海洋出版社

计算机操作系统原理及其习题解答

苏开根
何炎祥 编著

TP316
204

北京希望电脑公司计算机技术丛书

计算机操作系统原理及其习题解答

苏开根　何炎祥　编著

希　望　　审校

海洋出版社

1993年·北京

内 容 简 介

本书以UNIX操作系统为范例，详细介绍了操作系统的基本概念、设计原理以及在构造过程中可能面临的各种问题和解决办法等。全书共分两大部分：操作系统原理(第一至第十五章)和习题解答（第十六至第十八章）。其中第一至第三章讲述了操作系统的基本概念、分类、文件系统及与外部设备的关系等。第四至第八章介绍了操作系统结构、经典的设计算法、CPU调度、存贮器管理、设备管理和死锁处理等。第九至第十五章介绍了较高级的课题，当前发展趋势以及若干已投入使用的操作系统的主要特点等。第十六至第十八章收录了操作系统课程中的基本问题和研究生考试中的疑难问题，在习题类型上进行了分类并给出了详细解答。以便读者抓住重点，学以致用，提高解题和解决实际问题的能力。

本书遵循理论与实践相结合的原则，循序渐进，繁简结合，深入浅出，易学易懂。

本书可作为高等院校计算机科学或工程专业的教材，亦可作为教师、研究生、大学高年级学生及有关科技工作者的参考书。

(京)新登字087号

计算机操作系统原理及其习题解答

苏开根 柯炎祥 编著

希 望 审校

海洋出版社出版（北京市复兴门外大街1号）

海洋出版社发行 北京东升印刷厂印刷

开本：787×1092 1/16 印张：22·625 字数：549 千字

1993年11月第一版 1993年11月第一次印刷

印数：1—5000

ISBN 7-5027-3508-6/TP·177 定价：15.60元

序　言

操作系统是计算机系统中十分重要的系统软件，也是计算机教学中必不可少的教学内容。本书详细地阐述了操作系统的各种基本概念和设计操作系统的基本原理以及构造操作系统过程中可能面临的种种问题及其解决方法。

本书前十五章讲述操作系统原理，其中第一、二章叙述了什么是操作系统，它的分类、演变过程、共性以及它与外部环境的关系。在这几章里，我们尽量避免涉及操作系统的设计细节。因此，这几章比较适合于只打算一般了解操作系统，但不准备深入到操作系统设计细节和较高级的课题中去的读者。第三章讨论了文件系统的内容。第四至第八章介绍了操作系统的结构和各种经典的设计算法，内容包括CPU调度、存贮器管理、设备管理及死锁处理等，并讨论了有关算法的特性和优劣。借助这些知识以及对简单操作系统的理解就不难设计和构造新的较复杂的操作系统。第九章讨论了并发进程方面的若干问题。第十至第十三章介绍了操作系统中较高级的课题和当前的发展趋向，内容包括并行程序设计、保护问题、操作系统的设计原理以及分布式操作系统。其中不少方面还处于研究阶段。我们之所以把这些材料收集进来是因为：第一，尽管有些方面的研究仍在继续进行，且解决其中一些问题的方法仍在探索之中，但我们认为，这些知识对于设计和构造现代操作系统是十分重要的，学生应该得到这方面的教育和训练；第二，操作系统的设计者以及从事有关工作的科技工作者也有必要知道这方面的研究状况和进展。

为使读者建立一个整体概念，我们特辟专章——第十四章来讨论如何将所介绍的概念和原理揉合在一个实际的操作系统中，我们选择了UNIX操作系统来完成这一教学环节，因为它的规模适度，便于理解，而且它也是应用得最为广泛的操作系统之一。在第十五章中我们简述了若干已投入使用的操作系统的主要特点。各章之后附有习题，可供读者温习和进一步研究。

为了配合操作系统原理课程的学习，我们对该课程中一些普遍性的问题及研究生考试中的一些疑难问题进行了分类整理并分别进行了求解，形成了习题解答部分，旨在帮助读者复习、巩固、理解操作系统的一般原理和基本实现技术，便于读者抓住重点，掌握要点，学用结合，以提高解题及解决实践中所面临的有关问题的能力。

本书后三章为习题解答部分（第十六至第十八章）。其中第十六章为分类习题解答。该部分首先给出了分类习题的题目及相应的解答，接着列出了分类复习要点。虽然这些分类习题主要选自前面各章之后的习题，但容易看出，它们与我们所见到的大多数同类教材中的习题几乎没多少差别，因而具有代表性。复习要点则是对操作系统原理中的一些关键概念、重要术语和各种典型算法、策略及其实现思想提供了一种简捷的温习方式，以便读者温故知新、触类旁通。当然，操作系统各部分的内容不是完全独立的，它们之间的联系甚多，有些甚至纵横交叉，考虑到这一点，我们特设第十七章来弥补。第十七章为综合习题解答，在该部分中，我们选择了一批难易程度不一，繁简兼有，典型和普遍相结合，书本知识与实际环境相联系的问题，并逐一给予了解答，希望本章对读者融汇贯通、全面理解操作系统的基本原理有所裨益。最后，在第十八章给出了一些思考题，供读者进一步思考和研究。这些思考

题无论是在深入学习操作系统原理的课程时，还是在剖析、设计一个实际的操作系统时都可能会遇到。

本书在内容和组织形式上有以下特点：首先，我们简明、清晰地讲述了操作系统的基本概念和原理，并努力把一些新观点、新成果及一些概念的演变原因反映在我们的书中。而且是按循序渐进的方式，从简单的基本概念向复杂的、高级概念过渡，因此，有利于读者接受和理解。其次，我们将并发进程的概念和有关内容放在较后（第九章）来介绍。我们认为，在开始讲授操作系统时，学生们并不怎么懂操作系统原理，为了能理解并发进程的概念，他们需要事先学习和理解CPU调度、存贮器管理等是如何在具有虚拟存贮器的独立的虚拟处理机上实现的。经过一段时间的学习之后，他们才能意识到为什么并发进程及相关知识是有用的、必需的。换句话说，只有在学生们具有一定基础知识后，有关并发进程、进程协同、同步和通信等概念才易于被他们所接受。因此，把这些内容安排在稍后介绍比较合适。

我们假定读者已经具有汇编程序设计、高级语言程序设计和计算机组织与结构方面的知识，而且我们不打算讨论I/O设备的任何细节，也不准备去编写设备驱动程序。

本书的第三至第六章、八章、第十一至十二章和十四章由苏开根编写。何炎祥编写了第一、二、七、九、十、十三和第十五章，并对前十五章进行了统编。习题解答部分由何炎祥编写，刘玉珍在开始阶段做了不少工作，在此特致谢意。

在编写过程中，陈良、陈宏宪、邓康、苏涛、黄宏奇、彭堂玉为本书付出了辛勤的劳动；北京希望电脑公司秦人华高级工程师给予了热情支持，在此一并表示感谢。

限于水平，书中错误难免，敬请读者赐教。

何炎祥

于珞珈山

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第一部分　操作系统原理

第一章　引言……（1）
1.1　什么是操作系统……（1）
1.2　早期的系统……（2）
1.3　简单的监控程序……（3）
1.4　性能……（5）
1.5　批处理系统与Spooling技术……（6）
1.6　多道程序系统……（7）
1.7　分时系统……（7）
1.8　实时系统……（8）
1.9　多处理机系统……（9）
1.10　操作系统的外部环境……（9）
1.10.1　操作系统与硬件的关系……（9）
1.10.2　操作系统与其他系统软件的关系……(10)
1.10.3　操作系统与用户的关系……(10)
1.11　练习……(10)
第二章　操作系统的服务……(11)
2.1　服务的种类……(11)
2.2　用户观点……(11)
2.2.1　系统调用……(12)
2.2.2　系统调用的实现……(13)
2.2.3　系统程序……(15)
2.3　操作系统观点……(16)
2.3.1　系统调用……(16)
2.3.2　I/O设备中断……(16)
2.3.3　程序错误……(18)
2.3.4　一般流程……(18)
2.4　小结……(19)
第三章　文件系统……(20)
3.1　文件概念……(20)
3.1.1　文件类型……(20)
3.1.2　基于磁带的系统……(21)
3.1.3　基于磁盘的系统……(21)
3.1.4　分块……(22)
3.1.5　文件操作……(23)
3.2　文件支持……(23)
3.2.1　文件操作……(23)
3.2.2　设备目录……(24)
3.3　存取方式……(25)
3.3.1　顺序存取……(25)
3.3.2　直接存取……(25)
3.3.3　其他存取方式……(26)
3.4　分配方式……(26)
3.4.1　空闲空间管理……(27)
3.4.2　毗连分配……(27)
3.4.3　链接分配……(29)
3.4.4　索引分配……(30)
3.4.5　性能……(31)
3.5　目录系统……(31)
3.5.1　单级目录……(32)
3.5.2　二级目录……(32)
3.5.3　树结构目录……(33)
3.5.4　非循环图目录……(35)
3.5.5　普通图目录……(36)
3.6　文件保护……(37)
3.6.1　命名……(37)
3.6.2　口令……(37)
3.6.3　存取控制……(38)
3.7　实现问题……(38)
3.8　小结……(39)
3.9　练习……(40)
第四章　处理机调度……(42)
4.1　多道程序概念回顾……(42)
4.2　调度概念……(43)
4.2.1　基本构成……(43)

4.2.2 调度队列……(46)
4.2.3 调度程序……(47)
4.3 调度算法……(48)
4.3.1 性能准则……(48)
4.3.2 先来先服务……(49)
4.3.3 最短作业优先……(49)
4.3.4 优先数……(51)
4.3.5 抢占算法……(51)
4.3.6 轮转算法……(52)
4.3.7 多队列调度……(53)
4.3.8 多级反馈队列……(55)
4.4 算法评估……(56)
4.4.1 分析评估法……(57)
4.4.2 模拟方法……(58)
4.4.3 实现法……(59)
4.5 多处理机调度……(59)
4.6 小结……(60)
4.7 练习……(60)
第五章 存贮管理……(63)
5.1 预备知识……(63)
5.2 裸机……(64)
5.3 常驻监控程序……(64)
5.3.1 保护设施……(64)
5.3.2 重定位……(65)
5.4 交换……(66)
5.4.1 后援存贮器……(67)
5.4.2 交换时间……(67)
5.4.3 重叠交换……(67)
5.5 多重分割……(68)
5.5.1 保护设施……(68)
5.5.2 固定区……(69)
5.5.3 可变分割……(72)
5.6 分页管理……(75)
5.6.1 硬件支持……(76)
5.6.2 作业调度……(77)
5.6.3 页表的实现……(78)
5.6.4 共享页面……(79)
5.6.5 保护……(79)
5.6.6 两种存贮观点……(80)
5.7 分段存贮管理……(81)
5.7.1 用户的内存观点……(81)
5.7.2 硬件支持……(82)
5.7.3 段表的实现……(82)
5.7.4 保护和共享……(83)
5.7.5 碎片……(85)
5.8 混合管理方式……(85)
5.8.1 分段分页技术……(85)
5.8.2 分页分段存贮管理……(86)
5.9 小结……(87)
5.10 练习……(87)
第六章 虚拟存贮器管理……(90)
6.1 覆盖……(90)
6.2 请求式页面调度……(92)
6.3 请式调页的性能……(95)
6.4 页面替换……(96)
6.5 虚拟存贮器概念……(98)
6.6 页面替换算法……(99)
6.6.1 FIFO……(99)
6.6.2 最佳替换……(100)
6.6.3 LRU算法……(101)
6.6.4 LRU的近似法……(102)
6.7 分配算法……(104)
6.7.1 块的极小数……(104)
6.7.2 全局分配与局部分配…(105)
6.7.3 分配算法……(106)
6.8 颠簸……(106)
6.8.1 局部化原理……(107)
6.8.2 工作集模型……(108)
6.8.3 缺页中断率……(109)
6.9 其他考虑……(109)
6.9.1 预分页……(109)
6.9.2 I/O 互锁……(110)
6.9.3 页面大小……(110)
6.9.4 程序结构……(111)
6.9.5 存贮层次……(112)
6.10 小结……(113)
6.11 练习……(114)
第七章 磁盘调度……(117)

7.1 物理特性……(117)
7.2 先来先服务调度……(119)
7.3 最短查找时间优先……(119)
7.4 SCAN……(120)
7.5 盘调度算法的选择……(121)
7.6 扇区排队……(121)
7.7 小结……(122)
7.8 练习……(122)
第八章 死锁处理……(123)
8.1 死锁问题……(123)
8.1.1 系统模型……(124)
8.1.2 死锁定义……(124)
8.2 死锁特性……(124)
8.2.1 必要条件……(125)
8.2.2 资源分配图……(125)
8.2.3 解决死锁的方法……(127)
8.3 死锁预防……(127)
8.3.1 互斥……(127)
8.3.2 占用并等待……(127)
8.3.3 非抢占……(128)
8.3.4 循环等待……(128)
8.4 死锁避免……(128)
8.4.1 多个例示的资源类和 Banker 算法……(130)
8.4.2 单一例示资源类……(132)
8.5 死锁检测……(133)
8.5.1 资源类中含若干例示…(133)
8.5.2 每类资源仅有单个例示……(134)
8.5.3 检测算法的应用……(134)
8.6 死锁恢复……(135)
8.6.1 终止进程……(135)
8.6.2 抢占资源……(136)
8.7 处理死锁的混合方法……(136)
8.8 小结……(137)
8.9 练习……(138)
第九章 并发进程……(140)
9.1 优先图……(140)
9.1.1 定义……(140)
9.1.2 并发条件……(140)
9.2 描述并发执行的方式……(142)
9.2.1 Fork 和 Join 结构……(142)
9.2.2 并发语句……(144)
9.2.3 比较……(145)
9.3 进程概念回顾……(146)
9.3.1 进程的状态……(146)
9.3.2 与优先图的关系……(146)
9.4 进程层次……(147)
9.4.1 进程上的操作……(148)
9.4.2 静态和动态进程……(149)
9.5 临界段问题……(149)
9.5.1 临界段问题的定义……(152)
9.5.2 两进程的软件解决方法……(153)
9.5.3 N 进程的软件解决方法……(156)
9.5.4 硬件解决方法……(158)
9.6 信号量及其P,V操作……(160)
9.6.1 用法……(160)
9.6.2 实现……(161)
9.7 经典的进程协同问题……(162)
9.7.1 生产者/消费者问题…(163)
9.7.2 Reader/Writer 问题……(164)
9.7.3 哲学家用餐问题……(165)
9.8 进程间通信……(166)
9.8.1 命名方式……(167)
9.8.2 缓冲……(169)
9.8.3 异常条件的处理……(170)
9.9 小结……(171)
9.10 练习……(172)
第十章 并行程序设计……(175)
10.1 动机……(175)
10.2 模块化……(175)
10.2.1 进程……(175)
10.2.2 过程……(175)
10.2.3 抽象数据类型……(176)
10.3 同步……(177)

10.3.1 临界域……(178)
10.3.2 条件临界域……(180)
10.3.3 管程……(184)
10.4 练习……(190)
第十一章 保护……(191)
11.1 保护的目标……(191)
11.2 机制和策略……(191)
11.3 保护域……(191)
11.4 存取矩阵……(192)
11.5 存取矩阵的实现……(193)
11.5.1 全局表……(193)
11.5.2 存取表……(193)
11.5.3 权能表……(193)
11.5.4 锁/钥机制……(194)
11.5.5 比较……(194)
11.5.6 策略……(195)
11.6 动态保护结构……(195)
11.7 取消……(197)
11.8 几个系统的保护方案……(198)
11.8.1 UNIX ……(198)
11.8.2 Multics ……(199)
11.8.3 Hydra……(200)
11.9 基于语言的保护……(200)
11.10 保护方面的几个问题…… (202)
11.11 安全性…… (203)
11.12 小结…… (204)
11.13 练习…… (204)
第十二章 设计原理……(206)
12.1 目标……(206)
12.1.1 使用者目标……(206)
12.1.2 系统目标……(206)
12.2 机制与策略……(206)
12.3 分层方法……(207)
12.4 虚拟机……(208)
12.5 管程方法……(210)
12.6 多处理机管理……(210)
12.7 实现……(211)
12.8 系统生成……(212)
12.9 小结……(212)
12.10 练习…… (213)
第十三章 分布式操作系统……(214)
13.1 为什么需要分布式系统……(214)
13.1.1 资源共享……(214)
13.1.2 提高计算速度……(214)
13.1.3 可靠性……(214)
13.1.4 通信……(215)
13.2 拓扑结构……(215)
13.2.1 全互连结构……(215)
13.2.2 部分互连结构……(215)
13.2.3 层次结构……(216)
13.2.4 星结构……(216)
13.2.5 环结构……(216)
13.2.6 多存取总线结构……(217)
13.3 通信……(217)
13.3.1 发送策略……(217)
13.3.2 连结策略……(218)
13.3.3 争夺……(218)
13.3.4 保密……(219)
13.3.5 设计观点……(219)
13.4 系统类型……(220)
13.4.1 计算机网络……(220)
13.4.2 局部网络……(221)
13.5 文件系统……(221)
13.5.1 Arpanet FTP ……(221)
13.5.2 集中式途径……(222)
13.5.3 分布式途径……(222)
13.6 资源共享……(222)
13.6.1 数据迁移……(222)
13.6.2 作业迁移……(222)
13.7 事件定序……(223)
13.7.1 事发前关系……(223)
13.7.2 实现……(224)
13.8 同步……(224)
13.8.1 集中式途径……(224)
13.8.2 全分布途径……(225)
13.8.3 令牌传递途径……(226)
13.9 死锁处理……(227)
13.9.1 时间戳定序途径……(227)

13.9.2 死锁检测……(228)
13.10 坚固性……(230)
13.10.1 故障检测……(231)
13.10.2 重构……(231)
13.10.3 故障恢复……(232)
13.11 选择算法……(232)
13.11.1 Bully算法……(232)
13.11.2 环算法……(233)
13.12 小结……(234)
13.13 练习……(234)
第十四章 UNIX操作系统……(235)
14.1 UNIX的历史……(235)
14.2 设计思想……(237)
14.3 程序员界面……(238)
14.3.1 文件操作……(238)
14.3.2 进程控制……(240)
14.3.3 信号……(241)
14.3.4 信息操作……(242)
14.3.5 库子程序……(242)
14.4 用户界面……(242)
14.4.1 shell 和命令……(243)
14.4.2 标准 I/O……(244)
14.4.3 管道、过滤和shell脚本……(244)
14.5 文件系统……(245)
14.5.1 块和段……(245)
14.5.2 索引结……(245)
14.5.3 目录……(246)
14.5.4 文件描述字与索引结之间的映象……(246)
14.5.5 磁盘结构……(247)
14.5.6 实现……(248)
14.5.7 设计和分配策略……(248)
14.6 进程管理……(249)
14.6.1 进程控制块……(249)
14.6.2 CPU调度……(251)
14.7 内存管理……(252)
14.7.1 交换……(252)
14.7.2 分页……(252)
14.8 I/O 系统……(254)
14.8.1 高速块缓存……(255)
14.8.2 原始设备界面……(255)
14.8.3 C 表……(255)
14.9 进程间通信……(256)
14.9.1 socket……(256)
14.9.2 网络支持……(258)
14.10 小结……(259)
第十五章 操作系统历史的简单回顾……(261)
15.1 Atlas……(261)
15.2 XDS-940……(261)
15.3 THE和Venus……(262)
15.4 RC4000和Solo……(262)
15.5 CTSS……(263)
15.6 Multics……(263)
15.7 OS/360……(264)
15.8 其他系统……(265)

第二部分 习题解答

第十六章 分类习题解答……(266)
16.1 基本知识……(266)
16.1.1 习题解答……(266)
16.1.2 复习要点……(268)
16.2 操作系统的服务设施……(274)
16.2.1 复习要点……(274)
16.3 文件系统……(276)
16.3.1 习题解答……(276)
16.3.2 复习要点……(279)
16.4 CPU 调度……(284)
16.4.1 习题解答……(284)
16.4.2 复习要点……(288)
16.5 存贮管理……(291)
16.5.1 习题解答……(291)

16.5.2 复习要点……(296)
16.6 虚拟存贮器管理……(298)
16.6.1 习题解答……(298)
16.6.2 复习要点……(303)
16.7 磁盘调度……(306)
16.7.1 习题解答……(306)
16.7.2 复习要点……(308)
16.8 死锁处理……(309)
16.8.1 习题解答……(309)
16.8.2 复习要点……(314)
16.9 并发进程……(316)
16.9.1 习题解答……(316)
16.9.2 复习要点……(323)
16.10 并行程序设计……(325)
16.10.1 习题解答……(325)
16.11 保护……(329)
16.11.1 习题解答……(329)
16.12 操作系统的设计原理……(331)
16.12.1 习题解答……(331)
16.13 分布式操作系统……(332)
16.13.1 习题解答……(332)
第十七章 综合习题解答……(333)
第十八章 思考题……(344)

此为试读,需要完整PDF请访问: www.ertongbook.com

第一部分　操作系统原理

第一章　引　　言

现代计算机系统是由各种软硬件资源构成的。操作系统是系统软件的基础，它负责组织和管理整个计算机系统的软硬件资源；协调系统各部分之间，系统与使用者之间以及使用者与使用者之间的关系；使整个系统能高效地运转并为系统的使用者提供一个研制和运行程序的良好、方便的环境。因此，操作系统是现代计算机系统极为关键的组成部分。计算机系统越复杂，操作系统就显得越重要。特别是在软硬结合日趋紧密的今天，操作系统在计算机系统中将扮演着极为重要的角色。

为了理解操作系统，有必要说明它们是如何发展起来的。为此，本章将简述操作系统的研制、演变和发展过程，给出研究它们的原因并介绍操作系统干些什么以及如何干的。

1.1　什么是操作系统

操作系统差不多是每一个计算机系统最重要的组成部分之一。一个计算机系统可以粗略地划分为四部分（见图1.1）：

▲硬件（CPU、存贮器、I/O设备）；

▲操作系统；

▲支持程序（编译程序、数据库系统、各种事务处理程序）；

▲用户（人、机器或其他计算机）。

硬件提供基本的计算资源；支持程序提供了用于解决用户计算问题的各种手段。当然，可以有试图解决不同问题的许多用户，因此，可能有许多不同的支持程序。操作系统用来控制和协调供各种用户使用的各种支持程序之间的计算机资源的用法。

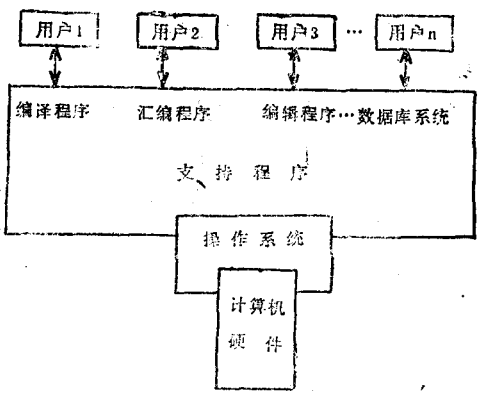

图1.1　计算机系统的组成部分

计算机系统的基本资源是由它的硬件、软件和数据提供的。操作系统提供了在计算机系统操作过程中适当使用资源的方式。可以把操作系统看作是一个**资源分配器**（resource allocator）。计算机系统有许多资源：CPU时间，内存贮区，文件存贮空间，输入／输出设备等等，它们对解决用户的问题是少不了的。操作系统扮演了这些资源管理者的角色，它应用户请求将这些资源分配给特定的程序和用户，并按一种公平的、有效的方式进行分配。

另一个观点是将操作系统看作是一个**控制程序** (control program)。(至少已有三个操作系统将这种观点直接显现在它们的名称中：大多数Burroughs计算机的基本操作系统名为Master Control Program (MCP)；IBM360/67的操作系统CP/67，即Control Program /67；流行的微型机操作系统CP/M的全称就是Control Program for Microcomputers)。该控制程序用来控制用户程序的执行以防止出错或滥用系统的资源。它是与操作或I/O设备的控制特别相关的。

一般说来，我们很难对操作系统给出完全的、准确的定义。计算机系统的基本目标是执行用户的程序，解决用户的问题。朝着这个目标，于是设计并构造出了计算机硬件。由于只有裸机是很难使用的，于是研制出了各种支持程序。各种程序需要某些公共操作，如像控制I/O设备，将这些控制公共功能和分配资源的程序段有机地组装在一起便构成了基本的操作系统。

操作系统的基本目标是方便用户。带有操作系统的计算机比未带有操作系统的计算机要容易使用得多。

操作系统的第二个目标是有效地操作计算机系统。这一点对于大型共享的多用户系统是特别重要的。因为这类系统通常十分昂贵，因此，应尽可能有效地利用它。这两个目标：方便性和有效性有时是矛盾的。过去，常常把有效性看得比方便性更重要。因此，大多数操作系统理论都集中在如何高效使用计算机资源的有关问题上。

为了弄清操作系统是什么以及它干什么，我们考虑一下30年来操作系统的发展过程。通过这种回顾或许能使我们弄清为什么需要操作系统、操作系统的基本组成部分以及为什么操作系统演变成现在的一些模式。

1.2 早期的系统

最初，只存在计算机硬件。早期的计算机是由控制台控制的、很大的（物理）计算机。程序员编好程序后直接在控制台上控制程序的运行。上机过程大致是：首先，程序员用手工方式将其编好的程序从纸带或卡片机上装入内存；然后，通过按适当的键或开关去加载开始地址并启动程序执行；在程序执行时，程序员／操作员通过控制台上的显示氖灯来监测它的执行情况，当发现错误时，就停止程序的运行，查看存贮单元和寄存器的内容并直接在控制台上调试程序，对它进行排错，然后打印输出或凿孔输出在纸带（或卡片）上，供尔后使用。

在这种环境中，程序员也是操作员，而且大多数系统采用签名或预约方式来分配机时。如果你想用机器，就得先去查看上机签名簿，看看是否有机时可用，如果有，就要一段时间并签上你的名字。但这种方式显然会导致机时不足或机时过量的问题。例如，假定你已要了一小时机时，但在运行过程中发现某些错误还难以排除，致使你不可能在一小时内运行完你的程序。而一小时之后的时间已被别人登记了。因此，你不得不下机等到适当的时候再来，另一方面，你可能在35分钟内完成了该程序的运行。由于你事先登记了一小时，计算机就白白闲置了25分钟。

随着计算机科学技术的发展，一些比较实用的软件和硬件相继问世。读卡机、行打机和磁带机已比较普遍。汇编程序、装配程序和连接程序也陆续设计出来了，它们对程序设计和

上机运行提供了方便。一些通用函数（或标准子程序）库已经出现。在程序设计过程中可直接使用这些函数（子程序）而不必重新编写它们。

执行输入和输出功能的子程序是比较繁琐而又十分重要的。每个I/O设备有其自己的特性，必须仔细地编程。专为I/O设备编写的这类子程序称为**设备驱动程序**(device drixer)。每一不同类型的I/O设备都有它自己的设备驱动程序。它们知道与特定设备相关的缓冲区、标识、寄存器、控制位和状态位的特性和用法。哪怕是一个非常简单的I/O操作，例如，从纸带机上读入一字符，都可能涉及到一系列与该设备相关的复杂操作。所以常常把设备驱动程序事先编好存入库中，需要时仅从库中调出使用即可。

编译程序的出现使得程序设计的任务更加容易，但对计算机的操作也更复杂了。例如，为了执行一个FORTRAN程序，程序员首先要将FORTRAN编译程序加载到计算机。这种编译程序通常保存在磁带上，所以又需要找出适当的磁带并安装到磁带机上。待运行的程序需从读卡机上写入另一磁带。然后，FORTRAN编译程序开始加工该程序并产生汇编语言形式的程序作为输出，汇编语言形式的程序还得经由汇编程序加工。于是又得找出保存汇编程序的磁带并将它装到磁带机上。汇编程序的加工结果需要进行连结以支持库子程序。最后得到可以执行的二进制代码形式。它可以装入内存，并由控制台来控制和监测它的运行。显然，这个过程得花相当多的时间。

运行一个程序有时也称运行一个**作业**。每个作业常常由若干独立的**作业步**所组成。例如，上述例子的一些作业步是：安装FORTRAN编译带，加载FORTRAN编译程序，运行FORTRAN编译程序，卸掉FORTRAN编译带，装上汇编程序带，加载汇编程序，运行汇编程序等等。任何作业步出现错误，都可能导致重新开始这整个过程。每个作业步都可能涉及到装带、卸带等操作。

1.3 简单的监控程序

不难看出，在运行一个作业的过程中，程序员在装、卸带上花了不少时间，操作在控制台上，而CPU却空闲着。早期计算机是很少的而且非常昂贵。例如当时价值$2,000,000的一台IBM7094计算机其预期的生命期只有5年。如果每年按365天计（每天24小时），那么其成本价值为每小时$45.66。另外再加上机房、水电、空调、纸张、维修、操作员等等费用就更多了。

因此，机时是非常宝贵的，计算机的拥有者总希望有较高的利用率以便尽早收回成本并取得赢利。

为了减少因组织一个作业所花费的时间，提高计算机的利用率，当时已从两方面着手。第一，雇用专业计算机操作员。这样，程序员不再直接在计算机上操作。只要完成了一个操作，操作员马上开始运行下一个作业，从而避免由于预约机时而造成的计算机空闲。由于操作员在装、卸带方面比程序员有经验，因而减少了运行一个作业所需的时间。程序员或用户只要指明需要用到什么带或哪些卡片以及有关待运行作业的简单说明即可。当然操作员不会调试一个不正确的程序，因为他并不理解该程序。因此，一旦程序出错，操作员便将存贮器和寄存器中的信息转贮到某处，让程序员根据转贮的信息去排错，他又立即去运行下一个作

业。显然采用这种方式，给程序员调试程序带来了不少困难。

第二，把作业按组或批来组织，让计算机一次运行一批作业。例如，假定操作员依次收到一个FORTRAN作业，一个COBOL作业和另一个FORTRAN作业。如果依次运行它们，那么就不得不安装和加载FORTRAN编译程序，接着安装和加载COBOL编译程序，再安装和加载FORTRAN编译程序。如果我们将两个FORTRAN作业作为一批，显然只需要做一次安装和加载FORTRAN编译程序的工作，节省了操作员的时间。这样，程序员把他们的作业交给操作员，操作员根据其要求将这些作业分成批并按批运行这些作业。每个作业的输出将回送给相应的程序员。

上述改进使计算机的使用率得到了提高，但仍然存在问题。例如，当一个作业停止时，操作员要观察控制台，去确定这个程序为什么停止（正常或异常终止）。如果必要，就进行信息转贮，然后从纸带机或读卡机上读入下一个作业并重新启动计算机运行。从一个作业到下一个作业的转换期间，CPU仍然是空闲的。

为了克服这一点，发展了**自动作业定序**，并用这种思想设计出了第一个初级的操作系统。这个操作系统的核心就是一个很小的控制程序，称为**常驻监控程序**（resident monitor），它总是驻留在内存中（见图1.2）。

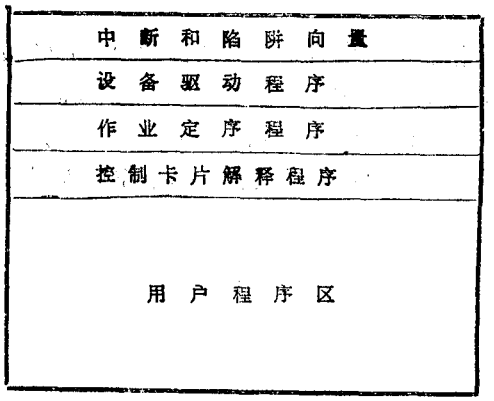

图1.2 常驻监控程序

最初，计算机的控制归属于常驻监控程序，再由常驻监控程序将控制转到某个（用户）程序。当该程序运行终止时，它又将控制返回到常驻监控程序，后者再将控制转到下一程序。这样，由常驻监控程序对作业实行自动定序，将控制从一个作业转到另一个作业。

然而，监控程序如何知道哪个程序要执行？哪个程序先执行？如何为执行的用户程序提供所需的数据？为了把这些信息直接提供给监控程序，引进了**控制卡片**，由这些卡片向监控程序提供必要的信息。例如，$FTN，$ASM，$RUN为三张控制卡片，其含意如下：

$FTN——执行FORTRAN编译程序。

$ASM——执行汇编程序。

$RUN——执行用户程序。

这些卡片告诉常驻监控程序哪个程序要运行。$JOB和$END则用来定义每个作业的界限：

$JOB——作业的第一张卡片。

$END——作业的最后一张卡片。

其它的一些控制卡片定义另外一些功能，如加载（$LOAD）程序等。控制卡片也可以带参数。参数可以用来定义程序名、用户名、上机日期等等。

不少系统中约定：凡第一列是美元符号（$）的卡片即为控制卡片。而IBM的**作业控制语言**（JCL），则以卡片的开始两列是双斜杠（//）表示控制卡片。其它的卡片用来书写不同的代码。图1.3中给出了组织一个简单的批处理系统的样本卡片叠。

监控程序由若干部分组成。一个很主要的部分是控制卡片解释程序。后者有时需要加载程序将系统程序和应用程序加载到存贮器中。因此，加载程序是监控程序的一部分。控制卡片解释程序和加载程序都要执行I/O操作，所以监控程序还包括设备驱动程序。

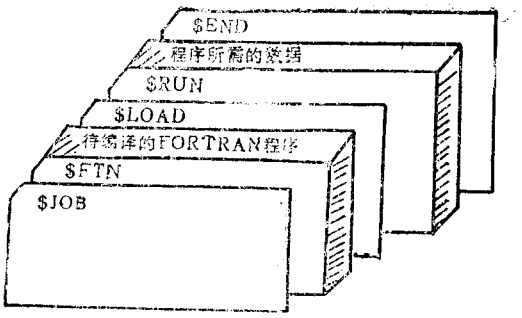

图1.3 一个简单批处理系统的卡片叠

监控程序根据控制卡片提供的信息自动地进行作业定序。当一个控制卡片指明一个程序要运行时，监控程序就将它加载到内存中并将控制转给它。当该程序完成时，它将控制返回到监控程序，后者再读进下一张控制卡片，加载适当的程序等等。重复这个过程，直到有关一作业的所有控制卡片都被解释完毕。然后监控程序自动地处理下一作业。

1.4 性能

虽然利用自动作业定序方法对计算机的性能有所改善，但CPU也是常常空闲的。这是因为机械的I/O设备的速度与电子器件的速度相比毕竟要慢得多。甚至一个慢速的、按微秒级工作的CPU每秒钟也要执行上百万条指令。一个快速读卡机每分钟也只能读1000张卡片。因此，CPU与它的I/O设备之间在速度方面可能相差几个数量级。

I/O设备的（相当）慢速使得CPU常常要等待I/O操作的完成。例如，一个编译程序或汇编程序每秒可处理300张卡片。一个快速读卡机每分钟可能读1200张卡片。那么编译或汇编1200张卡片的程序只需4秒CPU时间，但得花60秒去读它们。这样，CPU就空闲了56秒（约占93.3%的CPU时间）。因此，在这种情况下，CPU的利用率只有6.7%。在执行输出操作时也会出现类似的情况。产生这个问题的原因是：当执行I/O操作时，CPU就空闲以等待I/O操作的完成；而当CPU执行程序时，I/O设备又处于空闲状态。

脱机操作

后来出现了快速I/O设备，但CPU的速度增加得更快。因此，问题并未解决。

一种常用的解决办法是用较快速的磁带设备替代相当慢速的读卡机（输入设备）和行打机（输出设备）。50年代末期和60年代初期的大多数计算机系统是联机方式的批处理系统，即计算机直接从读卡机读入信息，然后将结果（或中间结果）信息直接写到行打机上，见图1.4(a)。稍后，为避免让CPU直接与慢速I/O设备打交道，以提高计算机CPU的利用率，采用了脱机方式，即将待执行的程序及其相关的数据等先从读卡机上复制到磁带上。当需要

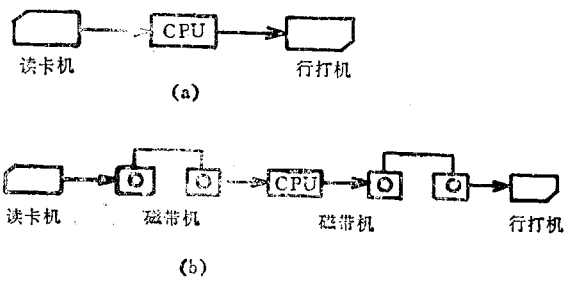

图1.4 (a) I/O设备的联机操作；(b)I/O设备的脱机操作

时，就从磁带机上读入信息。输出时可用类似方法，见图1.4(b)。

有时专门指定一个（或一些）小的计算机（如IBM 1401）来执行向磁带复制信息和从磁带上读出信息的任务。这种小计算机称为主机的卫星机。这种**卫星处理**方式就是多机系统的初级模型。显然，这种方式改善了整个计算机系统的性能。

脱机操作的主要优点是主机不再受到相当慢速的读卡机和行打机的制约，而只依赖于比较快速的磁带设备。因此，从直接I/O操作方式改成脱机I/O操作方式不必改动应用程序。例如运行在计算机系统中的某程序采用的是卡片输入方式。当需要一张卡片时，它就调用常驻监控程序中的读卡机的设备驱动程序。如果将读卡机改成脱机操作，那么只需改变设备驱动程序，即现在要读入一张卡片时，得调用磁带机的设备驱动程序。在这两种情况下，该应用程序都接收到相同的卡片信息。

用不同的I/O设备来伴随运行一个程序的能力称为**设备无关性**。设备无关性使得当某程序请求I/O操作时，操作系统有可能确定它实际上将使用哪台设备。因此，程序中凡用到I/O设备处都可用逻辑I/O设备来代表，物理的I/O设备则由操作系统确定。

1.5 批处理系统与Spooling技术

批处理系统也称作业流处理系统。它通常与Spooling (Simultaneous Peripheral Operation On-line) 技术相关，因此，也称Spooling系统。

Spooling技术的基本原理是利用后援存贮器（外存）作为输入输出缓冲，使作业的输入、计算和输出尽可能并行。另外，作业进入系统也通过外存缓冲，即作业先从输入设备送到外存，再从外存送到内存。用于输入输出缓冲的外存称为“井”。井的作用是用来调节作业的不均匀性，输入输出井的作用是调节计算与传输的不协调性。没有井，一道作业的传输和计算可能互等，造成CPU或外设的闲置；有了井，有可能改善这种状况。就作业缓冲而言，外存缓冲也有利于减少各用户上下机交接时间。一般一个作业结束到下一作业进入系统之间要经过一些操作步骤。以卡片机为例，先要将卡片装入卡片机，然后启动卡片机输入卡片；输入过程中发生故障还得重新开始等等。这段时间内系统资源，尤其是CPU未能充分利用。作业缓冲使作业输入过程与作业运行过程并行起来，即在外存中总是保持一定数量的缓冲作业，当需要调入新作业时，直接从外存缓冲作业中选取，不必现等卡片输入机输入。从而调节了作业到达的不均匀性。此外，作业缓冲还提供了调度的灵活性，即运行作业的次序不一定按其进入系统的先后，而是可从作业缓冲中选取，以提高系统的使用率。

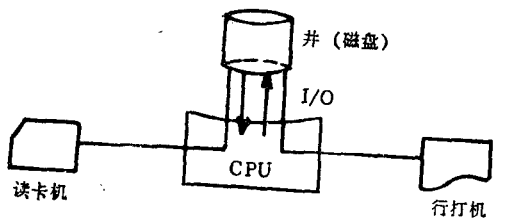

图1.5 Spooling系统

批处理系统的特点是作业从输入设备进入系统后直到结束（正常或异常结束），不允许用户对自己的作业进行控制和干预。这虽然提高了整个系统的性能，但给程序的调试带来了不便。

在Spooling系统中，利用后援存贮器作为输入输出缓冲的方法也叫脱机输入输出，如图1.5所示，一般采用所谓假脱机方式，即输入输出设备与井的联系仍然通过同一台

此为试读，需要完整PDF请访问：www.ertongbook.com