高等学校教材

操作系统原理简明教程

孟　静　编著

高等教育出版社

内容提要

本书深入浅出、简明易懂地介绍了操作系统的原理和使用。全书共七章:第一章为操作系统概论,第二章~第六章依次讲述处理机管理、内存管理、外存管理和文件系统、设备管理和进程通信的原理,第七章介绍分布式、网络、并行和嵌入式操作系统以及操作系统性能评价和结构设计技术。

作者所著的"面向21世纪课程教材"——《操作系统教程——原理和实例分析》出版后,受到全国许多高校老师的认可与好评。为了适应高校教学时数少以及非计算机专业对操作系统课程的教学要求,本书对原书中非重点、非主流实用的内容进行了删减。

本书既可作为高等学校计算机专业本、专科教材,也可作为非计算机专业的操作系统课程教材,同时也适合自学和考试复习使用。

图书在版编目(CIP)数据

操作系统原理简明教程 / 孟静编著. —北京:高等教育出版社,2004.7(2006重印)
ISBN 7-04-014621-5

Ⅰ.操… Ⅱ.孟… Ⅲ.操作系统-高等学校-教材 Ⅳ.TP316

中国版本图书馆CIP数据核字(2004)第061992号

策划编辑 耿 芳 **责任编辑** 耿 芳 **市场策划** 陈 振
封面设计 于文燕 **责任印制** 朱学忠

出版发行	高等教育出版社	**购书热线**	010-58581118
社 址	北京市西城区德外大街4号	**免费咨询**	800-810-0598
邮政编码	100011	**网 址**	http://www.hep.edu.cn
总 机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经 销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印 刷	北京明月印务有限责任公司	**畅想教育**	http://www.widedu.com
开 本	787×1092 1/16	**版 次**	2004年7月第1版
印 张	19.5	**印 次**	2006年6月第3次印刷
字 数	390 000	**定 价**	22.00元

物料号 14621-00

前　言

本书深入浅出、全面系统地介绍了操作系统的原理和使用。全书共七章：第一章为操作系统概论，第二章～第六章依次讲述处理机管理、内存管理、外存管理和文件系统、设备管理和进程通信的原理，第七章简介操作系统性能、结构，分布式和嵌入式操作系统等。

操作系统课程要求先修课为汇编语言、数据结构、计算机组成原理和C语言（其中有些课程可与本课程同学期开设）。

作者多年讲授本科操作系统原理课程，在备课和写作过程中参阅了大量英文教材和实际系统剖析文献，其教材已在众多高校实际教学中使用，受到众多教师同行和广大读者的喜爱和好评。本书是在作者所著的“面向21世纪课程教材”——《操作系统教程——原理和实例分析》基础上改编而成，针对目前高校教学时数少的特点，本书对原书中非重点、非主流实用、非基础的内容做了适度删减。本书有以下特点：

（1）内容既简明适度又系统全面；篇幅适中，难度适宜。

（2）注意理论联系实际。选取最新主流操作系统Linux、Solaris和Windows作为每章末的完整实例介绍，并在讲述基本原理时处处具体及时地联系实际系统。

（3）以操作系统内部工作过程（及相应结构）作为讲述核心和重点，并相应设计了大量图表和例子，从而使学生切实掌握操作系统内部工作原理。

（4）全书内容整体感和逻辑感强，提出了“硬件相关、应用无关”的操作系统本质思路，并围绕该本质思路统一组织各章节和各问题的讲述，从而切实解决了操作系统课程教学的“散杂”问题带来的理解难度。

（5）讲述层次清晰易懂，深入浅出，注意从外至内、从已知到未知。全书内容经过精心组织和多年精益求精，紧紧抓住读者思路，启发式而深入本质的讲述，激励读者思考，适合自学。

（6）书中内容全面反映操作系统最新技术发展，例如内存映像文件、稀疏编址、多级页表以及分布式、网络、并行和嵌入式操作系统技术等。

（7）与本书配套的有《操作系统教程实验指导和题解》，另外，作者的操作系统教学站点http://www.ict.ac.cn/chpc/os上提供PowerPoint课堂演示电子讲稿等辅助教学工具。

本书虽经多年修改和使用，但作者水平有限，疏漏在所难免，敬请读者不吝赐教。

作　者

2004.5.

目　录

第一章　操作系统概论

读者肯定是带着许多问题开始读这本书的，这些问题可以概括为如下 6 个：

1. What—操作系统是什么？做什么？
2. Why—为什么需要操作系统？
3. How—操作系统如何工作？如何使用？
4. What—本书学习哪些内容？
5. Why—学了本书后有何用处？
6. How—如何学？

本书第一章就来解答这 6 个问题。对其中的第三个问题只做大致的解答，其系统、深入的答案构成全书的主要内容。其余 5 个问题的答案也需要读者在全书的学习过程中逐步加深了解和理解。实际上，在学任何一门课程之前，都有类似的这 6 个问题。对任何一门课程的学习也都是对这 6 个问题的解答，课程主要内容也都是第三个问题。

1.1　操作系统是什么与为什么

也许读者说不出来操作系统是什么，但只要用过计算机，就肯定用过操作系统。现在先把“操作系统是什么”这个问题放一放，先来看一看用过的那(几)个操作系统。

1.1.1　引言：你所用过的操作系统

读者或多或少用过或听说过一些程序或软件。那么，下面这些软件中哪些是操作系统？

极品飞车、Windows、Turbo C++、Word、Visual FoxPro、UNIX、用户自己编写的一个 C 语言源程序、Turbo - ASM、VI、Linux。

虽然有些读者不能概括地说出操作系统的定义，但能知道上述软件中只有 Windows、Linux 和 UNIX 是操作系统，其余的软件都是用户程序和其他系统软件(表 1.1)。实际存在的操作系统有许多种(如果包括过去的就有几百种)，目前最常用的是上面这 3 种(UNIX 常见变种有 Solaris、AIX、HP UX 等，详见 1.5 节)。其他常用的还有 Mac OS(苹果计算机上的操作系统)、NetWare(Novell 公司的网络操作系统)、OS/2(IBM 为其高档个人计算机 PS/2 设计的操作系统)和 OS/400(IBM 小型机上的操作系统)等。

根据使用经验(虽然可能是有限的)，你能说一说自己用过的 Windows、Linux 和 UNIX 这些操作系统能做些什么吗？

(1) 用计算机做任何事，都需先运行某个相应的程序。在 Windows 系列操作系统下，通常是通过双击一个程序的图标或程序名来运行该程序，这个图标或程序名通常出现在桌面、桌面的“开始”菜单、资源管理器等处，如果开机后不出现桌面，那就什么程序都不能运行。

桌面是 Windows 操作系统显示的，程序图标的显示、双击鼠标动作的接收和翻译、启动执行相应的程序都属于 Windows 操作系统的功能。此外，Windows、Linux、UNIX 等操作系统下还可以通过命令方式来启动程序，这时也是由操作系统负责显示命令提示符、接受命令行、启动执行相应的程序。

表 1.1 以下软件中哪些是操作系统

软 件 名	说 明
极品飞车	游戏软件，属于应用软件
Windows	操作系统，是 IBM 个人计算机系列上最常用的窗口多任务操作系统
Turbo C++	编译程序，属于系统软件
Word	编辑排版软件，是 Windows 系列操作系统下的编辑软件
FoxPro	数据库管理软件
UNIX	操作系统，是多用户计算机上最常用的操作系统，其具体变种常见的有 SUN 公司的 Solaris、IBM 公司的 AIX、HP 公司的 HP UX 等
自己编写的一个 C 语言源程序	严格地说，这不算程序，只能算数据。这个源程序经编译、连接后产生的才是程序——可执行目标程序
Turbo - ASM	汇编程序，属于系统软件
VI	编辑软件，是 UNIX 下的编辑软件
Linux	操作系统，其界面类似于 UNIX，是目前最常用的操作系统之一

(2) 不管将计算机用于何种应用领域，都经常需要进行文件复制或删除、磁盘内容查看、建立文件夹等工作。在 Windows 操作系统下，可在资源管理器中通过菜单和鼠标操作来完成这些工作。资源管理器是 Windows 操作系统的一个部件，即文件复制、磁盘内容查看等工作都属于 Windows 操作系统的功能。在 Linux 和 UNIX 操作系统下，这些常见工作还可以通过命令方式来完成，例如 ls(查看目录内容)、cp(文件复制)、rm(删除文件)命令等，这些命令都是由操作系统实现的，即实现这些命令的程序代码都是操作系统代码的一部分。

(3) 在 IBM PC 上用汇编语言编写程序时，都需要 INT 语句。使用 INT 语句的目的大都是要做一些 I/O 工作，比如文件读写或打印等。INT 语句实际上是一种特殊的调用语句，它调用的是相应操作系统(例如 Windows，如果用户是在 Windows 下编写汇编程序)的内部功能。通过这种方式，操作系统向用户程序提供帮助(详见 1.2.2 节)。在 Linux 和 UNIX 操作系统中，也有类似的内部功能和相应的对外调用接口。

(4) 在 Windows 中，可以同时运行多个程序(即多任务)，例如，用户在等待一个文件下载(通过运行 IE 程序)的同时，可以编辑另一个文件(通过运行 Word 程序)，这就提高了工作效率和机器利用率。多任务方式是 Windows 提供的。20 世纪 80 年代流行的 DOS 操作系统中就没有类似的功能，因此不能在 DOS 下同时运行两个程序，而 Linux 和 UNIX 还可以让多

个用户同时使用一台计算机，这就更进一步提高了机器利用率。

至此，读者已有了一个印象：操作系统的功能很丰富、很庞杂、很零散，很难概括，好像什么事都做。但操作系统又不可能什么事都做，例如，操作系统不做天气预报，这是由专门的天气预报软件来做的；操作系统不做房屋设计，这是由专门的建筑 CAD 软件来做的；操作系统不是编译程序，用户使用什么语言编写的源程序就用对应该语言的编译程序……

总之，操作系统不直接解决具体的应用问题，也不负责编译源程序。

那么，操作系统到底做什么，不做什么，它准确的功能范围是什么，它到底是一个怎样的软件，有没有统一的内在本质呢？

1.1.2 操作系统是什么与做什么

操作系统(Operating System，OS)是计算机中最重要的系统软件，是这样一组系统程序的集成：这些系统程序在用户对计算机的使用过程中，即在用户程序运行中和用户操作过程中，负责完成所有与硬件因素相关的(硬件相关)和所有用户共需的(应用无关)基本工作，并解决这些基本工作中的效率和安全问题，为使用户(操作和上层程序)能方便、高效、安全地使用计算机系统，而从最底层统一提供通用的帮助和管理(图 1.1)。

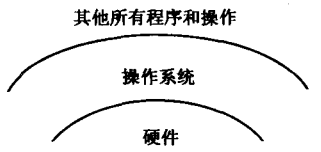

图 1.1 操作系统在计算机系统中的地位

那么，在用户操作和用户程序中，哪些工作内容是硬件相关和应用无关的呢？主要是以下四个方面的工作。操作系统是完成以下四个方面工作的诸系统程序的集成(表 1.2)：

(1) 负责启动执行每个用户程序，并负责完成每个用户程序的结束处理工作，使每个用户程序可以很方便、灵活地启动执行和中止。

(2) 在任何用户程序的运行过程中，负责完成所有硬件相关和应用无关的工作。每当用户程序运行过程中涉及这些工作，就通过一种特殊调用(称为系统调用)或中断方式，调用或进入操作系统来完成，从而为用户程序方便使用计算机系统，提供统一的帮助和管理。

(3) 为用户对计算机进行基本操作提供现成的实用程序和相应的管理，以便这些操作能方便、有效地完成。这里所说的基本操作是指任何应用或开发背景下都通用和普遍需要、经常发生的操作，例如复制文件、删除文件、显示磁盘目录内容或文件内容、格式化磁盘等。

(4) 改善上述三方面工作中的效率和安全问题，使计算机系统的各个部分和整个计算机系统得到高效、安全的使用。

以上四方面工作似乎互无关联，但它们都具有这样的共性：硬件相关和应用无关(见表 1.2)。反过来说，计算机使用过程中的所有硬件相关和应用无关工作就体现这个方面。

何谓硬件相关和应用无关？所谓一个(或一段)程序是硬件相关的，是指该程序代码中包含内外存及设备的物理地址，包含对设备接口寄存器和设备接口缓冲区的读写等操作。

硬件相关的代码必然随硬件的变化而变化。这样的硬件变化包括内外存物理存储空间大小的变化，程序和数据在内外存物理存储空间中存放位置的变化，设备数量和类型的变化，等等(但不包括 CPU 指令集的变化；否则所有的程序就都是硬件相关了)。所谓操作系统承担硬件相关工作，使其上层的用户程序都是硬件无关的，是指当被用户程序使用的硬件发生变化(除 CPU 指令集外)时，用户程序不必改变，人的操作更不必改变，即使是 CPU 指令集发生改变时，所需做的改变工作也是最少的。

表 1.2 操作系统的所有具体工作和它们的共性

		1	2	3	4
操作系统负责的所有工作有四个方面		负责启动每个程序执行并负责完成每个程序的结束处理工作	每当用户程序中使用到 I/O 设备、所存储的信息、内存等时，其中的硬件相关和应用无关工作都通过调用操作系统来完成	为用户的常用基本操作提供现成程序。这里的常用基本操作是指任何应用都需要和通用的操作	解决以上工作中的安全和效率问题
这些工作的具体内容和完成方式举例		例如，通常启动一个程序只需在命令提示符后输入程序名即可，此外操作系统还提供批处理、程序中启动等多种启动方式	例如，在个人计算机上的用户程序需打印字符时，只需以一条 INT 指令调用操作系统即可	例如，DOS 中为文件复制提供的现成程序有 COPY、DISKCOPY 等。用户需复制时只需运行这些现成程序，不用自己编程	例如，操作系统提供多任务方式，可以提高效率
这些工作都与硬件相关	实现该工作的过程代码和硬件因素密切相关(即需要设置与测试，使用物理地址、设备接口寄存器等)	启动和结束过程涉及到对 I/O 设备、内外存的使用与处理，需要物理地址与物理接口寄存器	例如，打印一个字符的过程中需要设置和查看打印机接口寄存器的每一位，需要了解和使用打印机接口的 I/O 物理地址	这些操作大量涉及对外存、I/O 设备的使用	例如，多任务方式的实现过程中需管理内存和I/O设备
	硬件相关必然复杂、繁琐、代码量大	一个程序启动和结束处理工作所需的代码要与外存接口寄存器、外存物理地址、内存物理地址打交道，其代码量可能比程序主体代码量还大	用户程序对 I/O 设备的使用所涉及的代码经常占程序主体代码的大部分，这些代码中多处涉及接口寄存器、物理地址等复杂、繁琐的硬件细节	用户程序对 I/O 设备的使用所涉及的代码经常占程序主体代码的大部分。这些代码中多处涉及接口寄存器、物理地址等复杂、繁琐的硬件细节	例如，提供多用户方式的操作系统比不提供多用户方式的操作系统要大得多，复杂得多
	硬件相关的工作，其实现代码不通用，当硬件变化时，需要重新编写或编译相应代码	例如，如果程序从软盘移到磁带上，其启动代码就需要重新编写	例如，计算机系统换了一台不同型号的打印机，则程序中涉及打印输出的代码都要重新编写		

续表

		1	2	3	4
这些工作都与应用无关	任何应用(使用)都需要该工作	用计算机做任何事都需运行相应程序,而每个程序都需要启动和结束处理工作	任何程序的运行过程中都需要使用内存、I/O设备、外存信息。只要使用I/O设备,就要与这些设备的接口寄存器打交道,只要使用内存,就要与物理地址打交道	不管是财务应用或是人口统计应用,任何应用中都需要进行这些工作	每个用户都希望提高使用效率并希望自己的数据有安全保障,每个机器主人者希望提高机器利用率
	在不同应用中,该工作的过程都是相同的	所有程序的启动过程都是雷同的(就同一种启动方式而言)。所有程序的结束处理工作也都雷同(就同一种结束原因而言)	例如,打印字符A与打印字符E的过程是一样的,只不过打印的数据(A与E)不同	复制一个财务总账文件和复制一个人口清单文件,其复制过程都是相同的,只不过复制的数据不同	
	与具体应用无直接关系(即与用户所关心的应用目标无直接关系)	例如,一个财务软件,其启动和结束处理工作同财务没有直接关系	财务软件中,打印一个数据的过程与财务逻辑上无直接联系	例如,复制的过程与复制的数据无关	
如果没有操作系统来完成这些工作,用户操作和用户程序会怎么样?		所有程序都必须是自启程序,即自包含引导装入代码,所有程序的启动操作都相当于一次关开机操作,麻烦、费时、不灵活	所有程序中都包含硬件相关代码,程序员必须了解相应硬件细节知识,编大量复杂代码。当硬件变化时,必须重编程序	用户要为这些常用操作自行编制程序	用户要自行解决所有安全和效率问题或不能解决
由操作系统完成这些工作后,用户操作和用户程序会怎么样?		用户程序中不必包含启动自己执行的引导装入代码,不必包含与本程序具体功能无关的异常结束处理代码。所有结束出口处只需一条"返回系统"指令。用户不必考虑程序启动与结束,不必了解相关硬件细节知识,不必频繁重复编写相应代码	用户程序中不包含硬件相关及应用无关代码,即不包含与接口寄存器、物理地址打交道的代码。用户不必了解相关硬件细节知识和编写相应的复杂繁琐代码	用户不必为这些常用操作亲自编程	用户不必考虑硬件相关、应用无关的安全和效率问题

一个工作是应用无关的,是指不管用计算机来做什么,不管在计算机上运行什么程序,只要使用相应硬件或相应信息时就要涉及到的工作,它是用户共需的,且工作过程都相同,有共性可循,却又与应用问题没有直接关系。

操作系统为用户的操作和程序完成所有硬件相关和应用无关的工作,目的和益处何在?硬件相关,必然意味着复杂繁琐、代码量很大、代码不通用和变化大,需要用户投入大量的精

力来设计实现和维护修改，并了解相应的大量硬件细节知识，故有必要统一管理，使用户摆脱负担。应用无关，就意味着更有必要统一管理（因为普遍和频繁涉及，与具体应用无直接关系）和能够统一管理（因为工作过程相同）。越是计算机使用中底层的、基本的工作，越具有硬件相关和应用无关的特点，对用户和系统的方便、效率、安全影响越大，越需要、值得并可能由操作系统来完成，解决其中的效率和安全问题，从而使用户程序成为硬件无关的，即独立于硬件，不包括硬件相关的代码，硬件改变时程序不必改变。使用户不必考虑这些底层基本使用过程和了解相关的硬件细节知识，避免频繁重复编写（或编译）与应用问题本身关系不大的大量复杂繁琐的代码，专心于应用本身，更好地达到最终的具体应用要求和目标。总之，操作系统为保证用户的操作和用户程序最终使用的方便、效率、安全，承担了所有硬件相关和应用无关的工作，从最底层提供统一帮助和管理。

操作系统为用户完成所有硬件相关和应用无关的工作，但并不意味着操作系统本身的所有功能都是硬件相关的。操作系统分为实用程序层、命令解释层、核心层，其中只有核心层才是硬件相关的，甚至在核心（层）内部也做了进一步的隔离，只有核心层中最底层的一些模块（例如 Windows 中的硬件抽象层、设备驱动程序等）才是硬件相关的。操作系统有狭义（核心）与广义（至实用程序层，甚至更高层——产品捆绑）之分的说法，也缘于此。

1.1.3　操作系统的规模、数量与重要性

从操作系统的上述四个工作可以看出，操作系统必然是个庞然大物。实际上也正是如此。例如，Windows、Linux、UNIX 都是从光盘安装（或网上下载），需占用几百兆字节的外存。

从 1955 年开始出现操作系统到其后的 40 多年间，在各种机器上实际运行的操作系统有几百个，它们中最小的有上万行代码，大的达几十万行（30 万行 PL/1）甚至几百万行，甚至更多。它们在实际运行时占用了大量的机器时空资源：占用了 20% ~ 45% 的 CPU 时间，操作系统本身占内存空间多达几十兆字节，占用外存空间多达几百兆字节，等等。这些数字都说明了操作系统的规模庞大。另外，不同机器上的操作系统一般来讲是不同的，同一种机器上也可以有不同的操作系统（例如 IBM 个人计算机上可以运行 Windows 和 Linux 等操作系统），同一种操作系统可以在不同机器上有不同版本或变种（例如，UNIX 可以在 IBM、SUN 等很多不同厂商的机器上运行），同一机器上的同一操作系统也有功能升级导致的不同版本（基本上一至五年推出一次新版），这些因素又都导致了操作系统的数量众多。

这些众多庞然大物的产生，不可避免地从用户各自开发，发展到厂家提供，从免费配给，发展到商品化和独立注册，它们的产生（实现）与维护支持使厂家耗资巨大。据 1975 年的估计，每个厂家的每个操作系统花费 200 万 ~ 1 000 万美元，而当时一个厂家同时支持十几个操作系统（多个产品线）、每个操作系统支持 2 个 ~ 3 个版本的情况并不少见。据 1981 年的估计，过去 25 年中所有厂家、用户的操作系统总费用为几百亿美元。2000 年推出的 Windows 2000 耗资几十亿美元。这种庞然大物的使用和维护，也给用户带来了繁重的学习

负担和管理负担，用户机构每年要付出大笔的维护和培训费用(尤其在版本升级时)。

是什么导致人们花费如此多的财力和时间在操作系统上呢？因为操作系统带给用户方便、效率和安全，从而给厂家和用户带来效益，虽然为它付出的代价巨大，但与它带来的方便、效率、安全和效益相比，这种代价是值得的。所以，操作系统虽是令人头痛的庞然大物，但每台计算机都离不了它，凡是与计算机打交道的人(不管是厂家还是用户)也都离不了它。今天，从最小的个人计算机到最大的巨型机，操作系统无处不在，缺之不可。用户很少使用一台没有操作系统的机器，或者说很少不在操作系统的帮助下使用计算机，以至到了对操作系统反而熟视无睹的地步。计算机产生初期没有操作系统时对计算机使用的艰难、低效已经成了遥远而陌生的事情。

1.2 操作系统如何工作

1.2.1 操作系统的第一个工作：负责所有程序的启动和结束

用计算机做任何事，都需先运行某个相应的程序。那么程序如何开始执行呢？从硬件知识可知，任一(可执行目标)程序的启动执行有两个前提条件：一是该程序在内存中(该程序已装入内存)，二是 CPU 中的程序计数器 PC 被置为该程序在内存的起始执行地址(这就意味着 CPU 执行的下一条指令是该程序的指令)。这样，程序很自然就开始运行了。那么，一个程序是由谁来装入和启动行的呢？关于“启动一个程序执行”的申请又是怎样提出的呢？解决这两个问题的答案形成了 6 种启动方式。

1. 程序的第一种启动方式——鼠标单击方式

Windows 下，通常通过双击一个程序的图标或程序名来运行该程序。该图标或程序名通常出现在桌面、桌面上的“开始”菜单、资源管理器等处。程序图标或程序名的显示、双击动作的接收和翻译及找到并启动执行相应程序，都属于 Windows 操作系统的功能，通常由 Windows 中的资源管理器完成，其过程见图 1.2。

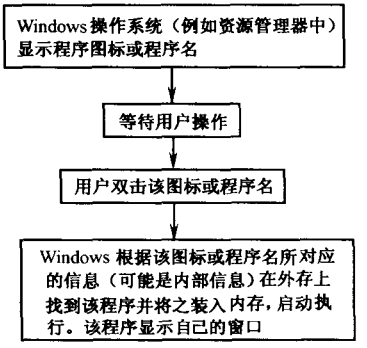

图 1.2 Windows 操作系统中以鼠标方式启动程序的过程

在 Windows 中，一个正在运行的程序通常与一个窗口相联系：启动了一个程序，就打开了一个相应窗口(虽然该窗口有时会被最小化或覆盖)；关闭了这个窗口，就关闭了这个程序。

这种鼠标操作方式和窗口界面形象生动，操作简单而有规律，用户不必记忆命令，是当今所有操作系统中最常用的程序启动方式。Linux 和 UNIX 操作系统中也提供了这种方式

(如 X - Windows 等)。

2. 程序的第二种启动方式——命令方式

所有操作系统下,都提供了命令方式来启动一个程序。

在 Windows 下,可单击"开始"→"运行"→输入程序名(及其所在路径)与参数来启动一个程序(姑且将这种启动方式称为运行方式),也可以单击"开始"→"程序"→"MS DOS 方式"→输入程序名(及其所在路径)与参数来启动一个程序(姑且称之为 DOS 方式,DOS 是 20 世纪 80 年代 IBM PC 上最流行的操作系统)。运行方式和 DOS 方式都是命令方式。

除了 Windows 下的运行方式外,Windows 下的 MS DOS 命令方式、Linux 和 UNIX 下的命令方式中都有命令提示符出现。命令提示符是操作系统在屏幕上向用户提供的一种提示标志,它的出现就表示"你可以在命令提示符后输入程序名来启动下一个程序了"。MS DOS 方式下命令提示符的最常见形式是"C >",Linux 和 UNIX 下命令提示符的最常见形式是"%"或"#"等。通常把在命令提示符后输入的程序名及其参数称为命令行(到回车为止)或命令。任一命令行(即任一条命令)的本质都是申请某一程序执行。在命令方式下的用户工作过程见图 1.3(a)。通常把操作系统中实现命令方式(以及批处理方式)的程序称为命令解释程序或命令解释器,其内部工作过程见图 1.3(b)。

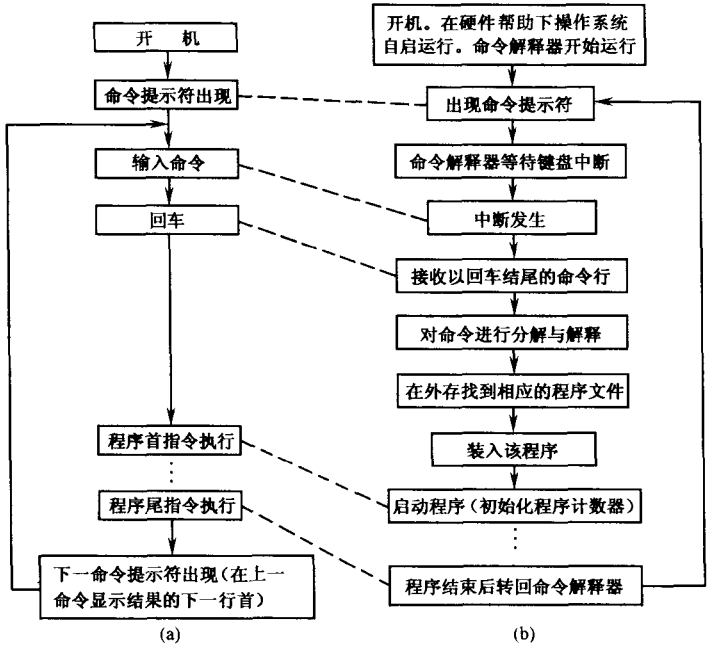

图 1.3　命令方式下的用户使用过程和内部实现过程

Windows 中 DOS 方式的命令解释器是 command.com，可以在 Windows 启动盘根目录下看到这个文件。UNIX 中命令解释器通常称为 Shell。同一操作系统可以有多个命令解释器，例如，UNIX 有 BSH、CSH 和 KSH 等。

命令方式在 20 世纪 80 年代是启动程序运行最主要的方式，但如今它已经让位于鼠标单击方式而成为第二类常用的程序启动方式了。从作业管理角度说，鼠标单击方式与命令启动方式都属于交互式作业。

如果通过命令行启动的程序是操作系统提供的现成程序（即操作系统的第三个工作，参见 1.2.3 节），则该程序、该命令（行）称为操作系统命令。

命令方式下，在命令提示符后输入一个命令而启动执行一个程序后，如何得知该程序是否运行完？是从下一个命令提示符的出现知道的。故命令提示符有两重含义：一是标志着上一个程序的结束（上一条命令的结束）；二是标志着可以启动运行下一个程序（可以输入下一条命令）。

既然在开机通电状态的任一时刻都有程序在运行，或者说有指令在执行，那么在命令方式下，当命令提示符后为空（即光标紧挨在命令提示符后闪烁）而操作者长久未归时，计算机上是在运行程序吗？如果是的话，是什么程序？

从图 1.3 可知，此时是负责命令的接受、解释和启动执行的命令解释器在运行。既然命令解释器负责接受、解释、执行命令行，而用户何时输入命令又是不确定的，因此为了在用户一旦输入命令后尽快地做出响应，命令解释器必须随时随刻准备接受命令，尤其是在命令提示符出现后（因为命令提示符出现的含义就是通知用户可以输入命令了）。所以，当命令提示符后为空时，表明命令解释器正在运行，即正在等待用户输入命令（例如通过一个循环来等待，这个循环不停地检测是否有键盘中断，即是否有用户开始输入命令）。

提问：① 为了以参数 P 执行程序 A，而在命令提示符后输入一条命令“A P 回车”，从开始输入命令到输入了命令尾的回车，直到程序 A 内的第一条指令执行前，计算机在做些什么？② 在命令方式下，从用户程序的最后一条指令执行完到下一命令提示符出现之间，计算机在做些什么？（这两个提问作为本章习题）

3. 程序的第三种启动方式——批方式

除鼠标方式、命令方式外，几乎所有操作系统都有对批文件的处理功能，在批文件中启动的程序称为以批方式启动，是程序启动的第三种方式。批方式是指将若干条命令（请注意命令即启动程序的请求）放在一个文件（即批文件）中，该批文件可以被启动执行（通过命令方式或鼠标方式），其执行过程就是由计算机自动连续执行该文件中的这组命令。批文件是传统的 DOS 中的说法（bat 文件），在 Windows 中，批文件称为脚本文件（script file 或 script），在 Linux 和 UNIX 中，批文件称为 Shell 脚本文件（Shell script）。在 Linux 和 UNIX 中，Shell 脚本文件的用途非常广泛，它实际上是一种“程序”（相应的“程序”设计语言称为 Shell 语言），用于把现有的程序（功能）组合起来实现更复杂的功能。一个典型的 Shell 脚本的内容如下（本 Shell 程序把每个 Linux 命令的名字和功能汇集在一个文件中）：

```
cd /usr/man/man1                          man1 目录下放着 Linux 所有命令文件
ls > /home/dragon/scls                    将 man1 目录下的所有命令文件名放入文件 scls
mjn = 1
while test $ mjn! = 236                   236 是 man1 下的命令文件个数
do
mjflong = 'head - n $ mjn /home/dragon/scls|tail - n 1 '    取得命令名
mjf = 'basename $ mjflong \ .2 '
man - S 2 $ mjf |cat - s |head - n 2|tail - n 1 > >/home/dragon/sclist   将命令解释放入文件
mjn = $ (mjn + 1)
done
```

在传统的 DOS 下,批文件(扩展名为.bat)功能中最典型的两个应用是:① 启动诸如中文环境这样的集成软件或一些参数复杂的软件,简单方便,而且不必要求用户了解很多启动步骤(装字库、启动程序、装汉字输入法等)、参数及相应知识。② 特别的 autoexec.bat 批文件(放在启动盘的根目录下)用来自动执行每次开机时的例行工作。DOS 批文件在 Windows 下仍然可以执行。典型的 autoexec.bat 文件内容如下:

```
path =. " c: \ windows \ command "
prompt = $ p $ q
```

批方式的工作过程见图 1.4。批方式的优点是:① 程序间可以连续执行,不经人工干涉。② 命令可以设计成有顺序控制的形式,称为作业控制语言,使程序可以组合执行。

批文件的名字和位置通常是用户自定的,但扩展名在有些操作系统中是固定的(如在 DOS 下批文件的扩展名必须为.bat)。要启动执行批文件,只需在命令提示符后输入该批文件的名字。另外,有一种批文件的名字和位置是固定的,可以不经用户启动而自动执行,称为自动批文件。在 DOS 下自动批文件名为 autoexec.bat,必须放在启动盘根目录下,在 DOS 启动时自动执行。在 UNIX 下自动批文件名为.profile、.cshrc 或.login 等,只能放在用户主目录下,在注册或 Shell 启动时(第一个命令提示符出现前)自动执行,用于完成每次注册或启动 Shell 时的例行工作。应注意自动批文件启动的程序与自启程序的区别(外存方式的限制有无,通用功能与专用功能,详见本小节标题 6)。

4. 程序的第四种启动方式——在一个程序中启动另一个程序

这种方式也许读者没有注意,但可能用过。至少用过一种编译器吧? 例如 Turbo C++,在编辑好源程序(可以在编译器中编辑,也可以在其他编辑软件中编辑),并将之编译(在编译器中进行)为可执行目标程序(假设命名为 P1)后,需要试运行 P1,和前面的编辑、编译步骤一样,试运行也可以在编译器中进行,而不必退出编译器去试运行后再重新进入编译器。在编译器中试运行,意味着由编译器来启动运行 P1,这就是在一个程序(Turbo C++)中启动执行另一个程序(P1)的典型例子。在其他应用问题中(比如并发程序设计、窗口程序等),还有很多类似的在一个程序中启动运行另一程序的情况和需要,甚至还有不必用户输入程

还有很多类似的在一个程序中启动运行另一程序的情况和需要，甚至还有不必用户输入程序名，便直接根据需要（根据预先存储的程序名或计算出来的程序名）在一个程序中启动运行另一程序的。实际上，前述三种程序启动方式（鼠标、命令、批）最终都是通过这种“程序中启动”方式来实现的，参见图 1.22。

在一个程序中启动另一个程序，这种启动方式便于程序的灵活方便启动和动态自动启动，其外部使用过程和内部实现过程见图 1.5。

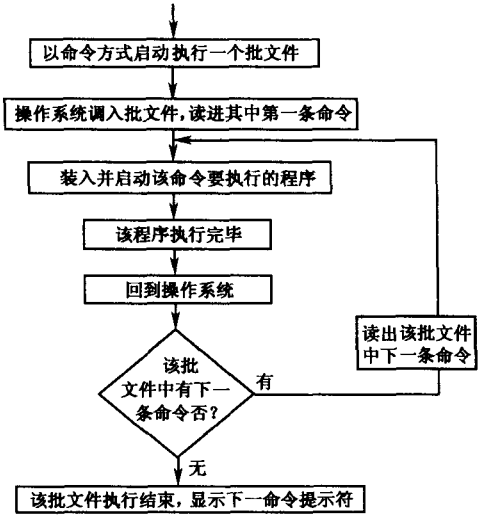

图 1.4 批方式下的工作过程

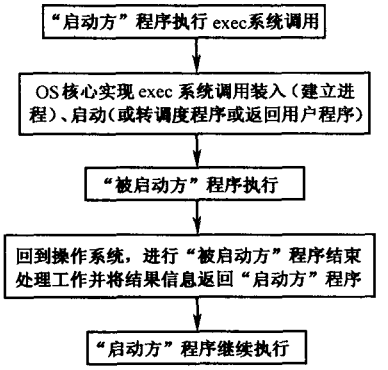

图 1.5 程序中启动方式的工作过程

提问：在一个程序中启动另一个程序的方式与子程序调用方式有何异同？（见本章习题）

5. 程序的第五种启动方式——纯粹由硬件装入并启动程序执行

可以用硬件（即靠纯硬件功能）装入并启动程序执行吗？可以。最早期的计算机就是通过硬件装入并启动程序的（见图 1.6）。用户先把装有可执行目标程序的纸带或卡片安装到纸带输入机或卡片输入机上，然后按一下机器面板上一个特定的“装入程序并启动执行”按钮（注意在此之前内存中什么也没有），硬件就从头开始把纸带上的内容顺序读到内存中，直至纸带上的一个特殊的程序结束标记为止，然后硬件开始从内存零地址处执行程序。这样的顺序连续装入过程是一个很机械的工作，硬件可以承受这样的复杂度，而且只要每个程序都从纸带头开始（即一个纸带一个程序），且都用硬件规定的结束标记，则每个程序的装入过程都是雷同的，用硬件装入也就没有通用性方面的问题。

但后来就很少用硬件装入及启动程序了，这是为什么呢？原因如下：

（1）纯硬件装入的前提是程序必须在一个存储介质上从头开始顺序连续存放，这就限制一个存储介质上只能放一个程序，而且必须从头开始顺序连续存放。在早期只有纸带或