下运行；而虚拟机本身则只可在用户态下执行。因此，就像物理机有两种工作方式一样，虚拟机也必须有两种工作方式：虚拟用户方式和虚拟管理方式。二者都是运行在物理用户方式下。像执行系统调用或执行特权指令一样，这些活动在实际机器上从用户方式转到管理方式，在虚拟机上也必须从虚拟用户方式转到虚拟管理方式。

例 12-6　对于一个只有一组界限寄存器（1 个上界/1 个下界）的基本交互式分时系统，为下面各个操作系统功能模块，选取可实现的技术：

(1) 存储管理；

(2) CPU 调度；

(3) 文件系统目录；

(4) 死锁处理。

解　有许多可能的选择，比如，

(1) 存储管理——利用具有一个解释程序副本和数据空间的 MVT，为每一用户分配存储空间；必要时可采用交换技术。

(2) CPU 调度——时间片轮转法（RR），它适用于分时系统。

(3) 文件系统目录——主文件目录和用户文件目录，简单的两级目录结构。

(4) 死锁处理——利用资源定序和 SpooLing 技术来预防死锁。

12.3　自 测 练 习

一、填空题

1. 设计操作系统的首要问题是_____和_____。

2. 分布式操作系统是为_____系统配置的操作系统，它在多机环境下，负责控制和管理以_____方式工作的多种系统_____，_____的同步和执行，_____之间的通信、调度等控制事务，自动实行全系统范围内的_____分配和_____平衡，并具有高度并行性的一种高级操作系统。

3. 构造分布式操作系统的途径主要有_____、_____和_____。

4. 分布式操作系统的逻辑结果主要有五种，它们分别是_____、_____、_____、_____和_____。

5. 设计操作系统的一个重要原则是机制与策略的分离，机制确定_____，而策略确定_____。

6. 层次结构就是把系统程序按照_____分成若干基本模块，再根据其_____分别划归不同层次，各层之间具有_____性，而不能构成_____。

二、判断题（正确的在括号中记√，错误的记×）

1. 一个虚拟机的虚拟内存可大于或小于物理机的实际内存。（　　）

2. 在分层结构中，高层为低层提供服务，低层调用高层。（　　）

三、选择题

1. 在分布式操作系统的设计中，可归为高层的有哪些？（　　）

a) 站点之间的通信；　　b) 任务分配和负载平衡；

c) 站点之间的同步；　　d) 错误检测和系统重构。

2. 在单机操作系统中，任务调度的基本单位是什么？（　　）

a) 进程；　b) 线程；　c) 任务队列；　d) 指令。

3. 在分布式操作系统中，任务调度的基本单位是什么？（　　）

a) 进程；　b) 线程；　c) 任务队列；　d) 指令。

4. 考虑一个分层式的操作系统，假定最低层是硬件，最高层是用户。对于下列 5 个成分由低到高的次序是什么？（　　）

① 分页式存储管理；　② CPU 调度；

③ I/O 调度和处理；　④ Pascal 编译程序；

⑤ 控制卡片解释程序。

a) ①②③④⑤；　b) ③②①④⑤；

c) ②①③⑤④；　d) ②③①⑤④。

5. 图 12.1 所示的是一个模块关系图，矩形代表一个模块，箭头代表模块间的依赖关系，在同一水平线的模块属于同一层次，则该图是什么结构？（　　）

a) 该图是全序层次结构；

b) 该图是半序层次结构；

c) 该图是层次结构；

d) 该图不是层次结构。

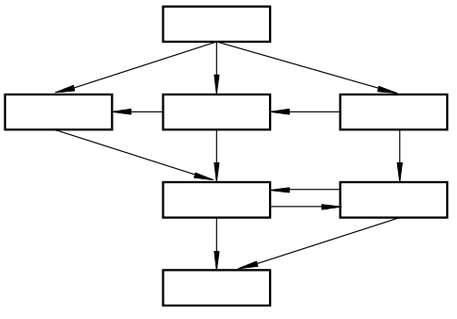


图 12.1

四、问答题

1. 假定要为工程系的学生设计一个提供 BASIC 和 APL 的分时操作系统，该系统有 22 个 CRT 终端，一个行式打印机和两个较大的可移动头磁盘，硬件支持“用户/监控”态，带保护的 I/O 指令，并提供分页存储管理机制。简述对如下问题的处理方法。

(1) CPU 调度；　　(2) 存储管理；

(3) 磁盘调度；　　(4) 文件系统；

(5) 用户标识和存取控制保护；　　(6) 死锁。

2. 虚拟机概念的益处有哪些？

3. 对于一个有二组界限寄存器（1 个上界/1 个下界）的基本交互式分时系统，解释下面的操作系统成分：

(1) 存储管理；　　(2) CPU 调度；

(3) 文件系统目录；　　(4) 死锁处理。

4. 在一个虚拟机环境中，只提供宿主机的副本而不提供任一机器的副本是否要简单些？

12.4　自测练习答案

一、填空题

1. 设计操作系统的首要问题是 定义其目标 和 系统描述 。

2. 分布式操作系统是为 分布式计算机 系统配置的操作系统，它在多机环境下，负责控制和管理以 协同 方式工作的多种系统 资源 ， 进程 的同步和执行， 处理机 之间的通信、调度等控制事务，自动实行全系统范围内的 任务 分配和 负载 平衡，并具有高度并行性的一种高级操作系统。

3. 构造分布式操作系统的途径主要有 从头开始 、 修改扩充 和 层次式 。

4. 分布式操作系统的逻辑结果主要有五种，它们分别是 DOS 核 、 集成式 、 client/server 模型 、 中央式 和 分散式 。

5. 设计操作系统的一个重要原则是机制与策略的分离，机制确定 如何做 ，而策略确定 做什么 。

6. 层次结构就是把系统程序按照 功能 分成若干基本模块，再根据其作用和相互关联分别划归不同层次，各层之间具有单向依赖性，而不能构成循环。

二、判断题

1. √　　2. ×

三、选择题

1. b）、d）　　2. a）　　3. c）
4. d）　　5. b）、c)

四、问答题

1. (1) CPU 调度——采用时间片轮转（RR）调度算法。
   (2) 存储管理——采用请调式分页存储管理技术。
   (3) 磁盘调度——采用 SSTF 技术。
   (4) 文件系统——索引文件，用文件的索引表去匹配内存中的页表。
   (5) 用户标识和存储保护——采用用户 id 和口令。
   (6) 死锁——采用 SPOOLing 技术来避免对大多数 I/O 设备的直接控制，对剩下的资源采用资源定序方法处理。

2. 一般来说，系统调用和文件系统等，都不是由裸机提供的。这样，在裸机上添加一层软件来扩充其功能，形成一台比原来机器性能更好、功能更强的机器，这种经软件改造过的机器称为虚拟机。虚拟机可以递归扩充。物理机资源通过共享可变为多个虚拟机，CPU 调度程序使各个进程共享 CPU，从而每个用户都感觉有自己的处理机。而请求分页技术可为每个虚处理机提供它自己使用的虚拟存储器。

3. 可参考例题解答中的例 12-6。唯一的变化在存储管理方面。在本题中，可对每一用户提供两个存储段：一段用于存放固定的、只读的 APL 或 BASIC 解释程序或其他操作系统程序的可再入代码副本，它任何时候都保留在内存中；另一段随用户的不同而不同，它可以是每一个用户的程序和数据空间。仍然可以用 MVT 技术来分配这些段。

4. 是的。宿主机的硬件设施可用来支持该虚拟机。对于任一机器，每一事务可能都得解释执行。

第 13 章

综 合 测 试

模拟试题 I

一、填空题 （每空 1 分，共 20 分）

1. 系统服务主要分为_____和_____两种。
2. 进程的五个基本状态是：_____、_____、_____、_____和_____。
3. 将系统花在页面替换上的时间远远多于执行进程的时间的情况称为_____。
4. 进程调度分为_____、_____和_____三类。
5. 常见的磁盘调度算法有_____、_____、_____和 C-SCAN 等几种。C-SCAN 算法在实际实现时常变为_____算法。
6. 一次仅允许一个进程使用的资源称为_____。例如，打印机、磁带机、输入机等物理设备。进程访问它的一段程序，叫做_____。
7. 虚拟机有_____和_____两种实现方式，二者都运行在_____方式下。

二、判断题 （正确的在括号中记√，错误的记×。每题 2 分，共 20 分）

1. 设备独立性就是指系统具有使用不同设备的能力。 ( )
2. FCFS 进程调度的实现过程可以用 FIFO 队列管理。 ( )
3. 在请式调页中，增加内存帧一定可以降低缺页中断率。 ( )
4. HRRN 调度算法有利于长作业的执行。 ( )
5. s 在 SPOOLing 系统中，用户进程可以直接高效地使用字符设备。 ( )
6. 磁盘空间分配中，采用链接分配方式分配存储不会产生外部碎片，但可能产生内部碎片。 ( )
7. 如果系统用 Banker 算法处理死锁，那么，当某进程要增大其 Max 值，且仅当每一进程的 Max 请求数不超过可用资源的总数时，系统才保持在安全态，不会产生死锁。 ( )

8. 在一个系统中，有一台大型主机和若干终端，所有终端通过网络与主机相连，终端仅能用于文字输入，而主机则接收这些输入信息然后进行处理。该系统是一个分布式系统。（　　）

9. 进程图表示了进程的创建关系。在一个进程图中，P_i 到 P_j 的边隐含 P_i 只能在 P_j 之后执行。（　　）

10. 在各种磁盘调度算法中，最短寻道时间优先法是最优的磁盘寻道算法。（　　）

三、简答题 （共 20 分）

1. (7分) 什么是操作系统？其功能是什么？在研究操作系统的过程中主要取得了哪些重要成就？

2. (7 分) 什么是死锁？什么是饥饿? 死锁和饥饿有何差别？

3. (6 分) 设两个相继的语句 S_1 和 S_2 的读集合分别是 $R(S_1)$、$R(S_2)$, 写集合分别是 $W(S_1)$、$W(S_2)$，则 S_1 和 S_2 可以并发地执行并产生相同的结果必须满足哪些条件？

四、问答题 （共 40 分）

1. (10 分) 假定把表 13.1 所示的 4 个作业同时提交给系统并进入后备队列，若使用最短作业优先调度算法，则作业的平均等待时间是多少？若使用最高优先数队列的调度算法，则作业的平均周转时间是多少？

表 13.1

作　　业	所需运行时间/h	优先数
1	2	4
2	5	9
3	8	2
4	3	8

2. (15 分) 设 n 个进程共享 m 个资源，每个进程一次只能预定或释放一个资源，每个进程最多需要 m 个资源，所有进程总共的需求少于 m+n 个资源。证明：此时不会发生死锁。

3. (15 分) 设有一个数据区，有若干进程要去读或写它。写是互斥的，读可以同时进行。考虑进程分别遵循下面两种原则：

(1) 读者优先。Reader 进程不需要等待，除非有某个 Writer 进程正在执行。也就是说，如果某个 Reader 进程正处于等待状态，则新的 Writer 进程就不能开始写操作。

(2) **写者优先。**Writer **进程不需要等待，除非有某个** Reader/Writer **进程正在执行。也就是说，如果某个** Writer **进程正处于等待状态，则新的** Reader **进程就不能开始读操作。**

请用 P/V **操作写出读/写过程的同步算法，并给出所设信号量的初值。**

模拟试题Ⅰ参考答案

一、填空题

1. **系统服务主要分为系统调用和系统程序两种。**
2. **进程的五个基本状态是：运行、就绪、阻塞、创建和消失。**
3. **将系统花在页面替换上的时间远远多于执行进程的时间的情况称为抖动。**
4. **进程调度分为长程调度、中程调度和短程调度三类。**
5. **常见的磁盘调度算法有** FCFS**、**SSTF**、**SCAN **和** C-SCAN **等几种。**C-SCAN **算法在实际实现时常变为** C-LOOK **算法。**
6. **一次仅允许一个进程使用的资源称为临界资源。例如，打印机、磁带机、输入机等物理设备。进程访问它的一段程序，叫做临界段。**
7. **虚拟机有** 虚拟用户 **和** 虚拟管理 **两种实现方式，二者都运行在** 物理用户 **方式下。**

二、判断题

1. ×	2. √	3. ×	4. ×	5. ×
6. √	7. √	8. ×	9. ×	10. ×

三、简答题

1. **操作系统是计算机用户和计算机硬件之间的接口程序模块，它是计算机系统的核心控制软件，其职能是控制和管理系统内各种资源，有效地组织多道程序的运行，从而为用户提供良好的工作环境，达到使用方便、资源分配合理、安全可靠等目的。**

操作系统是现有软件系统中最复杂的软件之一。到目前为止，在操作系统的研究开发方面取得了进程、内存管理、信息保护与安全、调度与资源管理和系统结构五项主要成就。

2. **死锁是在系统运行过程中的某一时刻，一组进程中的每一个进程都占用着一些资源，同时又想得到该组中其他进程占用的资源。这样，该组中无论哪一个进程都得不到满足，因而该组中的所有进程都无法继续运行，这种情况称为系统**

发生了死锁。饥饿是系统并没有死锁，但至少有一个进程被无限期地推迟。饥饿不同于死锁，死锁是这样一种情形，即其中某进程正等待一个决不会发生的事件；而饥饿现象是指某进程正等待这样一个事件，它发生了但总是受到其他进程的影响，以致轮不到（或很难轮到）该进程。

3. 必须满足下列三个条件：

(1) $R(S_1) \cap W(S_2) = \{\}$

(2) $W(S_1) \cap R(S_2) = \{\}$

(3) $W(S_1) \cap W(S_2) = \{\}$

四、问答题

1. 对于给定的四个作业，当采用最短作业优先调度算法时，作业的执行次序是 1，4，2，3，它们的等待时间分别为 0，2，5，10。所以，作业平均等待时间是

$$(2 + 5 + 10)\text{ h}/ 4 = 4.25\text{ h}$$

若用最高优先数优先调度算法，则作业的执行次序是 2，4，1，3，其周转时间分别为 5，8，10，18。因此， 作业的平均周转时间是

$$(5 + 8 + 10 + 18)\text{ h}/ 4 = 10.25\text{ h}$$

2. 证明：依题意，对于$\forall i \in (1，n)$，有

$$l \leq Max_i \leq m$$

$$\sum_{i=1}^{n} Max_i < m + n$$

$$\sum_{i=1}^{n} Need_i = \sum_{i=1}^{n} Max_i - \sum_{i=1}^{n} Allocation_i$$

假设系统处于死锁状态，则有 $\sum_{i=1}^{n} Allocation_i = m$

所以 $$\sum_{i=1}^{n} Need_i < (m + n) - m = n$$

因此，至少存在一个进程 P_i，$Need_i=0$。与题意矛盾，假设不成立。

所以，该系统不会产生死锁。

3. (1)
```
var mutex：semaphore;      {初值=1}
    wrt：semaphore;        {初值=1}
    readcount：integer;    {初值=0}
```

Reader 进程：

```
repeat
```

Writer 进程：

```
repeat
```

```
P(mutex);
  Readcount：= readcount + 1;
  If    readcount = 1 then P(wrt);
V(mutex);
  …
  读数据区
  …
P(mutex);
  Readcount  ：= readcount – 1;
  If    readcount    = 0 then V(wrt);
  V(mutex);
until false;
```

```
P(wrt);
  …
  写入数据区
  …
 V(wrt);
until false;
```

(2)

```
var mutex：semaphore;  {初值=1}
 rd：  semaphore;          {初值=1}
 wrtcount：integer;        {初值=0}
```

Reader 进程：

```
repeat
  P(rd);
    …
    读数据区
    …
  V(rd);
until false;
```

Writer 进程：

```
repeat
  P(mutex);
    Wrtcount：= wrtcount + 1;
    If    wrtcount = 1 then P(rd);
  V(mutex);
    …
    写入数据区 s
    …
  P(mutex);
    Wrtcount  ：= wrtcount – 1;
    If    wrtcount    = 0 then V(rd);
  V(mutex);
until false;
```

模拟试题 II

一、填空题 （每空 1 分，共 20 分）

1. Denning 认为到目前为止在操作系统的研究开发方面取得了_____、

_____、_____、_____和_____五项主要成就。

2. 进程控制块含有进程的_____信息和_____信息，它是_____中最关键的部分。

3. 对进程调度算法的评估方法通常有_____、_____和_____。

4. 当前操作系统的内存管理方法主要有_____、_____和_____三种。

5. Banker 算法是著名的死锁_____算法，它不严格限制产生死锁的必要条件的存在，只是检测死锁并保证系统处于_____状态。

6. 进程之间常常相互作用，并存在某种彼此依赖或相互制约的关系，这些关系按其性质可分为 _____ 和 _____ 两种关系。

7. 任何分布式操作系统中的关键机制都是进程间的通信，经常使用的技术主要是_____和_____。

二、判断题 （正确的在括号中记√，错误的记×。每题 2 分，共 20 分）

1. SPOOLing 对批处理多道程序设计是必须的。 ( )

2. I/O 设备的速度远小于 CPU 的速度。 ( )

3. 在分时系统中快速响应是必须的。 ( )

4. 在文件系统采用的磁盘空间分配算法中，链接分配方法比毗连分配方法慢，因为在存取文件时磁头可能会在各块之间来回移动。 ( )

5. 内存管理的分段方法和 MVT 方法的不同之处在于分段方法有外部碎片，而 MVT 没有。 ( )

6. 磁鼓比磁盘更适于做分页设备。 ( )

7. 采用修改位的算法可以减少不必要的页面替换。 ( )

8. 若系统处于不安全状态，则一定产生了死锁。 ( )

9. 进程是一个独立运行单位，能与其他进程并行执行。而通常的程序段不能作为一个独立运行单位，也不能和其他进程并行地执行。 ( )

10. 分布式操作系统中消息传递的先于关系不具有传递性。 ( )

三、简答题 （共 20 分）

1. (7 分) 进程和程序的关系和区别是什么？

2. (8 分) 分布式操作系统的任务分配和负载平衡方法有哪些？有什么缺点？

3. (5 分) 什么叫临界资源？什么是临界段？

四、问答题 (共 40 分)

1. (10 分) 分页式内存管理和分段式内存管理的主要区别是什么？

2. (15 分) 考虑哲学家用餐问题，下面的算法可能导致死锁吗？试说明理由。

若是，请对算法作出适当的修改。

```
var fork： array[0..4] of semaphore      (初值=1);
        i： integer;
procedure philosopher(i： integer);
begin
    repeat
      think;
      wait(fork[i]);
      wait(fork[i+1] mod 5);
      eat;
      signal(fork[(i+1) mod 5]);
      signal(fork[i]);
  until false;
end;
begin
    parbegin
        philosopher(0);
        philosopher(1);
        philosopher(2);
        philosopher(3);
        philosopher(4);
    parend
end.
```

3. (15 分) **设若干个生产者和若干个消费者共用一个有** n **个单元的有限缓冲区，为了使产销进程协调进行，有人利用** P/V **操作，对问题作如下安排：**

引入两个私有信号量和一个公用信号量：full(**初值为** 0**，产品计数**) **，**avail(**初值为** n**，可用缓冲区数**) **和** mutex(**初值为** 1**，表示没有进程正在使用缓冲区**)**。下面程序不完全，请在方框内填进适当的** P/V **操作。**

```
var full, avail, mutex： semaphore;
begin
    full ：= 0;
    avail ：= n;
    mutex ：= 1;
parbegin
    producer：
```

```
        repeat
            …
            produce next product;
            …
        ①
        ②
            Add to buffer;
        ③
        ④
        Until false;
    Consumer：
        repeat
        ⑤
        ⑥
            take from buffer;
        ⑦
        ⑧
            …
            consume product;
            …
        Until false;
  parend;
end.
```

模拟试题II参考答案

一、填空题

1. Denning 认为到目前为止在操作系统的研究开发方面取得了进程、内存管理、信息保护与安全、调度与资源管理和系统结构五项主要成就。

2. 进程控制块含有进程的描述信息和控制信息，是进程映像中最关键的部分。

3. 对进程调度算法的评估方法通常有分析评估法、模拟方法和实现法。

4. 当前操作系统的内存管理方法主要有分页式、分段式和段页式三种。

5. Banker 算法是著名的死锁避免算法，它不严格限制产生死锁必要条件的存在，只是检测死锁并保证系统处于安全状态。

6. 进程之间常常相互作用，并存在某种彼此依赖或相互制约的关系，这些关系按其性质可分为同步和互斥两种关系。

7. 任何分布式操作系统中的关键机制都是进程间的通信，经常使用的技术主要是 消息传递 和 远程过程调用 。

二、判断题

1. √　2. √　3. ×　4. √　5. ×
6. √　7. √　8. ×　9. √　10. ×

三、简答题

1. 进程是程序在并发环境中的执行过程。其基本特征是动态性、并发性、独立性、异步性和结构性。

进程和程序之间的主要区别是：

(1) 程序是静态概念，本身可以作为一种软件资源保存；而进程是程序的一次执行过程，是动态概念，它有一定的生命期，是动态地产生和消亡的。

(2) 进程是一个能独立运行的单位，能与其他进程并发执行，进程是作为资源申请和调度单位存在的；而通常的程序段不能作为一个独立运行的单位。

(3) 程序和进程无一一对应关系。一方面，一个程序可由多个进程共用；另一方面，一个进程在活动中又可顺序地执行若干个程序。

2. 任务分配和负载平衡方法主要有发送者主动、接收者主动、预约、投标等方法。主要缺点是：

(1) 系统状态的失效性；

(2) 系统状态的不完整性和不确定性；

(3) 策略的单一性；

(4) 缺乏自我调节能力。

3. 在计算机系统中，同时有许多进程，它们共享着各种资源，然而有许多资源一次却仅能为一个进程所使用。我们把一次仅允许一个进程使用的资源称为临界资源。例如，物理设备属于临界资源，如，打印机、磁带机、输入机等。除了物理设备外，还有很多变量、数据、表格、队列等也都是由若干进程所共享的。进程访问共享临界资源的一段程序，叫做临界段。

四、问答题

1. 分页式内存管理和分段式内存管理的主要区别如表 13.2 所示。

表 13.2

序号	分页式管理	分段式管理
1	单一连续空间	二维地址空间
2	页是信息的物理单位	段是信息的逻辑单位
3	页的大小固定，有系统分划，对用户透明	页的大小固定，用户可见
4	(不具有分段式此类的特点)	便于动态链接、存储保护，便于增长、修改和共享
5	分配页面大小的存储空间	分配段大小的存储空间，要采用拼接技术

2. 可能导致死锁的发生。当每个哲学家都同时先拿起其左边的筷子然后再试图拿右边筷子时就产生了死锁。可以通过限制最多只有四个哲学家同时进餐，以避免死锁。算法可修改为

```
var fork： array[0..4] of semaphore      (初值=1);
           room： semaphore              (初值=4);
           i： integer;
procedure philosopher(i： integer);
begin
      repeat
         think;
         wait(room);
         wait(fork[i]);
         wait(fork[i+1] mod 5);
         eat;
         signal(fork[(i+1) mod 5]);
         signal(fork[i]);
         signal(room)
  until false;
end;
begin
      parbegin
            philosopher (0);
            philosopher (1);
            philosopher (2);
```

```
        philosopher (3);
        philosopher (4);
    parend
end.
```

3. ① P(avail)　② P(mutex)　③ V(mutex)　④ V(full)
   ⑤ P(full)　⑥ P(mutex)　⑦ V(mutex)　⑧ V(avail)

模拟试题III

一、填空题 （每空 1 分，共 20 分）

1. 人们早期在计算机上运行一个程序必须经过_____ 、_____、_____和_____四个步骤。

2. 最高响应比优先(HRRN)算法采用的是_____和_____两种算法的折衷方案。

3. 页表的三种实现方法是用_____实现、用_____实现和用_____实现。

4. 总线结构中通常采用的 I/O 方式有_____和_____两种。

5. 采用链接分配方式分配磁盘空间不会产生_____碎片，但可能产生_____碎片。

6. 操作系统中，可以并行工作的基本单位是_____，它也是核心调度及资源分配的基本单位。当因资源竞争可能会引起死锁时，可以采用死锁避免和预防、_____两种策略来解决，其中，_____所付出的代价较高。

7. 并发进程中涉及到_____的程序段称为临界段，两个进程同时进入相关的临界段会造成_____的错误。

8. 在一个用先于关系确定事件顺序的分布式操作系统中，如果事件 a 和事件 b 之间不存在 a→b 和 b→a 的关系，则可以称 a 和 b 为_____事件，它们可以_____执行。

二、判断题 （正确的在括号中记√，错误的记×。每题 2 分，共 20 分）

1. 在操作系统提供的大量服务中，最低层的服务是系统调用。 (　　)

2. 进程的基本特征是动态性、并行性、独立性、异步性和结构性。 (　　)

3. 最高优先级（HPF）算法总是让具有高优先级的进程获得优先服务，因此，是抢占式的算法。 (　　)

4. RR 算法的性能依赖于时间片的大小，当时间片过大时称为处理机共享。

(　　)

5. 决定缺页中断时间的主要因素包括：中断服务时间、交换页面的时间和重启进程的时间。(　　)

6. 中型计算机系统通常采用总线结构进行设备的数据交换。(　　)

7. 在分配磁盘空间的三种方法中，链接分配方法最慢，因为磁头可能不得不在存取文件之间移动。(　　)

8. 某系统由相同类型的四个资源组成，若资源可被三个进程共享，每个进程最多可申请两个资源，则该系统不会发生死锁。(　　)

9. 进程从运行状态进入就绪状态的原因可能是时间片用完了。(　　)

10. 在分布式操作系统中，进程间的通信可以借助于公共存储器，也可以采用消息传递的方式。(　　)

三、简答题 （共 30 分）

1. (7 分) 请描述在当前运行进程状态改变时，操作系统进行进程切换的步骤。

2. (7 分) 在请式调页中，若采用 LRU 算法作为替换算法来选择页面替换的牺牲者，有哪些常用的实现方法？

3. (8 分) 在组织文件时，可采用哪些数据结构？它们各自的优缺点是什么？

4. (8 分) 有三个进程 P_1、P_2 和 P_3 并发工作。进程 P_1 需用资源 S_3 和 S_1；进程 P_2 需用资源 S_1 和 S_2；进程 P_3 需用资源 S_2 和 S_3。试问：

(1) 若对资源分配不加限制，会发生什么情况？试举例说明。

(2) 为保证进程正确工作，应采用怎样的资源分配策略？为什么？

四、问答题 （共 30 分）

1. (8 分) 假定要在一台处理机上执行下列作业：

作　　业	执行时间	优先级
1	4	2
2	3	1
3	8	3
4	1	4
5	2	3

且假定这些作业到达的次序是 1，2，3，4，5。

(1) 用 Gantt 图来说明分别使用 FCFS，RR(时间片=1)，SJF 以及非抢占优先调度算法（Priority）时这些作业的执行情况；

(2) 针对上述每一调度算法，给出与每个作业相应的轮转时间；

(3) 就上述每一调度算法，求出与每个作业相应的等待时间。

2．（10 分）一个进程被分配了四个页帧。表 13.3 列出了这四个页的虚拟页号，各页被装入页帧的时间，各页最近被访问的时间，以及各页的引用位和修改位（时间从进程启动时开始计算）。

表 13.3

虚拟页号	页帧号	装入时间	引用时间	引用位	修改位
2	1	60	161	0	1
1	2	130	160	0	0
0	3	26	162	1	0
3	4	20	163	1	1

(1) 假如发生一个访问页号为 4 的页面缺页中断，试分别使用 FIFO、LRU、NUR 算法来选择淘汰页面。

(2) 对于下面的内存访问序列：

4，0，0，0，2，4，2，1，0，3，2

如果使用窗口大小为 4 的工作集方法来取代固定分配页帧的方法，则会出现多少次缺页中断？

3. 有四个进程 $prod_1$、con_1 和 $prod_2$、con_2，共享一个具有 n 个缓冲区的缓冲池 BUF[1..n]（n≥1）。其中，$prod_1$ 是一个计算进程，把计算结果填入缓冲区；con_1 是一个输出进程，它把 $prod_1$ 进程放在缓冲区的计算结果输出打印；$prod_2$ 是一个输入进程，把数据读入缓冲区；con_2 是一个计算进程，它计算由 $prod_2$ 放在缓冲区中的数据。每个进程按下述方式运行。

(1) 按 FIFO，取队列 buf 中的一个缓冲区，得到该缓冲区的编号 i(i ：= GET(buf))。

(2) 按缓冲区 BUF[i]分别进行

计算：COMPUTE（BUF[i]）

输出：OUTPUT（BUF[i]）

输入：INPUT（BUF[i]）

(3) 释放缓冲区 BUF[i]到相应队列 buf (RELEASE(BUF[i]，buf))。

(4) 重复第(1)步：goto (1)。

试问：应有怎样的数据结构来控制缓冲区的共享？并请用信号量操作及上述已知过程写出这四个进程的运算步骤。

模拟试题III参考答案

一、填空题

1. 人们早期在计算机上运行一个程序必须经过预约机器时间、将程序手工装入内存、指定开始地址启动程序运行和从控制台上监控程序的执行四个步骤。

2. 最高响应比优先(HRRN)算法采用的是 FCFS 和 SJF 两种算法的折衷方案。

3. 页表的三种实现方法是用专用寄存器实现、用内存指定区域实现和用联想寄存器实现。

4. 总线结构中通常采用的 I/O 方式有中断处理方式和 DMA 方式两种。

5. 采用链接分配方式分配磁盘空间不会产生外部碎片，但可能产生内部碎片。

6. 操作系统中，可以并行工作的基本单位是进程，它也是核心调度及资源分配的基本单位。当因资源竞争可能会引起死锁时，可以采用死锁避免和预防、死锁检测和恢复两种策略来解决，其中，死锁检测和恢复所付出的代价较高。

7. 并发进程中涉及到共享变量的程序段称为临界段，两个进程同时进入相关的临界段会造成与时间有关的错误。

8. 在一个用先于关系确定事件顺序的分布式操作系统中，如果事件 a 和事件 b 之间不存在 a→b 和 b→a 的关系，则可以称 a 和 b 为 并发 事件，它们可以 并发 执行。

二、判断题

1. √	2. ×	3. ×	4. ×	5. √
6. ×	7. ×	8. √	9. √	10. ×

三、简答题

1. 在当前运行进程的状态改变时，操作系统进行进程切换的步骤如下；

(1) 保存处理器内容。

(2) 对当前运行进程的 PCB 进行更新，包括改变进程状态和其他相关信息。

(3) 将这个进程的 PCB 移入适当的队列（就绪、因事件阻塞、就绪挂起等）。

(4) 挑选其他进程执行。

(5) 对所挑选进程的 PCB 进行更新，包括将其状态改为运行。

(6) 对存储器管理数据结构进行更新。

(7) 恢复被选择进程上次移出时的处理器状态。

2. 在实现 LRU 算法时，通常使用下面的方法来确定哪一页是牺牲者：

(1) 利用硬件计数器和（或）时钟。页表中包含页每次使用之后被更改的时间，具有“最老”时间的页即为牺牲者。

(2) 利用页号栈。每次使用某页时，将对应的页号压入栈顶，已在栈中的页号则依次下移，具有栈底页号的那个页即为牺牲者。

(3) 利用一个访问位。每次使用某页时，就将其访问位置值，并在约定的时间间隔后清除该位，牺牲者即为其访问位已被清除的那个页。

(4) 利用一个访问字节。一到给定的时间间隔，每个页的访问字节就右移，但在使用某页时，其对应的访问字节的符号位已被置值。将具有最低值访问字节的那个页作为牺牲者。

3. 在组织文件时，可以采用的数据结构及各自的优缺点如下：

(1) 线性表。优点是查找简单；缺点是存取文件的速度较慢。

(2) 链接表。优点是易于实现增、删操作；缺点是需要为指针分配存储空间。而且当指针指错位置时，其后的文件信息将会丢失或者发生混乱。

(3) 排序表。优点是查找快速；缺点是排序表总是要求所需的表是排过序的，这意味着在创建和删除文件时需要做一些额外的工作。

(4) 链接二叉树。优点是查找较快速；缺点与链接表相同。

(5) 杂凑表。优点是查找速度最快；缺点是要事先考虑到文件的最大个数而且还得设法解决冲突问题。

4. 回答如下：

(1) 若对资源分配不加限制，则有可能会发生死锁。例如，进程 P_1、P_2 和 P_3 分别获得资源 S_3、S_1 和 S_2 后，再继续申请另一资源时都要等待，且这是循环等待。由于各进程在等待新资源时均不释放已占用的资源，从而导致死锁，如图 13.1 所示。

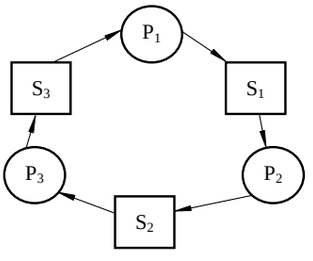


图 13.1

(2) 为保证进程能正确工作，可采用如下资源分配策略之一：

① 采用静态预分配策略，即一次申请所有的资源。由于执行前已获得所需的全部资源，故不会出现占用资源又等待别的资源的情况，从而可以预防死锁的发生。

② 采用资源定序方法，即将所有资源按类型线性排队，并按递增规则编号。由于进程只能以递增方式申请资源，故不会导致循环等待现象，从而预防了死锁。

③ 采用银行家算法。因为在分配时，已保证了系统处于安全状态，从而避免了死锁的发生。

四、问答题

1. (1) 表示其执行情况的 Gantt 图如图 13.2 所示。

FCFS

1(4)	2(3)	3(8)	4(1)	5(2)

RR

1	2	3	4	5	1	2	3	5	1	2	3	1	3	3

SJF

4	5	2	1	3

Priority

2	1	3	5	4

图 13.2

(2) 与这些作业相对应的轮转时间如表 13.4 所示。

表 13.4

作　业	FCFS	RR	SJF	Priority
1	4	13	12	7
2	7	11	6	3
3	15	18	18	15
4	16	4	1	18
5	18	9	3	17

(3) 与这些作业相应的等待时间如表 13.5 所示。

表 13.5

作　业	FCFS	RR	SJF	Priority
1	0	9	8	3
2	4	8	3	0
3	7	10	10	7
4	15	3	0	17
5	16	7	1	15

2. (1) 若使用 FIFO 方法来选择淘汰页面，则应为装入时间最早的页面，即为第 3 页。若使用 LRU 方法来选择淘汰页面，则应为最近访问时间最早的页面，即为第 1 页。若使用 NUR 方法来选择淘汰页面，则应选择未引用且未修改的页

面。这里应选择引用位和修改位均为 0 的页面，即为第 1 页。

注：根据表 13.3 中所示的信息，2 号页的引用位为 0，但修改位为 1，并且该页在 161 时刻被引用。0 号页的引用位是 1，并且该页在 162 时刻被引用。这表明在 2 号页在 161 时刻被引用后，系统执行了一次引用位清零操作。

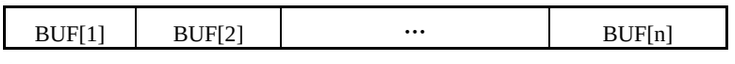
(2) 使用工作集方法的运行情况如表 13.6 所示。从表中可以看出，总共会产生 5 次缺页中断。

表 13.6

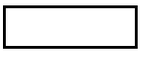
访问页面	当前工作集	是否产生缺页中断
4	{1，2，0，3}	是
0	{2，0，3，4}	否
0	{0，3，4}	否
0	{0，3，4}	否
2	{0，3，4，2}	是
4	{0，3，4，2}	否
2	{0，3，4，2}	否
1	{3，4，2，1}	是
0	{4，2，1，0}	是
3	{2，1，0，3}	是
2	{2，1，0，3}	否

3. 应有三个缓冲区队列 Pool、Out 和 in，其初始状态为

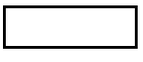
Pool：



Out：



In：



每个队列设置两个信号量，其中，一个信号量记录当前队列中的缓冲区个数，另一个信号量控制并发进程对同一队列的互斥操作：

var pool, out, in, M_p, M_o, M_i： semaphore

prod1：

repeat

P(pool);
P(M_p);
i = GET(pool);
V(M_p);
COMPUTE(BUF[i]);
P(M_o);
RELEASE(BUF[i], out);
V(out);
V(M_o);
until false;

con1：

repeat
P(out);
P(M_o);
i = GET(out);
V(M_o);
OUTPUT(BUF[i]);
P(M_p);
RELEASE(BUF[i], pool);
V(pool);
V(M_p);
until false;

prod2：

repeat
P(pool);
P(M_p);
i = GET(pool);
V(M_p);
INPUT(BUF[i]);
P(M_i);
RELEASE(BUF[i], in);
V(in);
V(M_i);
until false;

con2：

```
repeat
        P(in);
        P(Mi);
        i = GET(in);
        V(Mi);
        COMPUTE(BUF[i]);
        P(Mp);
        RELEASE(BUF[i], pool);
        V(pool);
        V(Mp);
until false;
```

模拟试题IV

一、填空题 （每空 1 分，共 30 分）

1. 操作系统是计算机系统中的一个_____，它管理和控制计算机系统中的_____。

2. 在操作系统中，不可中断执行的操作称为_____。

3. 并发进程中涉及到相同变量的程序段叫做_____，对这些程序段要_____执行。

4. 当系统采用资源有序分配方法预防死锁时，它破坏了死锁的必要条件中的_____条件。

5. _____优先运行的调度算法能够缩短作业的平均等待时间。

6. 在有一个 CPU 和两台外设 d_1 和 d_2 且能够实现抢占式优先数调度算法的多道程序环境中，同时进入优先级由高到低为 p_1，p_2，p_3 的三个作业，每个作业的处理顺序和使用资源的时间如下。

p_1：d_2(20ms), CPU(10ms), d_1(30ms), CPU(10ms)

p_2：d_1(20ms), CPU(20ms), d_2(40ms)

p_3：CPU(30ms), d_1(20ms)

假设其他辅助操作时间忽略不计，则 p_1，p_2，p_3 每个作业的周转时间分别为_____、_____和_____；CPU 的利用率为_____，d_1 的利用率为_____。

7. 在请求页式系统中，LRU 是_____置换算法，LFU 是_____置

换算法。

8. 目前认为逻辑文件有两种类型，即_____式文件与_____式文件。

9. 活动头磁盘的访问时间包括_____、_____和_____。

10. 在 Unix 文件管理系统中，为了对磁盘空间的空闲块进行有效的管理，所采用的方法是_____。

二、选择题 （每题 3 分，共 30 分）

1. 下列几种关于进程的叙述中，最不符合对操作系统中的进程的理解的是什么？ (　　)

a) 进程是在多程序并行环境中的完整的程序；

b) 进程可以由程序、数据和进程控制块描述；

c) 线程是一种特殊的进程；

d) 进程是程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。

2. 在操作系统中，P／V 操作是一种什么命令 (　　)

a) 机器指令；　　b) 系统调用命令；

c) 作业控制命令；　　d) 低级进程通信原语。

3. 若信号量 S 的初值为 2，当前值为–1，则表示有多少个等待进程？ (　　)

a) 0 个；　　b) 1 个；　　c) 2 个；　　d) 3 个。

4. 设有四个作业同时到达，每个作业执行时间均为 2h，它们在一台处理器上按单道方式运行，则平均周转时间为多少。 (　　)

a) 1h；　　b) 5h；　　c) 2.5h；　　d) 8h。

5. 采用什么存储管理不会产生内部碎片? (　　)

a) 分页式存储管理；　　b) 分段式存储管理；

c) 固定分区式存储管理；　　d) 段页式存储管理。

6. 把作业地址空间使用的逻辑地址变成内存的物理地址称为什么？ (　　)

a) 加载；　　b) 重定位；　　c) 物理化；　　d)逻辑化。

7. 虚存是什么？ (　　)

a) 提高运算速度的设备；　　b) 容量扩大了的内存；

c) 实际不存在的存储器；　　d) 进程的地址空间及其内存扩大方法。

8. 文件系统用什么组织文件？ (　　)

a) 堆栈；　　b) 指针；　　c) 目录；　　d) 路径。

9. 文件路径名是指什么？ (　　)

a) 文件名和文件扩展名；

b) 一系列的目录文件名和该文件的文件名；

c) 从根目录到该文件所经历的路径中各符号名的集合；

d) 目录文件名和文件名的集合。

10. 如果 I/O 所花费的时间比 CPU 处理时间短得多，则缓冲区的交换效率怎样？

( )

a) 最有效； b) 几乎无效； c) 均衡； d) 以上都不是。

三、简答题 （共 20 分）

1. (6 分)什么是操作系统？它有什么基本特征？

2. (7 分)某进程被唤醒后，立即投入了运行，有人说该系统采用了抢先调度方式，你说对吗？请说明原因。

3. (7 分)比较段式管理和页式管理的特点？

四、问答题 （共 20 分）

1. (10 分)设有 8 个进程 M_1，M_2，…，M_8，它们有如图 13.3 所示的优先关系，试用 P／V 操作实现这些进程间的同步。

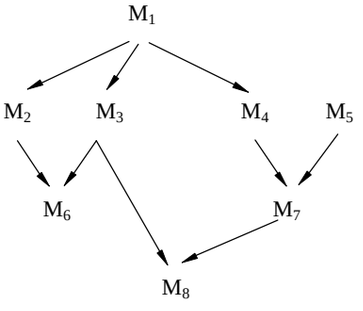


图 13.3

2. (10 分) 一程序在运行过程中依次用到程序空间的第 3，5，4，2，5，3，1，3，2，5，1，3，1，5，2 页，若采用 LRU 算法，则为该程序分配多少个实页最为合理？为什么？

模拟试题IV参考答案

一、填空题

1. 操作系统是计算机系统中的一个 核心控制软件 ，它管理和控制计算机系统中的 各种资源 。

2. 在操作系统中，不可中断执行的操作称为 原子操作 。

3. 并发进程中涉及到相同变量的程序段叫做 临界段 ，对这些程序段要 互斥 执行。

4. 当系统采用资源有序分配方法预防死锁时，它破坏了死锁的必要条件中的 循环等待 条件。

5. 短作业 优先运行的调度算法能够缩短作业的平均等待时间。

6. 在有一个 CPU 和两台外设 d_1 和 d_2 且能够实现抢占式优先数调度算法的多

道程序环境中，同时进入优先级由高到低为 p_1，p_2，p_3 的三个作业，每个作业的处理顺序和使用资源的时间如下。

p_1：d_2(20ms), CPU(10ms), d_1(30ms), CPU(10ms)

p_2：d_1(20ms), CPU(20ms), d_2(40ms)

p_3：CPU(30ms), d_1(20ms)

假设其他辅助操作时间忽略不计，则 p_1，p_2，p_3 每个作业的周转时间分别为 70ms 、 90ms 和 80ms ；CPU 的利用率为 77.8% ，d_1 的利用率为 77.8% 。

7. 在请求页式系统中，LRU 是 最近最少使用 置换算法，LFU 是 最低使用频率 置换算法。

8. 目前认为逻辑文件有两种类型，即 流 式文件与 结构 式文件。

9. 活动头磁盘的访问时间包括 查找时间 、 等待时间 和 传输时间 。

10. 在 Unix 文件管理系统中，为了对磁盘空间的空闲块进行有效的管理，所采用的方法是 成组链接法 。

二、选择题

1. a)	2. d)	3. b)	4. b)	5. b)
6. b)	7. d)	8. c)	9. b)	10. a)

三、简答题

1. 操作系统是配置在计算机硬件上的第一层软件，是对硬件系统的第一次扩充。在计算机系统中占特别重要的地位，其他所有软件都依赖于它的支持，既是硬件的管理者，也是软件的支持者。其基本特征有：并发、共享、虚拟和异步性。

2. 不对，这并不能说明采取了抢先调度方式。

反例：在非抢先调度的系统中，仅有生产者、消费者两个进程，且缓冲池大小为 1，这时两进程中任何一个被唤醒都会立即投入执行。

3. 二者的比较如表 13.7 所示。

表 13.7

段式管理	页式管理
· 逻辑单位	· 物理单位
· 用户可见	· 用户不可见
· 面向使用	· 面向系统
· 长度不定，由用户定	· 长度固定，由系统定
· 段号加位移访问	· 页号加位移访问
· 二维逻辑地址空间	· 一维逻辑地址空间
一般情况下，段长度>页长	

四、问答题

1. semaphore 是信号量数据类型，如图 13.4 所示。

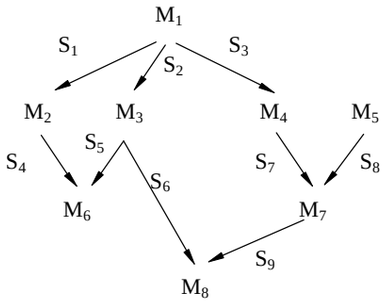


图 13.4

程序如下：

```
VAR S1, S2, S3, S4, S5, S6, S7, S8, S9：
  semaphore ：= 0,0,0,0,0,0,0,0,0;
begin
  begin     M1; V(S1); V(S2); V(S3); end;
  //code of process M1//
  begin     P(S1); M2; V(S4); end;
  //code of process M2//
  begin     P(S2); M3; V(S5); V(S6); end;          //code of process M3//
  begin     P(S3); M4; V(S7); end;                 //code of process M4//
  begin     M5; V(S8); end;                        //code of process M5//
  begin     P(S4); P(S5); M6; end;                 //code of process M6//
  begin     P(S7); P(S8); M7; end;                 //code of process M7//
  begin     P(S6); P(S9); end;                     //code of process M8//
end
```

2. 考虑该调用序列，共有五个不同的页面，若不考虑第一次分配的那一页，则经过 4 次缺页中断后，5 页全被调入，此后不再发生中断，所以分配五个实页，缺页 4 次。

若分配三个实页，则共产生 10 次中断，如下所示：

3	5	4	2	5	3	1	3	2	5	1	3	1	5	2
3	5	4	2	5	3	1	3	2	5	1	3	1	5	2
	3	5	4	2	5	3	1	3	2	5	1	3	1	5
		3	5	4	2	5	5	1	3	2	5	5	3	1
	×	×	×		×	×		×	×	×	×			×

若分配四个实页，则共产生 4 次中断，如下所示：

3	5	4	2	5	3	1	3	2	5	1	3	1	5	2
3	5	4	2	5	3	1	3	2	5	1	3	1	5	2
	3	5	4	2	5	3	1	3	2	5	1	3	1	5
		3	5	4	2	5	5	1	3	2	5	5	3	1
			3	3	4	2	2	5	1	3	2	2	2	3
	×	×	×			×								

由此可以看出，分配四个实页最佳。

模拟试题 V

一、填空题 （每空 1 分，共 30 分）

1. 现代操作系统两个最基本的特征是_____和_____。

2. 死锁产生的四个必要条件是互斥控制、_____、_____、_____。

3. 当处理机空闲时，进程调度程序从_____队列中选取一个进程执行。

4. 如果系统中所有作业是同时到达的，则使作业平均周转时间最短的作业调度算法是_____。

5. 段页式存储管理是将作业分_____，_____内分_____。其分配以_____为单位。在不考虑使用联想存储快表情况下，每条访问内存的指令需要_____次访问内存。其中第_____次是查作业的页表。

6. 文件目录中用_____记录文件的一维地址（逻辑地址），而实际读/写磁盘需用_____、_____及_____三维地址。

7. 为实现 CPU 与外部设备的并行工作，系统引入了_____硬件机制。

8. 为了提高磁盘的读/写速度，在内存中建立了_____。

9. 在 Unix 系统 V 中，如果一个盘块的大小为 1KB，每个盘块占 4B，那么，一个进程要访问偏移量为 263168B 处的数据时，需要经过_____次间址。

二、选择题 （每题 3 分，共 30 分）

1. 操作系统提供给程序员的接口是什么？ （ ）
   a) 进程； b) 系统调用； c) 库函数； d) B 和 C。

2. 建立多进程的主要目的是提高什么的利用率？ （ ）
   a) 文件； b) CPU； c) 内存； d) 外设。

3. 临界区是什么？ （ ）
   a) 一个缓冲区 ； b) 一段共享数据区；
   c) 一段程序； d) 一个互斥资源。

4. 产生死锁的原因是什么？ （ ）
   a) 资源共享； b) 并发执行的进程数太多；
   c) 系统资源不足； d) 进程推进顺序非法。

5. 某系统采用短作业优先的调度算法，现有作业序列：作业 1（提交时间 8.00，

运行时间 1.50）；作业 2（提交时间 8.30，运行时间 0.80）；作业 3（提交时间 9.00，运行时间 0.10）；作业 4（提交时间 9.30，运行时间 0.30）。单位为 h，以十进制计，其平均带权周转时间是多少？ (　　)

a) 4.65；　　b) 3.00；　　c) 5.52；　　d) 12.23。

6. 在请求分页存储管理中，如果所需的页面不在内存，则产生缺页中断，它属于什么中断？ (　　)

a) 硬件故障；　b) I/O；　　c) 外；　　d) 程序中断。

7. 在文件系统中，若采用一级目录结构，则存在的最主要的一个问题是什么？ (　　)

a) 目录表的大小难以确定；

b) 磁盘容量大时，文件检索速度太慢；

c) 用户使用不方便；

d) “重名”问题，即文件命名冲突。

8. 在关于 SPOOLing 的叙述中，描述不正确的是哪些？ (　　)

a) SPOOLing 系统中不需要独占设备；

b) SPOOLing 系统加快了作业的执行速度；

c) SPOOLing 系统使独占设备变成了共享设备；

d) SPOOLing 利用了处理器与通道并行工作的能力。

9. 在 Unix 系统中，用户通过什么读取磁盘文件中的数据？ (　　)

a) 作业申请表；　b) 原语；　　c) 系统调用；　d) 中断。

10. Unix 系统中，把输入／输出设备看做是什么？ (　　)

a) 普通文件；　　b) 目录文件；　c) 索引文件；　d) 特殊文件。

三、简答题 （共 20 分）

1. (7 分) 一台计算机有 8 台磁带机，它们由 n 个进程竞争使用，每个进程可能需要 3 台磁带机。请问：n 为多少时，系统没有死锁危险，并说明其原因。

2. (8 分) 在请求分页存储管理方式中，若采用先进先出（FIFO）页面淘汰算法，会产生一种奇怪的现象：分配给作业的实页越多，进程执行时的缺页率反而升高。试举一例说明这种现象。

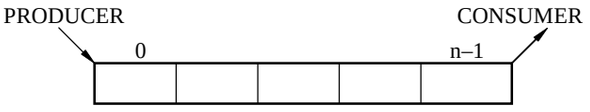
3. (5 分) 何谓虚拟设备？请说明 SPOOLing 系统是如何实现虚拟设备的。

四、问答题 （共 20 分）

1. 以下是生产者-消费者问题（Producer-Consumer Program）的一种解法。

(1) 写出 Producer 进程和 Consumer 进程各自的临界区。

(2) 给出解决临界区互斥问题的一种方法。




```
VAR   counter： 0..n
  buffer： array[1..n] of item;
PRODUCER                                    CONSUMER
  repeat                                      repeat
        …                                           while counter=0 do no-op;
        produce an item in nextp;                   nextc = buffer[out];
        …                                           out ：= (out+1) mod n;
        buffer[in] ：=nextp;                        counter ：= counter – 1;
        Gin ：= (in+1) mod n;                       …
        counter ：= counter +1;                   consumer the item in nextc;
  until false;                                    …
                                                  until false;
```

2. **有三个程序** A、B、C，**它们分别单独运行时的** CPU **和** I/O **占用时间如图** 13.5 **所示。**

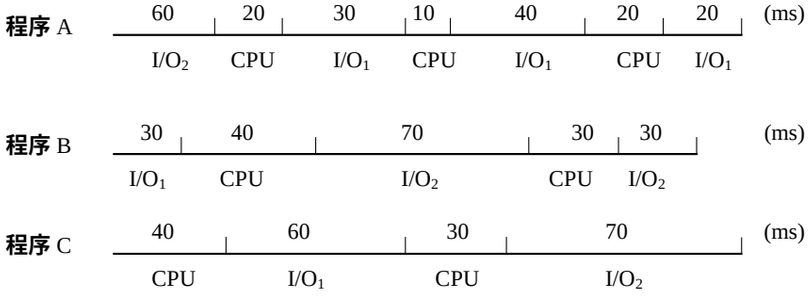


图 13.5

现在考虑三个程序同时开始执行。系统中有一个 CPU **和两台输入/输出设备（**I/O_1 **和** I/O_2**）同时运行。**3 **个程序的优先级为** A **最高，**B **次之，**C **最低，优先级最高的程序可以中断优先级低的程序，但优先级与输入/输出设备无关。请回答下面的问题：**

(1) **最早结束的程序是哪个？**

(2) **最后结束的程序是哪个？**

(3) **三个程序执行到结束分别用了多长时间？**

(4) **计算这段时间的** CPU **利用率（**3 **个程序完全结束为止）。**

模拟试题Ⅴ参考答案

一、填空题

1. 现代操作系统两个最基本的特征是 并发 和 共享 。

2. 死锁产生的四个必要条件是互斥控制、 占用并等待 、 非抢占 、 循环等待 。

3. 当处理机空闲时，进程调度程序从 就绪 队列中选取一个进程执行。

4. 如果系统中所有作业是同时到达的，则使作业平均周转时间最短的作业调度算法是 短作业优先 。

5. 段页式存储管理是将作业分 段 ， 段 内分 页 。其分配以 页 为单位。在不考虑使用联想存储快表情况下，每条访问内存的指令需要 3 次访问内存。其中第 2 次是查作业的页表。

6. 文件目录中用 块号 记录文件的一维地址（逻辑地址），而实际读／写磁盘需用 磁头 、 柱面 及 扇区 三维地址。

7. 为实现 CPU 与外部设备的并行工作，系统引入了 中断 硬件机制。

8. 为了提高磁盘的读写速度，在内存中建立了 磁盘缓冲区 。

9. 在 Unix 系统 V 中，如果一个盘块的大小为 1KB，每个盘块占 4B，那么，一个进程要访问偏移量为 263168B 处的数据时，需要经过 1 次间址。

二、选择题

1. b)　2. b)　3. c)　4. d)　5. b)
6. d)　7. d)　8. d)　9. c)　10. d)

三、简答题

1. $n \leq 3$ 时系统无死锁危险。因为 $n=4$ 时，若每个进程占用两台并等待，则系统死锁，所以 $n<4$。当 $n=3$ 时，最大需求之和 $\sum_{i=1}^{3} \text{Max}_i \leq 9$，最大分配和 $\sum_{i=1}^{3} \text{Allocation}_i = 8$ $\sum_{i=1}^{3} \text{Need}_i \leq \sum_{i=1}^{3} \text{Max}_i - \sum_{i=1}^{3} \text{Allocation}_i = 1$。因此，必有一进程 $\text{Need}_i = 0$。所以系统在 $n \leq 3$ 时不会死锁。

2. 假如某作业有五个虚页，则页面访问踪迹如下表所示（“×”代表缺页中断）：

0	1	2	3	0	1	4	0	1	2	3	4	
0	1	2	3	0	1	4	4	4	2	3	3	
	0	1	2	3	0	1	1	1	4	2	2	
		0	1	2	3	0	0	0	1	4	4	
	×	×	×	×	×	×			×	×		3 个实页，8 次中断
0	1	2	3	3	3	4	0	1	2	3	4	
	0	1	2	2	2	3	4	0	1	2	3	
		0	1	1	1	2	3	4	0	1	2	
			0	0	0	1	2	3	4	0	1	
	×	×	×			×	×	×	×	×	×	4 个实页，9 次中断

注：此答案不唯一，只要例子正确合理即可得分。

3. (1) 虚拟设备就是利用共享设备区模拟的独占设备，起到共享和高速的作用。

(2) 当进程申请独占设备时，系统并非真将设备分配给它，而是在输入/输出井中申请空闲盘块区，缓存其数据，对于进程而言，就好像自己独占设备一样，即把一台独占设备变成若干台对应的逻辑设备。

四、问答题

1. (1)

PRODUCER 临界段

```
buffer[in] ：= nextp;
I ：= (in+1) mod n;
counter ：= counter +1;
```

CONSUMER 临界段

```
nextc = buffer[out];
out ：= (out+1) mod n;
counter ：= counter –1;
```

(2) CONSUMER 有 while 循环条件保证 counter=0 时不再消费，但 PRODUCER 没有，需引入 empty 信号量加以控制，还要引入互斥信号量 mutex：

```
VAR empty： integer ：=n;          （假设初始缓冲区为空）
    mutex： integer ：=1;
PRODUCER                          CONSUMER
  repeat                            repeat
    …                                 while counter=0 do no_op;
    produce an item in nextp;         P(mutex) ;
    …                                 nextc = buffer[out];
    P(empty) ;                        out ：= (out+1) mod n;
    P(mutex) ;                        counter ：= counter –1;
```

```
        buffer(in) ：= nextp;                    V(mutex) ;
        in ：= (in+1) mod n;                     V(empty) ;
        counter ：= counter +1;
        V(mutex) ;
    until false;
```

2. (1) A 程序　(2) C 程序　(3) A 程序：200ms　B 程序：240ms　C 程序：330ms　(4) CPU 利用率为 57.6%

附录

思 考 题

为使报考计算机专业研究生的读者们了解当前各高校计算机专业课程的考试情况，我们收集了一些比较有代表性的高校计算机专业“操作系统原理”课程的考研试题，在此作为思考题列出，以供读者参考。对于部分可以在前面章节中找到答案或类似题目的试题，我们仅给出解答提示。

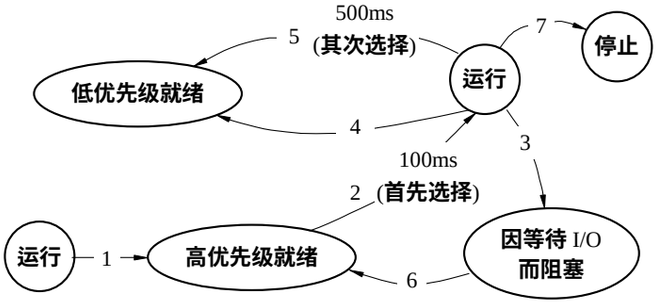
1. 消息缓冲通信技术是一种高级通信机制，由 Hansen 首先提出：

(1) 试叙述高级通信机制与低级通信机制 P、V 原语操作的主要区别；

(2) 请给出消息缓冲机制（有界缓冲）的基本原理；

(3) 消息缓冲机制（有界缓冲）中提供发送原语 Send（receive，a），调用参数 a 表示发送消息的内存区首地址，试设计相应的数据结构，并用 P、V 原语操作实现 send 原语。

2. 某系统的进程状态如附图 1 所示。



附图 1

(1) 说明一个进程发生变迁 3、4、6 的原因。

(2) 根据此进程状态图，说明该系统的 CPU 调度策略和调度效果。

3. 设系统中有 3 种类型的资源（A，B，C）和 5 个进程（P_1，P_2，P_3，P_4，P_5），A 资源的数量为 17，B 资源的数量为 5，C 资源的数量为 20。在 T_0 时刻系统状态如附表 1 所示。若系统采用银行家算法（Banker’s algorithm）实施死锁避免策略。问：

(1) T_0 时刻是否为安全状态？若是，请给出安全序列。

(2) 在 T_0 时刻若进程 P_2 请求资源（0，3，4），是否能实施资源分配？为什么？

(3) 在(2)基础上，若进程已请求资源（2，0，1），是否能实现资源分配？为什么？

(4) 在(3)基础上，若进程 P_1 请求资源（0，2，0），是否能实施资源分配？为什么？

附表 1

	最大资源需求量			已分配资源数		
	A	B	C	A	B	C
P_1	5	5	9	2	1	2
P_2	5	3	6	4	0	2
P_3	4	0	11	4	0	5
P_4	4	2	5	2	0	4
P_5	4	2	4	3	1	4
	A		B		C	
剩余资源数	2		3		3	

4. 在消息传递通信方式下，

(1) 发送进程和接收进程在通信过程中可以采取哪三种同步方式？

(2) 试以下面给出的发送进程和接收进程（将接收到的数据存入 S）为例，说明当接收进程执行到标号为 L_2 的语句时，采用这三种同步方式，X 的值可能各是多少？

发送进程 P：	接收进程 Q：
M=10；	S=100;
L1:send M to Q;	L1:receive S from P;
L2:M=20;	L2:X=S+1;
Goto L1;	

5. 设系统中仅有一类数量 M 的独占型资源，系统中 N 个进程竞争该类资源，其中，各进程对该类资源的最大需求量为 W，当 M、N、W 分别取下列值时：

(1) M = 2，N = 2，W = 1　　(2) M = 3，N = 2，W = 2

(3) M = 3，N = 2，W = 3　　(4) M = 5，N = 3，W = 2

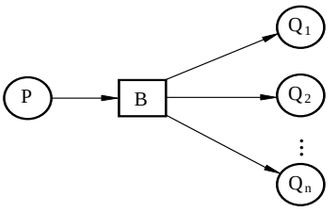
(5) M = 6，N = 3，W = 3

试判断哪些情况会发生死锁，为什么？

6. 何谓进程的同步与互斥？解决进程同步与互斥问题的机制有哪些？试采用其中一种机制，分别举一简例说明如何解决进程同步和进程互斥问题。

7. 试利用 P、V 原语，形式化或非形式化地描述下列进程的动作序列：进程

P 使用缓冲区 B 向 IN 进程 Q_1，Q_2，…，Q_M 发送消息（如附图 2 所示），要求每当 P 向 B 发消息时，只有当所有的进程 Q_i（i=1，2，…，M）都读取这条消息后，P 才可向 B 发送新的消息。



附图 2

8. 用 P、V 操作和信号量解决进程之间的同步互斥问题：有 n 个进程将字符读入到一个容量为 80 的缓冲区中（n > l），当缓冲区满后，由另一个进程 P_b 负责一次取走这 80 个字符。这种过程循环往返，请写出几个读入进程（P_1，P_2，…，P_n）和 P_b 的动作序列（可用文字或表达式来描述动作序列，设 P_1 每次读一个字符到缓冲区中）。

9. 现代操作系统一般都提供多进程（或称多任务）运行环境，试回答以下问题：

(1) 为支持多进程的并发执行，系统必须建立哪些关于进程的数据结构？

(2) 为支持进程状态的变迁，系统至少应提供哪些进程控制原语？

(3) 在执行每一个进程控制原语时，进程状态发生什么变化？相应的数据结构发生什么变化？

10. 已知 3 个并发进程 W、C_0、C_1 共享一位（l b）缓冲 B，W 不断向 B 写 0 或 1，C_0 对 0 计数，C_1 对 1 计数。仅当 B 中数字被 C_0 或 C_1 读出计数后，W 才能再写。请用 wait 及 signal 操作实现三进程的同步：

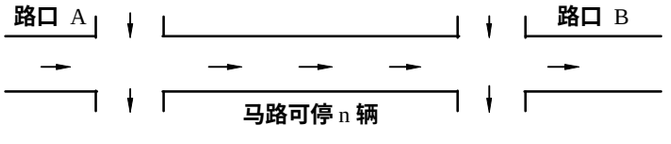
(1) 设置信号灯，给出信号灯初值；

(2) 画出各进程的流程。

11. 今有 3 个并发进程 R、M、P，它们共享一个可循环使用的缓冲区 B，缓冲区 B 共有 N 个单元。进程 R 负责从输入设备读信息，每读一个字符后，把它存放在缓冲区 B 的一个单元里；进程 M 负责处理读入的字符，若发现读入的字符之间有空格，则把它改为“，”；进程 P 负责把处理后的字符取出并打印输出。缓冲区单元中的字符被进程 P 取出后，又可用来存放下一次读入的字符。请以 P、V 操作为同步机制写出能使它们正确并发执行的程序。

12. 假定有一个信箱可存放 N 封信，当信箱不满时，发信者可把信件送入信箱；当信箱中有信时，收信者可从信箱中取信。用指针 P、K 分别表示可存信和取信的位置，请用管程（Monitor）来管理这个信箱，使发信者和收信者能正确工作。

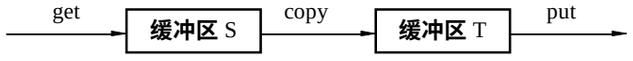
13. 对附图 3 所示的交通管理例子（各方向上的汽车是单行、直线行驶），试



附图 3

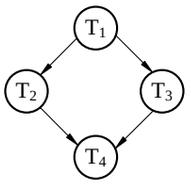
用 P、V 操作实现各方向上汽车行驶的同步。

14. 设有进程 A、B、C，分别调用过程 get、copy 和 put 对缓冲区 S 和 T 进行操作。其中，get 负责把数据块输入缓冲区 S，copy 负责从缓冲区 S 中提取数据块并复制到缓冲区 T 中，put 负责从缓冲区 T 中取出信息打印（见附图 4）。试描述 get、copy 和 put 的操作过程。



附图 4

15. 设有一个作业由 4 个进程组成，这 4 个进程必须按附图 5 所示的次序运行，试用 P、V 操作表达这 4 个进程的同步关系。



附图 5

16. 设有一个应用软件有两个线程（可理解为能并发执行的两个函数，且它们能通过变量名，对同一个全局变量进行操作），其中，一个线程名为 read_data，每隔 10 ms 从外部读入一对实数存入一个全局缓冲区，另一个线程名为 display_data，它每隔 10ms 从前述的全局缓冲区中取出一对由 read_data 读入的数据。假设操作系统不能保证非常精确地按设定的时间间隔进行线程调度。请设计全局缓冲区的结构，并给出尽量好地解决这一问题的算法。

17. 下表所示作业中，每个作业在单道下运行时等待 I/O 时间为 0，问：采用简单的时间片轮转法在四道下运行，用户的满意程度比单道下提高还是变坏？为什么？

作业号	1	2	3	4
提交时间/h	9. 0	9. 0	9. 0	9. 0
运行时间/h	5. 0	5. 0	5. 0	5. 0

18. 假定要在一台处理机上执行如下表所示的作业。且假定这些作业在时刻 O 以 l、2、3、4、5 的顺序到达。

作业	执行时间	优先级
1	10	3
2	1	1
3	2	2
4	1	4
5	5	2

(1) 给出分别使用 FCFS（先来先服务算法）、RR（时间片轮转算法，时间片= 1）、SJF（最短作业优先算法）以及非抢占式优先调度算法（优先级数字越小，优先级越高）时，这些作业的执行顺序；

(2) 针对上述每种设计算法，分别给出平均周转时间和平均带权周转时间。

19. 纯分页系统和请求式分页系统的主要差别是什么？假定在一个请求页式存储管理系统中，某作业 J 所设计的页面依次为：

3，2，1，4，4，5，3，4，3，2，1，5

并已知主存中有 3 个可供作业 J 使用的空白存储块（块的大小与页面大小相同），试说明 FIFO 和 LRU 两种算法进行页面置换时，缺页中断的次数各是多少？

20. 某请求页式管理系统，用户编程空间有 40 个页面。每一个 200H 字节。假定某时刻用户页表中虚页号和物理块号对照表如下：

虚页号	0	2	5	17	20
物理块号	5	20	8	14	36

求虚地址 0A3CH、223CH 分别对应的物理地址。

21. 设正在处理器上执行的一个进程的页表如下：

虚页号	状态位	访问位	修改位	物理块号
0	1	1	0	4
1	1	1	1	7
2	0	0	0	—
3	1	0	0	2
4	0	0	0	—
5	1	0	1	0

注：当某页被访问时，其访问位置为 l。

表中的虚页号和物理块号是十进制数，起始页号（块号）均为 0。所有的地址均是存储器字节地址。页的大小为 1024B。

(1) 详述在设有快表的请求分页存储管理系统中，一个虚地址转换成物理内存地址的过程。

(2) 下面虚地址对应于什么物理地址：5499，2221。

22. 关于分页系统，回答下列问题：

(1) 在页表中，哪些数据项是为实现请调页面设置的？哪些数据项是为实现置换一页而设置的？

(2) 设其系统为每个作业进程分配 3 个内存块，某作业进程在运行中访问页

面的轨迹为 1、4、3、1、6、8、1，且每一页都是按请求装入的。问：在先进先出页面置换算法（FIFO）和最久未使用页面置换算法（LRU）下，产生的缺页中断次数各是多少？

要求：画出必要的数据结构，并说明每次淘汰的页面。

(3) 在什么情况下，上述两种页面淘汰算法执行的效果是一样的？为什么？

23. 假定某操作系统存储器采用页式存储管理，页的大小为 64B。假定一进程的代码段的长度为 702B，页表如附图 6 所示。该进程在联想存储器中的各表项如附图 7 所示。

页号	页帧号	页号	页帧号
0	F0	6	F6
1	F1	7	F7
2	F2	8	F8
3	F3	9	F9
4	F4	10	F10
5	F5		

附图 6

页号	页帧号
0	F0
1	F1
2	F2
3	F3
4	F4

附图 7

现进程有如下的访问序列：其逻辑地址为八进制的 105、217、567、1120、2500。试问：给定的这些地址能否进行转换？若能，请说明地址转换过程及相应的物理地址；若不能，则说明理由。

24. 考虑下页的访问串：

1，2，3，4，2，1，5，6，2，1，3，7，6，3. 2，1，2，3，6

假定有 4，5，6 三个页块，应用下面的页面替换算法，各会出现多少次缺页中断？

(1) LRU（最近最久未使用算法）；

(2) FIFO（先进先出算法）；

(3) Optimal（最佳算法）。

注意：所给定的页块初始均为空，因此，首次访问一页时就会发生缺页中断。

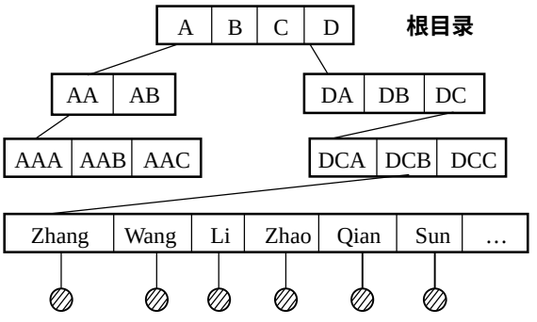
25. 在实现文件系统时为加快文件目录的检索速度，可利用“文件控制块分解法”。假设目录文件存放在磁盘上，每个盘块 512B，文件控制块占 64B，其中文件名占 8B。通常将文件控制块分解成两部分，第一部分占 10B（包括文件名和文件内部号），第二部分占 56B（包括文件内部号和文件其他描述信息）。

(1) 假设某一目录文件共有 254 个文件控制块，试分别给出采用分解法前和分解法后，查找该目录文件某一文件控制块的平均访问磁盘次数。

(2) 一般地，若目录文件分解前占有 n 个盘块，分解后改用 m 个盘块存放文件名和文件内部号部分，请给出访问磁盘次数减少的条件。

26. 文件系统采用多重结构搜索文件内容。设块长为 512B、每个块号长 3B，如果不考虑逻辑块号在物理块中所占的位置，分别求二级索引和三级索引时可寻址的文件最大长度。

27. 文件系统属性目录如附图 8 所示，已知每个目录项占用 256B，磁盘的一块为 512B。



附图 8

(1) 查询文件 Wang 的路径是什么？

(2) 系统需读取几个文件后才能查到 Wang？

(3) 每个文件至少读取几块盘？

(4) 给出一个加速文件访问速度的目录项结构。

28. 假定磁带的记录密度为每英寸 800 个字符，每个逻辑记录长为 160 字符，块间隙为 0.6 英寸，今有 1000 个逻辑记录需要存储，分别计算：

(1) 当进行不成组操作和以 5 个逻辑记录为一组的成组操作时，磁带介质的利用率。

(2) 物理记录至少为多大时，才不致浪费超过 50%的磁带存储空间？

29. 有如附图 9 所示的文件目录结构，问

(1) 可否进行下列操作，为什么？

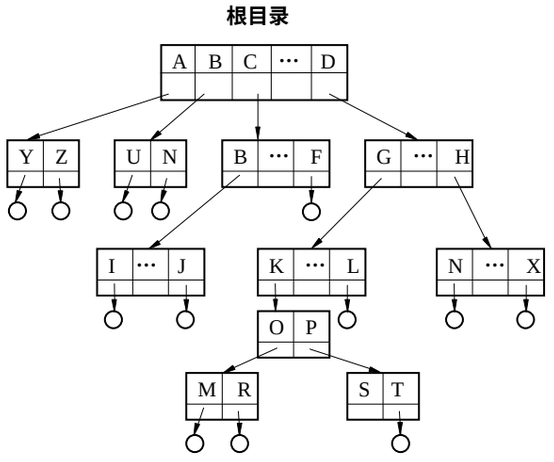
① 在目录 D 中建立一个文件，取名为 A;

② 将目录 C 改名为 A。

(2) 若 E 和 G 是两个用户各自的目录，那么，

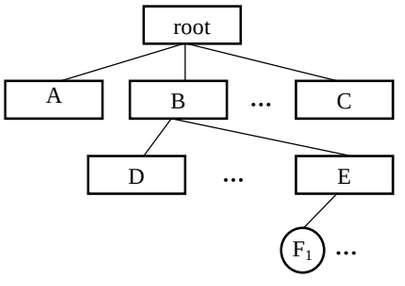
① 若目录 E 的用户要共享文件 M，如何实现？

② 在一段时间内，使用目录 G 的用户主要使用文件 G 和 T，应如何处理？其目的是什么？



附图 9

30. **一个文件系统目录结构如附图** 10 **所示。文件采用的物理结构是串联结构，文件** F_1 **由** 500 **个逻辑记录组成，每个磁盘块可存放** 20 **个逻辑记录。现在欲读取** F_1 **中的第** 406 **号记录，若文件系统的根目录现已存在于内存，则最少需读多少个磁盘块，才能取出** F_1 **的第** 406 **号记录。**



附图 10

31. **某移动臂磁盘的柱面由外向里顺序编号，假定当前磁头停在** 100 **号柱面且移动臂方向是向里的，现有如下表所示的请求序列在等待访问磁盘：**

请求次序	1	2	3	4	5	6	7	8	9	10
柱面号	190	10	160	80	90	125	30	20	140	25

(1) **写出分别采用“最短查找时间优先算法”和“电梯调度算法”时，实际处理上述请求的次序。**

(2) **比较上述两种算法，就移动臂所花的时间（忽略移动臂改向时间）而言，哪种算法更合适？简要说明之。**

32. **磁盘系统调度中，采用** SCAN（扫描）**调度算法为任务队列** 67、65、124、14、122、37、183、98 **服务。试计算服务结束时，磁头总共移动了几个磁道？假设磁头总在第** 0 **道和第** 199 **道之间移动；开始服务时，磁头刚从** 60 **移到** 67。

33. 设某移动头磁盘有 200 道，编号为 0～199，磁头当前正处在 130 道上，且正向 0 磁道方向移动，对于如下访问磁盘请求序列（磁道号）：

70，120，80，160，60，150

求在 FCFS、SSTF（最短寻道时间优先）及 SCAN 调度算法下的磁头移动顺序及移动总量(以磁道数计)。

34. 假设有 4 个记录 A、B、C、D 存放在磁盘的某个磁道上，该磁道被划分为 4 块，每块存放一个记录，安排如下表所示：

块号	1	2	3	4
记录号	A	B	C	D

现在要顺序处理这些记录，如果磁盘旋转速度为 20ms 转一周，则处理程序每读出一个记录后花 5 ms 进行处理。试问：

(1) 处理完这 4 个记录的总时间是多少？

(2) 为了缩短处理时间应进行优化分布，应如何安排这些记录？并计算处理的时间。

35. 当前磁带读/写位于柱面号 20，此时，有多个磁盘请求，以下列柱面号顺序送至磁盘驱动器：10, 22, 20, 2, 40, 6, 38。寻道（trade）时，移动一个柱面需 6 ms，试按下列三种算法计算所需寻道时间（柱面移动顺序及所需总寻道时间；忽略到达指定柱面后所需寻道时间）。

(1) 先到先服务；

(2) 下一个最邻近柱面；

(3) 电梯算法（当前状态：向上）。

36. 在文件系统中对磁盘空间可采用连续分配方案，该方案类似于内存分区分配技术。我们注意到，辅存设备的碎片问题可以通过整理磁盘命令而消失，一般的磁盘并没有重定位寄存器，那么，如何对文件进行重定位呢？

思考题参考答案

1. (1) 主要区别在于消息传递方式具备数据传输的能力，而 P、V 原语仅是通过信号量这种简单的数据结构进行有限的通讯。

(2) 在这种机制中，缓冲区的大小是有限的，有一个公用的变量表示缓冲区的大小。当缓冲区满时，发送原语将导致进程阻塞；当缓冲区空时，接受原语将导致进程阻塞。

(3) 假设缓冲区大小为 n，则相应的数据结构如下：

```
type msg=…;
var Mailbox：array[1..n] of msg;
    in:integer=1;
    emptyl:semaphore=n;
    full:semaphore=0;
    mutex:semaphore=1;
Send(receiver,a)
    Begin
        P(empty);
        P(mutex);
        Capacity:=capacity–1;
        Mailbox[in]:=a;
        V(mutex);
        V(full)
    End;
```

2. (1) 变迁 3 的原因是进程要等待 I/O；变迁 4 的原因是进程运行完时间片后还未完成，被调度到低优先级队列；变迁 6 的原因是进程等待的 I/O 已经完成。

(2) 该状态转换图说明该系统使用的是带有优先级的时间片轮转调度策略，其调度效果的描述可参考本书第六章的相关内容。

3. 此题可按照“银行家算法”来判断系统状态是否安全。详细信息可以参考本书第四章的知识要点和例 4-27。

4. (1) 这三种同步方式为：①阻塞发送，阻塞接收；②无阻塞发送，阻塞接收；③无阻塞发送，无阻塞接收。

(2) 在方式 1 中，X 的值是 11；在方式 2 中，X 的值是空值（接受进程还没有执行标号 L2 的语句）或是 11；在方式 3 中，X 的值是空值或 11 或 101（接收进程执行 L2 语句时，是否接收到消息也会对 X 的值产生影响）。

5. 利用“银行家算法”进行解答。

6. 请参考本书第三章的知识要点。

7. 此题要注意只有当所有 Q 进程都读取完消息后，P 进程才能继续发送消息的要求，所以设置 Q_1 到 Q_n 共 n 个信号量，初值为 1，还有一个互斥信号量 mutex，初值也为 1。这两个进程的实现如下所示：

```
P：                          Qi：
  Repeat                       Repeat
    P(Q1);                       P(mutex);
```

```
P(Q2);                    Read(msg,B);
….                        V(mutex);
P(Qn);                    V(Qi);
P(mutex);               Forever
Put(msg,B);
V(mutex);
Forever
```

8. 有 n 个进程向一个缓冲区读入字符，他们之间要互斥执行，并且当缓冲区满时，要停止读操作，因此，设置信号量 stopread，初值为 1。缓冲区满后还要通知进程 P_b 取走数据，因此，设置信号量 awake，初值为 0。互斥信号量为 mutex，初值为 1。缓冲区 buf 是一个可以存放 n 个字符的数组，count 是一个计数器，初值为 0。进程的实现如下，其中 P_i 代表 P_1 到 P_n 的任何一个。

P_i：

```
Repeat
  P(stopread);
  P(mutex);
  Read(char)
  Put(char,buf[count])
  count:=count+1;
  if count>80 then V(awake)
     else V(stopread);
  V(mutex);
Forever
```

P_b：

```
Repeat
  P(awake);
  P(mutex);
  Getchar(buf);
  count:=0;
  V(stopread);
  V(mutex);
Forever
```

9. (1) 参考本书第二章知识要点中的进程描述部分。

(2) 文系统应该提供如下一组原语：创建进程、删除进程、挂起进程、激活进程、睡眠和唤醒共六个。

(3) 参考本书第二章知识要点中的进程控制部分及课本[1]中表 2.7 及表 2.8 的内容。

10. 此题与题 7 是类似的题目，可参考题 7 的答案。

11. 这是生产者/消费者问题的一种扩展，其中，M 兼有生产者和消费者的双重身份。要设置四个信号量：empty_rm 初值为 n，表明有多少个空位置可供 R 使用；full_rm 初值为 0，表明还有多少个字符可供 M 处理；full_mp 初值为 0，表明有多少个已处理的字符可供 P 打印；mutex 是互斥信号量，初值为 1。另外缓冲区定义为：B:array[0..n–1] of char；和三个位置指针：in,m,out:integer 供 R、

M、P 使用。进程实现如下：

R：

```
Repeat
  Read(char);
  P(empty_rm);
  P(mutex);
  B[in]:=char;
  in:=(in+1) mod n
  V(mutex);
  V(full_rm);
 Forever
```

M：

```
Repeat
  P(full_rm);
  P(mutex);
  if B[m]=空格 then
   B[m]:=',';
  m:=(m+1) mod n
  V(mutex);
  V(empty_rm);
  V(full_mp);
Forever
```

P：

```
Repeat
  P(full_mp);
  P(mutex);
  print(B[out]);
   out:=(out+1) mod n
  V(mutex);
Forever
```

12. 这是一个用管程解决生产者/消费者的问题，可以参考本书第三章并发控制的相关内容或课本[1]中的讲述。

13. 路口 A 和路口 B 可以被看成为一种临界资源，分别为其设定互斥信号量 crossA 和 crossB，它们的初值设定为 1。因为中间一段马路可以停靠 n 辆车子，所以，设置资源信号量 length，初值设定为 n。

A 路口方向向下的车子：

```
P(crossA);
 通过路口 A;
V(crossA);
```

B 路口方向向下的车子:

```
P(crossB);
 通过路口 B;
V(crossB);
```

方向向右的车子：

```
P(length);
P(crossA);
 通过路口 A;
V(crossA);
 在马路中行进;
P(crossB);
 通过路口 B;
V(crossB);
V(length);
```

14. 此题与题 11 不同，主要区别在于 get、copy 和 put 三者不共享同一个缓冲区，所以可以用两个生产者/消费者问题的解决方案叠加成本题的解答。假设缓冲区 S 的大小为 n，缓冲区 T 的大小为 m。为 S 的同步和互斥设定三个信号量：empty_s 初值为 n；full_s 初值为 0；mutex_s 初值为 1。为 T 的同步和互斥设定三个信号量：empty_t 初值为 n；full_t 初值为 0；mutex_t 初值为 1。并为这两个缓冲区设定位置指针 sin、sout 和 tin、tout，它们的初值都为 0。三个过程的实现如下：

get：

```
Repeat
  Getdata(block);
  P(empty_s);
  P(mutex_s);
  Writeblock(S,sin);
  sin:=(sin+1) mod n
  V(mutex_s);
  V(full_s);
Forever
```

copy：

```
Repeat
  P(full_s);
  P(mutex_s);
  Readblock(S,sout);
  sout:=(sout+1) mod n
  V(mutex_s);
  V(empty_s);
  P(empty_t);
  P(mutex_t);
  Writeblock(T,tin);
  tin:=(tin+1) mod n;
  V(mutex_t);
  V(full_t);
Forever
```

put：

```
Repeat
  P(full_t);
  P(mutex_t);
  Readblock(T,sout);
  sout:=(sout+1) mod n;
  V(mutex_t);
  V(empty_t);
  Print(block);
Forever
```

15. **参考本书第三章并发控制中有关用** P/V **操作实现先序关系的讲述和例** 3-30 **的解答。**

16. **由于系统仅仅是不能精确地按照时间间隔进行调度，也就是这两个进程不能精确地同步，但是，相差又不大，因此，缓冲区的大小设计为可以容纳读/写线程两次读或写操作的数据即可。缓冲区和信号量定义如下：**

```
buf: array[0..3] of float;
empty: semaphore:=2;
full: semaphore:=0;
mutex: semaphore:=1;
in,out: integer:=0;
```

两线程的实现如下：

```
read_data:
    Repeat
      P(empty);
      P(mutex);
      Read(afloat);
      buf[in]:=afloat;
      in:=(in+1) mod 4;
      Read(afloat);
      buf[in]:=afloat;
```

```
display_data:
    Repeat
      P(full);
      P(mutex);
      afloat:= buf[out];
      display(afloat);
      out:=(out+1) mod 4;
      afloat:= buf[out];
      display(afloat);
```

in:=(in+1) mod 4;	out:=(out+1) mod 4;
V(mutex);	V(mutex);
V(full);	V(empty);
Forever	Forever

17. 分别根据先来先服务方式和时间片轮转方式计算出平均周转时间来进行比较，发现采用简单时间片轮转方式在四道下运行的平均周转时间变长了，因此，用户的满意程度会变坏。

18. 此题没有什么技巧和难点，仅需要按照相应的调度算法进行计算。相关内容可以参考本书的第六章处理机调度中的讲述。

19. 在本书的第五章内存管理的例题中有类似的题目，仅是页面访问序列和参数的不同，读者可以参照例 5-18、例 5-20 和例 5-25 的答案给出本题的解答。

20. 由于每个页面有 200H 个字节，所以页内偏移地址是 9 位二进制位。把虚地址 0A3CH 转换成二进制，后 9 位是页内偏移地址，前面是虚页号，该地址的虚页号是 5，对应物理块号 8 号，转化为物理地址为 103CH。同理，虚地址 223CH 对应的物理地址是 1C3CH。

21. (1) 参考本书第五章内存管理中的知识要点部分。

(2) 虚地址 5499 对应物理地址 0499；虚地址 2221 的页在外存中，需调入该页才能进行物理地址转换。

22. 参照第五章内存管理的知识要点的讲述和例 5-18、例 5-20 和例 5-25 的解答。

23. 注意：不要被联想存储器及其内容所迷惑，它只是用来提高系统的逻辑地址到物理地址的转化速度，它的内容是页表的一部分。此题中的前四个逻辑地址可以转换，读者可以按照分页方式中逻辑地址到物理地址的转换规则进行转换。第四个地址 2500 不能转换，其原因是该逻辑地址越界了，这段代码总长为 702B，不会有八进制的 2500 的逻辑地址。

24. 本书的第五章内存管理的例 5-18、例 5-20 和例 5-25 与本题类似，仅是页面访问序列和参数不同，读者可以参照那些题目的答案给出本题的解答。

25. 利用“文件控制块分解法”加快文件目录的检索速度，其原理是减少查找文件内部号而产生的访问磁盘次数。因为在进行查找文件内部号的过程中，不再需要把文件控制块的所有内容都读入，所以在查找过程中所需读入的存储块减少（即减少了访问磁盘的次数）。但是，采用这种方法访问文件，当找到匹配的文件控制块后，还需要进行下一次磁盘访问，才能读出全部的文件控制块信息。这就是为何采用这种方法在一定条件下并不能减少访问磁盘次数的原因。

(1) 采用分解法前，查找该目录文件的某一个文件控制块的平均访问磁盘次数为

$$64\times(254\div2)\div512=16$$

采用分解法后，查找该目录文件的某一个文件控制块的平均访问磁盘次数为

$$10\times(254\div2)\div512+1=4$$

(2) 访问磁盘次数减少的条件为 $m<n-2$。

26. 二级索引文件的最大长度为 87040B；

三级索引文件的最大长度为 14796800B。

27. (1) 查询文件 Wang 的路径是：D/DC/DCB/Wang。

(2) 因为目录本身也是一个特殊的文件，所以系统要读取 3 个文件才能查到 Wang。

(3) 根据题意，每个盘块仅能装下两个目录项，所以读取根目录中的 D 目录项要读两块盘，读 DC 目录项要读两块盘，读目录项 DCB 要读一块盘，读取文件控制块要读一块盘，读文件本身的信息至少要读一块盘，共 7 块盘。

(4) 可采用题 25 中的“文件控制块分解法”中所述目录项结构来提高文件的访问速度。

28. (1) 当进行不成组操作时，

160 字符÷800 字符＝0.2 英寸

利用率＝0.2 英寸÷（0.2 英寸+ 0.6 英寸）＝25%

当进行成组操作时，

(160 字符×5)÷800 字符＝1 英寸

1000÷5＝200 个块(每个块 5 个逻辑记录)

6×200＝120 英寸(间隙)

利用率＝200 英寸÷（200 英寸+ 120 英寸）＝62%

(2) 物理记录大于等于 3 时，才不致浪费超过 50%的存储空间。

29. (1) ① 由于目录 D 中没有已命名为 A 的文件，因此，在目录 D 中，可以建立一个取名为 A 的文件。

② 因为在文件系统的根目录下已经存在一个取名为 A 的目录，所以根目录下的目录 C 不能改名为 A。

(2) ① 用户 E 要共享文件 Q，需要用户 E 有访问文件 Q 的权限。在访问权限许可的情况下，用户 E 可通过相应的路径来访问文件 Q，即用户 E 通过自己的主目录 E 找到其父目录 C，再访问到目录 C 的父目录根目录，然后依次通过目录 D、目录 G、目录 K 和目录 O 访问到文件 Q。若用户 E 当前目录为 E，则访问路径为：../../D/G/K/O/Q，其中“..”代表一个目录的父目录。

② 用户 G 需要依次访问目录 K 和目录 P，才能访问到文件 S 即文件 T。为了提高访问速度，可以在目录 G 下建立两个链接文件，分别链接到文件 S 和文件 T 上。这样，用户 G 就可以直接访问这两个文件了。

③ 用户 E 可以通过修改文件 I 的存取控制表来对文件 I 加以保护，不让别的用户使用。具体的办法是：在文件 I 的存取控制表中，只留下用户 E 的访问权限，其他用户对该文件无操作权限，从而达到不让其他用户访问的目的。

30. 由于根目录已经存在于内存之中，所以不用读盘访问根目录。访问目录 B 要读 1 块盘，访问目录 E 要读 1 块盘，访问文件的控制块要读 1 块盘，读文件的 406 号记录要读 21 块盘，所以共访问 24 块盘。

31. (1) 该问的答案省略，读者仅需要按照题目中所述的两个算法对请求进行处理，即可得到实际处理的请求次序。

(2) 根据实际处理请求的次序，计算移动磁臂的时间（用磁臂移过磁道的总数表示，即实际处理的相邻的请求的柱面号差值的总和），时间少的算法更适合。

32. 按照 SCAN 算法对请求进行调度，产生实际处理请求的队列，把相邻两个请求的磁道号的差加起来，就得到磁头总共移动磁道数的总量。

33. 此题的答案省略，读者仅需要按照题目中所述的两个算法对请求进行处理，即可得到实际处理的请求次序。然后，按照实际处理请求的队列，把相邻两个请求的磁道号的差加起来，就得到磁头总共移动磁道数的总量。

34. (1) 按照题目中表格所示的安排方案，处理程序每读一个记录然后处理都要花费 10ms 的时间，处理完后要等磁盘转 3/4 圈后，才能读下一个记录，因此，要等待 15ms 的时间，所以读完并处理完 4 个记录总共用了 25×3+ 10＝85ms 的时间。

(2) 应该让程序读出并处理完一个记录后，紧接着读下一个记录，因此，把记录在磁道上交错放置，1 号块放 A，2 号块放 C，3 号块放 B，4 号块放 D。按照这种放置方式，处理程序处理完 A 时，磁头正好在 B 记录的位置，可以紧接着读 B。处理完 B 后，要等磁盘转过记录 A 后，才能读记录 C，因此多等待 5ms。所以这种方式下处理程序所花费的总时间为：10+15+10+10=45ms。

35. 此题可按照题 31、题 32 和题 33 的提示进行解答。

(1) 先来先服务：(10+12+20+38+14+32)×6=756ms

(2) 下一个最邻近：(2+2+10+4+4+36+2)×6=360ms

(3) 电梯算法：(2+16+2+20+10+4+4)×6=348ms

36. 因为在管理文件时，有一个称之为“文件控制块”的结构供系统使用。该结构内记录了文件起始的物理位置，当我们对磁盘进行碎片整理时，如果移动了某个文件，就需要把文件读入内存，再重新写入磁盘，此时，文件控制块中文件的物理起始位置已变为存储文件的新位置。文件就是利用这个记录进行重定位的。

参 考 文 献

1 **何炎祥主编. 操作系统原理. 武汉：华中科技大学出版社**, 2001

2 Agarwal A, Horowitz M, Hennessy J. An Analytical Cache Model. ACM Transactions on Computer Systems, May 1988

3 Ananda A, Tay B, Koh E. A Survey of Asynchronous Remote Procedure Calls. Operating Systems Review, April 1992

4 Andleigh P Unix System Architecture. Englewood Cliffs, NJ：Prentice Hall, 1990

5 Andrianoff S. A Module on Distributed Systems for the Operating System Course. Procedings, Twenty-First SIGCSE Technical Symposium on Computer Science Education. SIGCSE Bulletin, February 1990

6 Bach,M. The Design of the UNIX Operating System. Englewood Cliffs, NJ：Prentice Hall, 1986

7 Bacon J. Concurrent Systems. Reading, MA：Addison-Wesley, 1994

8 BarBosa V. Strategies for the Prevention of Communication Deadlocks in Distributed Parallel Programs. IEEE Transactions on Software Engineering, November 1990

9 Bic L, Shaw A. The Logical Design of Operating Systems. 2nd ed. Englewood Cliffs, NJ：Prentice Hall, 1988

10 Brown R, Denning P, Tichy W. Advanced Operating Systems. Computer, October 1984

11 Carr R. Virtual Memory Management. Ann Arbor,MI：UMI Research Press,1984

12 Casavant T, Singhal M. Distributed Computing Systems. Los Alamos, CA：IEEE Computer Sicuety Press, 1994

13 Davis W. Operating Systems：A Systematic View. 3rd ed. Reading, MA：Addison-Wesley, 1987

14 Grosshans D. File Systems：Design and Implementation. Englewood Cliffs, NJ：Prentice Hall, 1986

15 Horner D. Operating Systems：Concepts and Applications. Glenview,IL：Scott, Foresman, 1989

16 Johnson R. MVS Concepts and Facilities. New York：McGraw-Hill, 1980

17 Krakowiak S, Beeson D. Principles of Operating Systems. Cambridge, MA：MIT Press, 1988

18 Lister A, Eager R. Fundamentals of Operating Systems.4th ed. London：Macmillan Education Ltd, 1996

19 Nutt G. Centralized and Distributed Operating Systems. Englewood Cliffs, NJ：Prentice Hall, 1994

20 Raynal M. Algorithms for Mutual Exclusion. Cambridge, MA：MIT Press, 1986

21 Silberschatz A, Galvin P. Operating System Concepts. Readings, MA：Addison Wesley, 1996

22 Stankovicm J, Ramamritham K, et al. Advances in Real-Time Systems. Los Alamitos, CA：IEEE Computer Society Press, 1993

23 Tanenbaum A, Rebesse R. Distributed Operating Systems. Computing Surveys, December, 1995

24 Turner R. Operating Systems：Design and Implementations. New York：Macmillan, 1996

25 Willam S. Operating Systems.2nd ed . New Jersey：Prentice-Hall International Inc., 1997

26 **何炎祥等. 计算机操作系统原理及其习题解答. 北京：海洋出版社**, 1993

27 **何炎祥等. 分布式操作系统设计. 北京：海洋出版社**, 1993

28 **何炎祥等. 并行程序设计方法. 北京：学苑出版社**, 1995

29 **何炎祥等. 高级操作系统. 北京：科学出版社**, 1999

30 **孟庆昌. 操作系统教程. 西安：西安电子科技大学出版社**, 1993

参 考 文 献

1 Agarwal A, Horowitz M, Hennessy J. An Analytical Cache Model. ACM Transactions on Computer Systems, May 1988

2 Ananda A, Tay B, Koh E. A Survey of Asynchronous Remote Procedure Calls. Operating Systems Review, April 1992

3 Andleigh P Unix System Architecture. Englewood Cliffs, NJ: Prentice Hall, 1990

4 Andrianoff S. A Module on Distributed Systems for the Operating System Course. Procedings, Twenty-First SIGCSE Technical Symposium on Computer Science Education. SIGCSE Bulletin, February 1990

5 Bach,M. The Design of the UNIX Operating System. Englewood Cliffs, NJ: Prentice Hall, 1986

6 Bacon J. Concurrent Systems. Reading, MA: Addison-Wesley, 1994

7 BarBosa V. Strategies for the Prevention of Communication Deadlocks in Distributed Parallel Programs. IEEE Transactions on Software Engineering, November 1990

8 Bic L, Shaw A. The Logical Design of Operating Systems. 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1988

9 Brown R, Denning P, Tichy W. Advanced Operating Systems. Computer, October 1984

10 Carr R. Virtual Memory Management. Ann Arbor,MI: UMI Research Press,1984

11 Casavant T, Singhal M. Distributed Computing Systems. Los Alamos, CA: IEEE Computer Sicuety Press, 1994

12 Davis W. Operating Systems: A Systematic View. 3rd ed. Reading, MA: Addison-Wesley, 1987

13 Grosshans D. File Systems: Design and Implementation. Englewood Cliffs, NJ: Prentice Hall, 1986

14 Horner D. Operating Systems: Concepts and Applications. Glenview,IL: Scott, Foresman, 1989

15 Johnson R. MVS Concepts and Facilities. New York: McGraw-Hill, 1980

16 Krakowiak S, Beeson D. Principles of Operating Systems. Cambridge, MA: MIT Press, 1988

17 Lister A, Eager R. Fundamentals of Operating Systems.4th ed. London: Macmillan Education Ltd, 1996

18 Nutt G. Centralized and Distributed Operating Systems. Englewood Cliffs, NJ: Prentice Hall, 1994

19 Raynal M. Algorithms for Mutual Exclusion. Cambridge, MA: MIT Press, 1986

20 Silberschatz A, Galvin P. Operating System Concepts. Readings, MA: Addison Wesley, 1996

21 Stankovicm J, Ramamritham K, et al. Advances in Real-Time Systems. Los Alamitos, CA: IEEE Computer Society Press, 1993

22 Tanenbaum A, Rebesse R. Distributed Operating Systems. Computing Surveys, December, 1995

23 Turner R. Operating Systems: Design and Implementations. New York: Macmillan, 1996

24 Willam S. Operating Systems.2nd ed . New Jersey: Prentice-Hall International Inc., 1997

25 **何炎祥等. 计算机操作系统原理及其习题解答. 北京: 海洋出版社，** 1993

26 **何炎祥等. 分布式操作系统设计. 北京: 海洋出版社，** 1993

27 **何炎祥等. 并行程序设计方法. 北京: 学苑出版社，** 1995

28 **何炎祥等. 高级操作系统. 北京: 科学出版社，** 1999

29 **孟庆昌. 操作系统教程.西安：西安电子科技大学出版社，** 1993