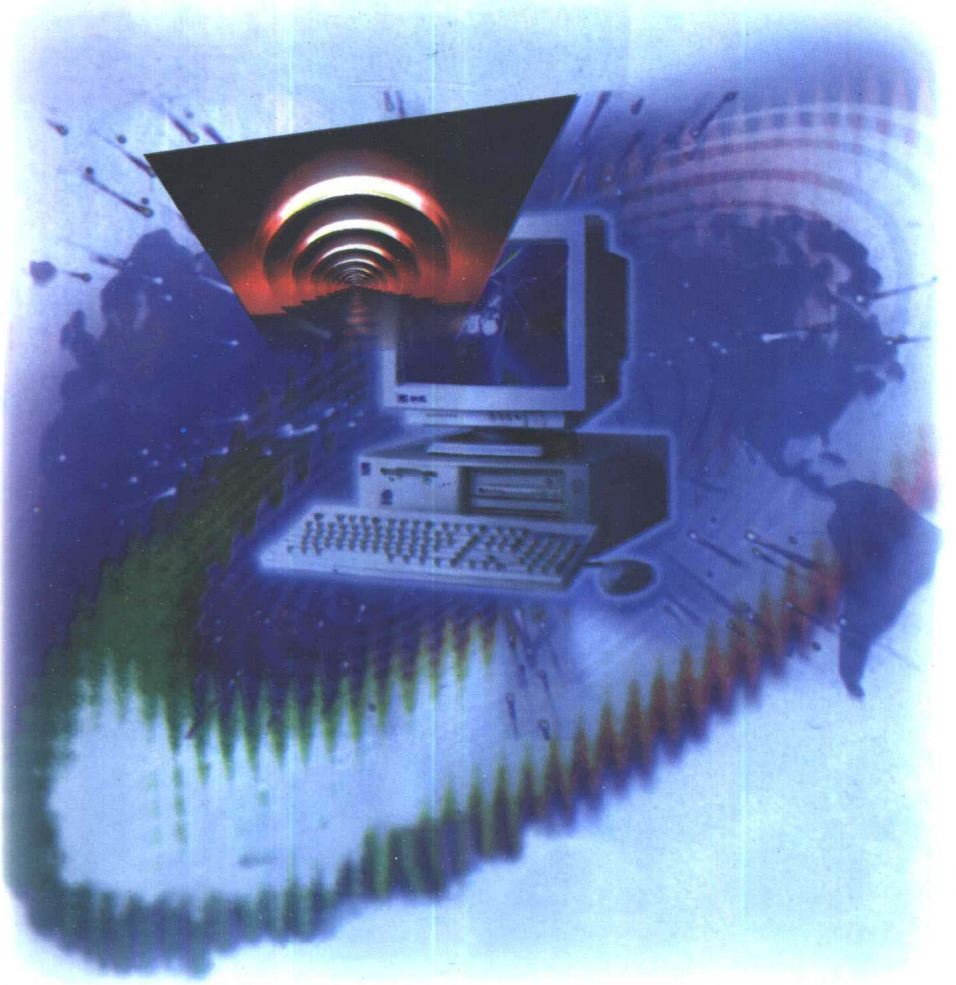
高等学校
电子信息类 规划教材



高级操作系统

徐甲同 编著

西安电子科技大学出版社

·高等学校电子信息类规划教材

高级操作系统

徐甲同　编著

西安电子科技大学出版社
1998

内容简介

本书全面系统地论述了计算机网络操作系统、分布式操作系统和多处理机操作系统的理论及相关技术，反映了当代计算机操作系统发展的新动向和新水平。

全书共九章。第 1 章为操作系统的结构设计；第 2、3 章讲述网络操作系统；第 4、5、6 章讲述分布式操作系统；第 7、8 章讲述多处理机操作系统；第 9 章介绍远程数据库访问技术的原理与实现。

全书层次分明，概念精确，逻辑性强，内容丰富，叙述深入浅出，适合作为高等学校计算机专业研究生和本科高年级学生的教材，也可供相关专业师生及技术人员学习、研究高级操作系统参考。

高等学校电子信息类规划教材

高级操作系统

徐甲同　编著

责任编辑　杨　兵　云立实

西安电子科技大学出版社出版

西安市乾兴印刷厂印刷

陕西省新华书店发行　新华书店经售

开本 787×1092 1/16　印张 15 4/16 字数 356 千字

1998 年 1 月第 1 版　1998 年 1 月第 1 次印刷　印数 1 - 4 000

ISBN 7 - 5606 - 0551 - 6/TP • 0273(课)　定价：13.80 元

出 版 说 明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》，我们组织各有关高等学校、中等专业学校、出版社，各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21世纪教学内容和课程体系改革的要求，编制了《1996—2000年全国电子信息类专业教材编审出版规划》。

本轮规划教材是由个人申报，经各学校、出版社推荐，由各专业教学指导委员会评选，并由我们与各专指委、出版社协商后审核确定的。本轮规划教材的编制，注意了将教学改革力度较大、有创新精神、有特色风格的教材和质量较高、教学适用性较好、需要修订的教材以及教学急需、尚无正式教材的选题优先列于规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科发展具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的学校、教师、学生和其他广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

电子工业部教材办公室

前言

本教材系按电子工业部的《1996—2000年全国电子信息类专业教材编审出版规划》，由计算机专业教学指导委员会编审、推荐出版。本教材由西安电子科技大学徐甲同担任主编，主审康继昌教授，责任编委卢正鼎教授。

本教材的参考学时数50学时，其主要内容包括网络操作系统、分布式操作系统和多处理机操作系统。对此，编著者把这些内容都列入高级操作系统课程的核心内容。无疑它比传统的单机或集中式操作系统更复杂、更先进和更高级。

全书共九章。第一章讲述操作系统的结构设计，介绍了几种现代操作系统的结构设计模式。第二、三章介绍网络操作系统，同时分析了局域网操作系统NetWare。第四章至第六章讲述分布式操作系统，讨论了分布式操作系统中的同步、通信以及死锁等问题。第七章、第八章讲述多处理机操作系统。第九章介绍远程数据库访问(RDA)的原理和实现。

使用本教材时注意以下几点：

① 学生在学习本课程之前，应修完“程序设计”、“数据结构”、“操作系统”等课程，并且最好具有数据库原理和计算机网络方面的知识；

② 由于本书各组成部分相对独立，如果教学时数不够，第一、九章可以不讲。

在本教材编写过程中参考了不少有关资料，这些资料都列入书后的参考文献中。对这些资料的作者、编者深表谢意。

在本书的出版过程中，得到了西北工业大学康继昌教授的热情支持，他在百忙中仔细阅读了全部书稿，提出了许多宝贵意见，在此表示衷心的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编者

1997年10月

目录

第 1 章

操作系统结构设计

结构设计是一个具有普遍性的问题，任何一个大型复杂的工程都应予以认真的考虑，复杂的操作系统的设计，当然也不例外。

一个操作系统的研制过程大体上要经历四个阶段：总体设计、实现设计、调试与维护。其中总体设计包括功能设计和结构设计两个方面。实现设计包括确定数据结构、画流程图和编码三个方面。有关功能设计的问题已在《操作系统教程》[1]中做了详尽的讨论。本章主要讨论结构设计及其实现。

1.1　操作系统结构设计概述

一、结构设计的意义

在操作系统的发展初期，由于系统规模比较小，逻辑关系也比较简单，人们关心的是功能设计和效率，而结构设计问题往往不被重视。随着计算机应用领域的日益扩展，使用要求不断提高，使得操作系统的规模越来越大，结构也越来越复杂。现代操作系统的设计呈现出以下特征：一是复杂程度高。表现在程序庞大、接口复杂、并行程度高。二是研制周期长。从提出要求、明确规范起，经结构设计、编码调试直至系统投入运行，需要几年的时间才能完成。即使开发完成，仍不能投入使用，尚需有充分利用其潜力的应用程序的支持。此外，还必须通过提供文档、培训和实践去学会如何使用。这就意味着，我们现在所拥有并使用的是 10 年或 20 年以前的操作系统技术。三是正确性难以保证。例如，据有关资料介绍，IBM/360 操作系统花了近 5 000 人年的工作量，而每一新的版本都隐藏着约 1 000 个错误；一个拥有 200 万条指令的实时系统，平均每天发现一个错误；一个系统在正常运行了 5 年之后，仍然发现了逻辑上的错误。这些事实都充分说明，操作系统的可靠性是一个十分严重的问题，是一个必须认真考虑和加以解决的问题。

结构设计问题的提出，推动了结构程序设计的形成和发展。什么是结构程序设计？所

谓结构程序设计就是指，为了使程序具有一个合理结构，以保证和便于验证其正确性而规定的一套程序设计方法，或者说是按照一组能增强程序的可阅读性和可维护性的规则而进行程序设计的方法。用这种方法所设计出来的程序，称为结构化程序。若干年来的经验证明，采用这种方法以及与此相关的一些措施，可使程序的错误大大减少，大大地缩短了程序的研制周期。例如，E. W. Dijkstra 曾对他设计的 THE 操作系统断定没有逻辑错误，而且事实上也证实了他的这一大胆的断言。Hansen 所研制的 SOLO 操作系统(包括顺序和并发 PASCAL 编译程序)，如果按传统的手工方式，需要 20～30 人年，由于他采用了结构程序设计方法，结果只花了不到 3 个人年。此外，结构程序设计更为深远的意义还在于：使程序设计从依赖于程序员的技巧变成了一门科学。

二、操作系统结构设计的目标

操作系统结构设计的主要目标如下。

1. 可靠性

操作系统应保护自己不受内部故障和外部侵扰的损害。它应该在任何情况下以预定的轨道运行。应用程序也不应该损害操作系统。

引起操作系统不可靠的因素也是多方面的，但其最重要、最实质性的因素是它的并发性和共享性。由于存在并发性，使得系统执行顺序不唯一。共享性使得进程间产生直接或间接的制约关系，从而造成系统的不稳定性。因此，系统中出现的错误往往不能被重现，于是通过动态方法找出错误原因和确定错误位置是很困难的。有人作过统计，约有三分之二的错误发生在设计阶段。然而一个有良好结构的操作系统，不仅能大大减少错误，而且可以预先证明系统是否存在错误。

2. 简明性

简明性也称清晰性。在进行结构程序设计时，就应该做到：简明、清晰、易读。这样做有利于相互交流和软件维护。

3. 适应性

所谓适应性，有两种含义。其一是可扩充性。操作系统必须易于扩充并随着用户需求的变化而容易修改、增删。其二是可移植性。操作系统应该方便地从一台计算机上移植到另一台计算机上。显然，在结构设计阶段就注意这一问题是十分重要的，也只有在这一阶段就开始考虑这一问题，才有可能达到预期的目的。

4. 高效性

操作系统自身的开销，如占用的主存和辅存空间、占用的处理机时间等，对系统的效率都有直接的影响。减少系统开销、通信开销就能提高系统的效率。目前有一种倾向，就是过分强调可靠性。他们认为，频繁的严重故障所消耗的时间，远远超过提高效率所赢得的时间。过分地强调可靠性和过分强调效率都是片面的。因此，在进行结构设计时，不仅要注意可靠性，而且还要把“时空”效率也作为一个重要目标来对待，这就是说，在保证达到其它目标的前提下，应尽量提高系统的效率。

三、操作系统结构的分类

操作系统的结构有两个含义：一是操作系统的数据结构和控制结构，二是组成操作系

统程序的过程、方法和表示。我们把组成操作系统程序的单位称为操作系统的构件。分析现代操作系统，其构件除内核外，主要还有进程、类程和管程。采用不同的构件和构造方法可以组成不同结构的操作系统。

根据操作系统所使用的构件和通信方式的不同，可以把操作系统的结构分为如下几类。

1. 面向信件的结构

面向信件结构是构造操作系统的一种方法。按照这种方式，整个操作系统的功能由内核和若干进程来完成。进程之间通过统一的通信机构进行联系。在这种结构的系统中，系统程序由内核和一些进程组成。因此，进程不仅仅作为并发执行的单位，而且也作为系统的基本构造单位。在这种结构的系统中，系统内的通信靠通信机构传送信件来完成。所以，面向信件结构的操作系统的主要特征是：系统中有若干进程以及用于这些进程之间通信的显式通信机构。

2. 面向过程的结构

操作系统可以采用面向过程的机构，即操作系统的功能由进程和被它调用的过程来完成。被进程调用的过程往往分别集中于若干个模块中，所以面向过程的结构又称为进程模块结构。面向过程结构的操作系统的主要特征是：系统由若干进程、模块以及一个控制共享数据存取的进程同步机制组成。

3. 面向对象的结构

采用面向对象结构的操作系统的主要特征是：系统由若干对象以及实现对象之间的消息传送机制组成。

1.2　传统的结构设计法

一、模块组合法

模块组合法或称模块接口法，是伴随执行系统的出现而产生的。早期的执行系统是一个交织得比较紧密的复杂系统，对它做轻微的改动都可能影响到全局，使系统变得难以修改或扩充。为了使系统具有较大的适应性，提出了系统模块化的要求，于是引进了模块组合法。这种方法有两个特点：一是程序的模块化，二是模块间的接口。所谓模块化，就是在确定了系统总的功能之后，将其划分为若干具有独立功能的部分，并将每一独立部分编写成称之为“模块”的程序。为使每一模块不至于太复杂，又往往把大模块分成更小的模块，使系统成为一种“积木式”结构。所谓接口，则是在上述模块化的基础上，通过某种接口方式(如转子、调用或藉助固定通讯区、工作单元等)把这些大大小小的模块连接起来，形成一个完整体系。如图 1.1 所示。

这种方法的优点是结构紧凑，接口简单、直接，时空效率高。

从上述定义可以得出，这种方法中各模块之间的一切联系都是靠接口来实现的，因此，设计一个良好的接口是该方法的关键所在。

由于该方法的各模块是一种互相依赖、互相调用的关系，所以在系统运行后，处理机

将在各模块之间交替而无序地转来转去，并形成各式各样的循环。用图论的观点来观察，它是一个相当复杂的有向图。由此可见，这种系统整体性很强，模块间的关系仍然比较复杂，难以对它进行全局性的综合分析，也不容易实现局部修改或扩充，其可靠性和适应性都得不到很好的保证。在操作系统变得日益庞大和复杂的形势下，已不能适应客观发展的需要。

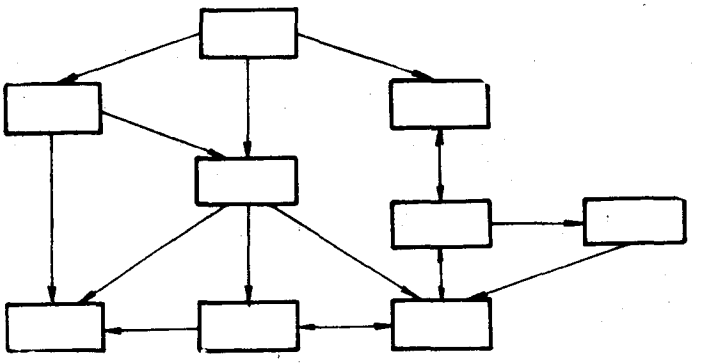


图 1.1 模块组合法

二、层次结构法

上述的模块组合法由于模块之间的调用是无序的，所以也称其为无序模块法。这种方法存在上面所说的一系列缺点，为了克服这些缺点就必须减少模块间的相互依赖关系，消除循环调用现象。层次结构法正是从这一点出发的。这一方法是 Dijkstra 在 ELX8 机器上为实现 THE 操作系统而提出来的，他的报告发表在“ACM. 1968. 5”上。层次结构法又叫有序分层法。

用层次结构法设计操作系统前，必须明确设计目标以及在何种宿主机上运行。层次设计的任务就是在宿主机 A_0 和目标 A_n 之间设计出若干软件层 A_1、A_2、…、A_{n-1}，使得能从 A_0 开始，以各软件层 $A_i(i=1, 2, \cdots, n-1)$为媒介，达到目标 A_n。我们称这样的软件层 A_i $(i=1, 2, \cdots, n)$为一个虚拟机。

设计虚拟机序列 A_1、A_2、…、A_n 的方法有两种：自底向上(Bottom - Up)逐步抽象法和自顶向下(Top - Down)逐步求精法。

自底向上法是以宿主机 A_0 为基础，对 A_0 的各种功能作第一次扩充，得到功能更强的新虚拟机 A_1；然后再以 A_1 为基础，进行第二次扩充，得到新虚拟机 A_2。如此下去，直到获得新虚拟机 A_n，这就是所要达到的目标。用这种方法产生的各个虚拟机 $A_i(i=1, 2, \cdots, n)$具有如下性质：

(1) A_i 所提供的功能和资源是建立 A_{i+1}的全部基础。

(2) 从 A_i 扩充到 A_{i+1}时，并不一定将 A_i 的全部功能同时扩充，那些未经扩充的功能延续到 A_{i+1}中，成为 A_{i+1}的功能。在构造 A_{i+2}时，则以 A_{i+1}的全部功能(即延续来的或新扩充的)为基础。

(3) 在定义 A_i 的新资源时，所用到的 A_{i-1}中的资源在 A_i 中不应再出现。

(4) A_n 的正确性，可通过逐步证明每一虚拟机 A_i 的正确性来得到。

自顶向下的实现过程与自底向上法正好相反，它从目标 A_n 出发逐步过渡到宿主机 A_0。首先设计一个适当的虚拟机 A_{n-1}，使得 A_n 能在该虚拟机上实现全部功能。但虚拟机 A_{n-1} 无法独立运行，还必须再设计一个新的虚拟机 A_{n-2}，而 A_{n-2} 以 A_{n-1} 作为新的目标。重复这一作法，最后得到一系列虚拟机 A_n、A_{n-1}、…、A_1、A_0，其中 A_0 是一个能独立运行的宿主机。这就是逐步求精过程。

自底向上和自顶向下是两种普遍的程序设计法。它们的共同点是均为层次结构，且按分层方法构造操作系统。不同点在于，前者为我们提供了获得树型层次结构的手段，可以从同一宿主系统 A_0 出发，同时得到不同的目标系统 A_n，而后者经过一系列连续扩充，却只得到一个目标系统。要想得到多个目标系统，必须多次使用自顶向下法。

在一个层次结构的操作系统中，如果不仅层次间具有单向依赖关系，而且每一层中各模块之间也是相互独立的，则这种结构称为是全序的，如图 1.2(a)所示。THE 操作系统就是全序的。如果一个层次结构的操作系统仅能做到各层次之间是单向依赖的，不构成循环，而允许在某些层内各模块之间出现循环，则称这种结构是半序的，如图 1.2(b)所示。半序结构比较容易实现。要做到全序，有时是很困难的，尤其是大型操作系统更是如此。

层次结构的最大优点是整体问题局部化。把一个大型复杂问题分解成若干单向依赖的层次，整个系统的正确性可由各层的正确性来保证，对全局的了解建立在对局部了解的基础上，因而系统结构清晰，层次分明，系统的修改和扩充以及调试和维护都比较方便。系统的可靠性也就大大提高。层次结构的缺点是通信开销较大，同步操作过于分散而影响系统安全，不适合于大型操作系统。

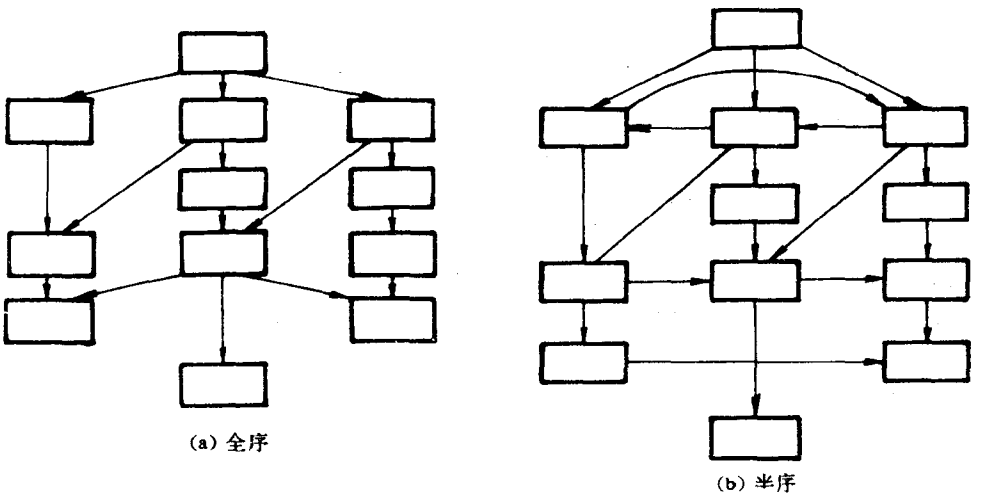


图 1.2　层次结构法

1.3 并发程序设计和管理设计法

一、并发程序设计

并发性是现代操作系统的主要特征之一。在多道程序设计环境中，程序的并发执行代替了程序的顺序执行，从而提高了计算机系统的效率。但另一方面，程序的并发执行也带来了新的问题：进程间的同步和通信。

现在让我们考察一个并发程序设计的例子。这个问题称为流水线系统。如图 1.3 所示。

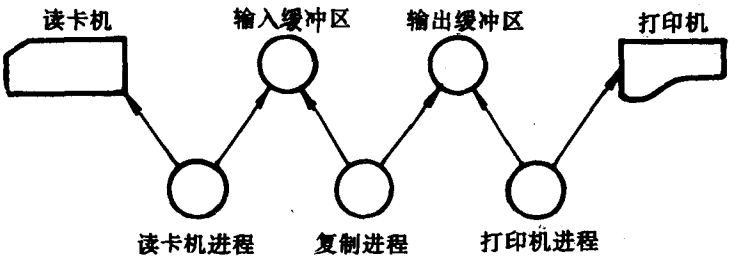


图 1.3 流水线系统

在图 1.3 中，有三个进程：读卡机进程、复制进程和打印机进程，两个缓冲区：输入缓冲区和输出缓冲区。

读卡机进程将卡片机上的卡片(文本信息)读入输入缓冲区；复制进程从输入缓冲区取出文本信息，按格式化要求进行加工，然后放入输出缓冲区；打印进程将输出缓冲区的内容在行式打印机上输出。

文本信息的格式化要求是：每个文件用一个空页开头，用一个空页结尾；每页都用一个空行开头，用一个空行结尾；每页不超过 60 行，每一行两端都留有空格。

一个直观而又简单的解决办法是让三个进程顺序执行，但这样做显然是很低效的。为了提高系统效率，这三个进程必须并发执行。但是我们知道，在这种情况下，如果对缓冲区的存取不加限制，可能会出现丢失信息等错误。因此，必须使这三个进程进行同步。

二、并发程序设计语言

不难看出，上述流水线系统中三个进程是并发执行的，用顺序程序设计语言来描述上述三个进程的并发活动是困难而又复杂的。如能采用并发程序设计语言，则可方便地描述操作系统中诸进程间的并发活动。这是因为在并发程序设计语言中都加进了进程间通信和同步机制。

在并发程序设计语言中，根据它所支持的通信原语的不同，从而形成了很多种类。其中，并发 PASCAL 使用参数传递，CSP 使用消息传送，而 Ada 则使用参数传递与消息传送相结合的方式。

并发 PASCAL 是一种操作系统结构化程序设计语言。它是 Brinch Hansen 于 1975 年

开发出来的，是在 Wirth 1971 年设计的顺序 PASCAL 的基础上改造而成的。其主要特点是支持程序结构的模块化，用它设计出来的系统主要由三种模块(进程、类程和管程)组成。

并发 PASCAL 定义了三种系统类型：进程类型、类程类型和管程类型，分别记为 process、class 和 monitor。系统类型的变量称为系统成分，它们是进程、类程和管程。这样，一个并发程序就由这三种系统成分所组成。进程是主动的系统成分，类程和管程是被动的系统成分。管程是实现进程之间同步的工具。在管程内，标准类型队列 queue 可用来延迟和恢复管程中调用进程的执行。任何时刻都不能有一个以上的进程在队列中等待。一个队列或者为空或者为非空。具有队列类型的变量在管程中只能被说明为一个永久变量。

下列标准函数用于队列：

empty(x)——其结果是一个布尔值，表示队列变量 x 是否为空。

delay(x)——调用进程被延迟在队列 x 中，并使它失去对给定管程变量的存取权，此时该管程可由其它进程调用。

continue(x)——调用进程从执行该操作的管程中返回。如果有另一进程在队列 x 中等待，则该进程立即恢复执行，由它重新控制管程变量的存取。

现在我们回到流水线系统的讨论。该系统应有三个进程：reader、copier 和 printer；两个管程：inbuffer 和 outbuffer；以及三个类程：filemaker、pagemaker 和 linemaker。三个进程被说明为进程类型：cardprocess、copyprocess 和 printprocess；两个管程被说明为管程类型：linebuffer。

为了启动上述三个进程的并发执行，系统应设置一个初启进程 initialprocess，它负责对三个进程和两个管程的说明与初始化。语句 init 用于三个进程的并发执行和定义它们的存取权。流水线系统的层次结构如图 1.4 所示，由 Brinch Hansen 用并发 PASCAL 给出的完整的程序如下。

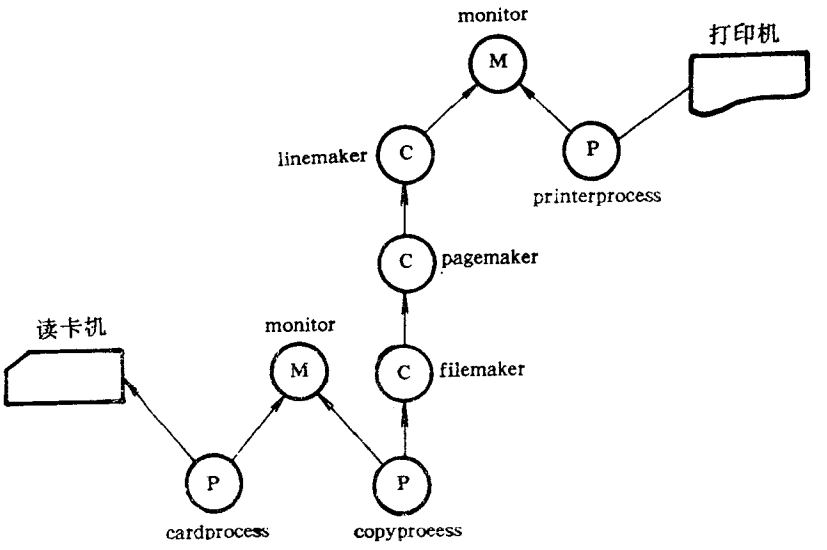


图 1.4　流水线系统的层次结构