CAOZUOXITONG

YUANLIYUSHIJIAN

操作系统原理与实践

林亚平 编

电子科技大学出版社

UEST PUBLISHING HOUSE

操作系统原理与实践

林亚平　编

电子科技大学出版社

[川] 新登字 016 号

内容提要

本书介绍计算机操作系统的基本原理和实用技术。全书共分六章，介绍了并发进程、存储器、输入输出、文件系统和操作系统实例等基本内容。书中兼顾基本原理和实现技术，每一章在介绍了基本原理之后，都辅之以程序示例或应用示例；书中还集中介绍了 MS/DOS、UNIX 和 Windows 三个流行的操作系统或操作系统环境，并配备有若干实验和习题。

本书在选材上，力求反映操作系统研究领域的新方法、新算法和新技术。本书可作为计算机专业或其他专业的计算机操作系统课程教材，也可供从事计算机应用工作的工程技术人员及有关院校教师和学生参考。

操作系统原理与实践

林亚平 编

*

电子科技大学出版社出版

（成都建设北路二段四号） 邮编 610054

电子科技大学出版社印刷厂印刷

新华书店经销

*

开本 787×1092 1/16 印张 9.5 字数 230 千字

版次 1995 年 11 月第一版 印次 1995 年 11 月第一次印刷

印数 1—5000 册

ISBN 7-81043-336-9/TP・123

定价：9.20 元

前　言

操作系统是计算机系统中最基本的软件，因此，操作系统课程也是计算机学科各专业的主干课程。同时，操作系统又是最为复杂和庞大的软件。它不仅要求程序语言、数据结构等基础软件课程的支持，也涉及到计算机原理等硬件方面的基本知识。从该课程的性质来看，操作系统既包含并发进程等理论性较强的内容，又强调内存、I/O 设备及文件系统管理等在技术上的实现。综上所述，操作系统是一门综合性较强的课程。因此，要理解它的基本原理和实现技术，首先需要一本兼顾这两方面内容的教材。有鉴于此，编者在吸收了多本同类教材的基础上，结合几年来的教学体会，编成了这本教材。其特点是从原理和实践两个方面着眼，力求用深入浅出和精炼的文字进行描述；力求反映新的算法、新的技术。考虑到现实应用，教材中一些举例较多地引用了 UNIX 和 MS-DOS 这两个有代表性的实际系统。与教材配套，本人还编写了实验指导书，并精选了若干习题供学生练习。

教材中难免有误漏及不妥的地方，敬请读者指正，以便修订。

编　者

目　录

第一章 导 论

计算机软件可分为系统程序和应用程序两大种类。系统程序管理计算机本身的操作，并为应用程序提供编程环境，应用程序直接面向用户，为之解决各类问题。最基本的系统程序是操作系统，它控制和管理计算机系统的各类资源，并为其它系统程序和应用程序提供基本的服务。

在硬件结构上；一个现代计算机系统通常由一个或多个处理机、主存、时钟、终端、磁盘、网络接口等I/O设备组成。对于如此复杂的系统，要编制程序正确地管理和使用这些设备，是一件十分困难的事情，更不用说程序的优化设计。如果每个程序员在编制自己的应用程序时，都得考虑如何驱动磁盘工作、读磁盘出错如何处理等等设备工作细节，则许多程序简直就无法编写。为使程序员摆脱这种困难，需要寻找某种方法把硬件的复杂性在程序员面前屏蔽起来。逐步形成的一种方法是在裸机上扩充一层软件，来管理系统中的各类资源，使之呈现在用户面前是一个易于理解和编程的接口或称虚拟机。这层软件便是本书研究的对象即操作系统。

1.1 什么是操作系统

图1-1所示是由硬件、系统程序和应用程序组成的现代计算机系统。

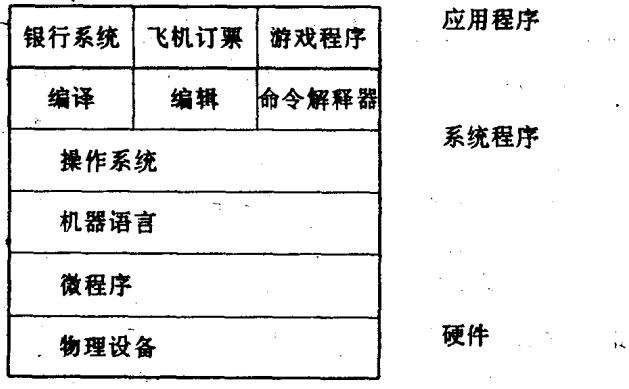

图1-1 现代计算机系统的构成

图中底部是硬件部分，由基本的物理设备和相关的微程序及机器语言构成。物理设备由集成电路芯片、导线、电源等电气元件组成。物理设备之上是一层称之为微程序的原始软件，它通常固化在只读存储器之中。微程序直接控制物理设备，并为其上层提供服务接口。微程序实际上是一个解释器，它负责取机器指令，并按照一定步骤和节拍执行指令。例如，为了执行一条加法指令，微程序必须确定被加数的地址，然后取此数，执行加法操作，再

将结果存储到某一单元。微程序解释的指令集定义了机器语言，尽管它实际上不是机器硬件，但由于它与机器硬件联系密切，习惯上产家和用户都把机器语言看成是实际机器的一部分。

机器语言一般具有 50 到 300 条指令，这些指令通常完成数据传送、算术和逻辑运算、判断和转移等功能。通过机器指令把某些数值装入专用的设备寄存器，I/O 设备便可以得到控制。例如，用指令把磁盘地址、内存地址、字节计数和传送方向等参数装入磁盘专用设备寄存器，便可执行磁盘的读写操作。实际中执行上述的 I/O 操作还需要更多的参数和各种状态信息，而且在编程时还要考虑时序等复杂操作。

操作系统的主要功能之一是要使上述的硬件编程复杂性尽可能对程序员透明，为程序员提供更为方便的编程指令集。例如，上述的磁盘操作可能需要几十条或更多的机器指令，经过复杂的编程才能完成。但操作系统可以预先编好这些程序，而以一条扩充的指令供程序员调用。程序员调用这类指令时，只需简单说明设备地址和所需操作便可实现磁盘的读或写，而完全不必考虑实际的磁盘是如何驱动和工作的。

操作系统之上是其余的系统软件，如命令解释器（也称 shell 即“外壳”）、编译程序、编辑程序等编程工具软件。尽管这些软件通常都由厂家提供，但它们不属于操作系统软件。本章后面将提到一般的计算机系统区分两种工作方式即内核态（也称管态）和用户态方式，操作系统是运行于内核态的那部分软件。它通过硬件保护而不受用户干预，编译、编辑等系统软件则运行于用户态，如果用户不习惯使用这些软件，他完全可以自行编制或修改之。而一般说来，用户是不能任意修改和替换诸如磁盘中断处理程序等隶属于操作系统的软件。硬件保护使用户不能采取上述手段。

综上所述，操作系统的主要目的之一是为了方便用户，从这一观点出发，我们可以认为操作系统是在裸机上扩充了一层软件而形成的虚拟机，这种虚拟机对用户来说，容易理解和使用，一些文献也称虚拟机为扩充机，或把操作系统描述为用户与硬件之间的接口程序，其基本含义是一致的。

操作系统的另一重要目的是有效地管理和使用计算机系统资源，提高资源的利用率。从这一观点出发，也可以把操作系统视为资源管理员。计算机系统中许多资源是多个用户共享的，如 CPU、内存和许多 I/O 设备等。操作系统必须从正确性、公平性和利用效率等方面考虑资源的分配。例如，若有三个运行中的程序同时要求在一台共享的打印机上输出信息，如果不给予适当的管理，就可能导致打印纸上的头几行属于某一程序，另外几行又属于另一程序，三个程序的打印结果交叉在一起而难以区分。为了消除这种混乱，操作系统可以把各程序的打印输出，先存储于磁盘中的缓冲区里，当某一程序完成后再把它输出的结果一次性地从磁盘缓冲区里复制到打印机，进行实际的打印。当打印机正在为某一程序打印结果时，其它程序可以继续输出，只不过是这些输出先在磁盘上缓冲，待打印机空闲时，再进行实际的打印。这样，操作系统即消除了原来的混乱，又保证了多个程序的并行工作，此外，通过在磁盘缓冲区进行排队管理，也照顾到各程序平等地使用打印机。

显然，操作系统的上述两个目的有时是矛盾的。在计算机发展早期，由于其资源十分昂贵，从效率这一方面考虑得更多，因此，操作系统的许多理论也主要是讨论如何优化地使用这些资源，在计算机高度普及的今天，强调方便用户则是软件设计更为重要的目标，以至于“用户友好”（user friendly）已成为计算机术语中的常用词。

1.2 操作系统的形成与发展

为了加深对操作系统的理解和认识，有必要了解操作系统形成和发展的历史。

操作系统和计算机体系结构的发展相互联系，为了方便地使用硬件，形成了操作系统，硬件的改进，又可以大大简化操作系统的设计，因此，在回顾操作系统的历史时，我们也将详细讨论各种硬件技术所起到的重要作用。

1.2.1 早期的系统

早期的计算机只配备有硬件，这些物理上十分庞大的机器通过控制台进行操作，程序员写好程序后，直接在控制台上用手动开关将程序装入内存，并监视程序的执行，一旦发现错误，便通过检查内存单元和寄存器的内容，在控制台上进行调试，其过程与现在某些控制用的单板机操作十分类似。这种环境的一个重要特性是人工操作机器，程序员同时也是操作员。大多数的系统都通过预约登记的方法分配机器的使用时间，但由于程序的实际运行时间难以预测，这种预先分配方法也常常使得机器处于闲置状态，而未得到充分利用。

随着读卡机，行打机和磁带等I/O设备的出现，改变了人工操作的输入/输出方式，汇编、装入和链接程序的引入也使得编程较为容易。此外，公用程序库的建立，大大减轻了程序员编程负担，特别是公共I/O设备处理程序的建立，对后来操作系统中的设备驱动程序设计有着重要的意义。

上述软、硬件的发展，导致了程序员和操作员工作的分离，程序员不再直接操作机器，当一个作业（程序或程序与数据的集合）终止后，操作员可以立即启动下一作业的运行，部分地改善了机器的利用率。

Fortran 和 Cobol 等高级语言的出现，进一步简化了编程工作，但随之也使得计算机的操作更为复杂。为了准备一个作业的运行，通常需要花费大量的作业建立时间。例如，运行一个 Fortran 作业、一般就要完成下列几个相对独立的作业步：装入 Fortran 编译带。运行编译程序对源程序进行编译，装入汇编带、运行汇编程序对编译产生的输出进行汇编，汇编产生的目标代码装入内存，执行目标程序等等。上述作业步引入的装带和卸带工作，导致处理机处于闲置状态，浪费了机器的时间，由于早期的机器十分昂贵，人们自然最为关心如何减少作业建立时间，以提高机器的利用率。

1.2.2 简单批量系统

为了减少作业建立时间，引入了批量处理系统。

批量处理的基本思想是把有类似需求的作业分组，以组为单位统一提交给机器处理。这种批量处理作业的方式，比较单个作业提交处理，所需作业建立时间要少得多。例如，某操作员依次接受了一个 Fortran 作业、一个 Cobel 作业和另一个 Fortran 作业。如果以上述顺序处理作业，则先要为 Fortran 作业建立所需的系统程序，之后要卸下 Fortran 的系统带，为 Cobol 作业建立系统程序，最后又需重新为 Fortran 作业建立系统程序。如果把两个 Fortran 作业一批处理，则只需一次 Fortran 作业的建立准备。

为了消除批量系统中作业在机器上转接的人工干预，需要在机器里驻留一个自动转接

作业的程序。这一思想导致了操作系统的前身监控程序的产生。监控程序实际上是一个常驻内存的小程序，其主要任务是完成作业在机器上的自动转接。机器开工时，控制权由监控程序掌握，之后，由监控程序负责把控制权在作业之间切换。

为了使监控程序识别作业的边界，以及作业中的程序和数据，程序员必须书写作业控制卡。例如，某一作业需执行 Fortran 编译，汇编和用户程序则可以使用下列三张控制卡说明：

$FTN ——执行 Fortran 编译

$ASM ——执行汇编

$RUN ——执行用户程序

其中符号“$”表示为控制卡，以区分一般的源程序和数据卡，也可以利用下列两张控制卡指示作业的开始和结束：

$JOB ——作业第一张卡

$END ——作业最后一张卡

当然，控制卡上还可以说明其它参数，如记帐信息、作业最大运行时间限定等等。书写作业控制卡的语言称为作业控制语言，如 IBM 的 JCL 作业控制语言等等。监控程序通过其包含的控制卡解释程序对控制卡进行识别。这种解释程序逐渐发展成为现代计算机系统中的命令解释程序。

此外，监控程序一般还包括中断和陷井向量、设备驱动程序等，其结构如图 1-2 所示。

以监控程序为基础的简单批量系统对错误的处理是通过硬件陷井实施的，如果用户程序因指令或寻址出错，硬件自动进入陷井，控制返回监控程序，所谓陷井是指系统内部产生的中断，通常它是由程序错误产生的结果。当一个错误引起一个陷井出现时，监控程序自动地输出内存和寄存器的内容，以供程序员排错，然后转去启动下一作业的运行。

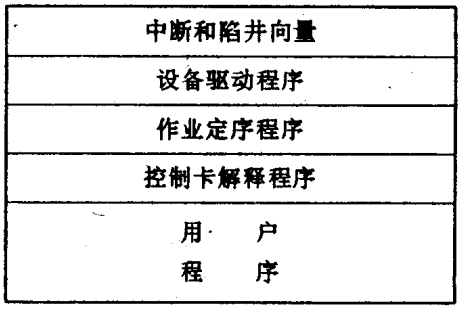

图 1-2 驻留监控的内存映象

但是，某些类型的程序错误会带来新的问题，例如，若某程序在执行读数据卡片的语句时，进入了死循环，如果不采取措施终止此程序，则在读完该程序的所有数据卡后，还会把其它作业的卡片也作为数据卡读入。由于这类错误不是硬件陷井可以识别，必须采用其它方法防止。解决的方法之一是在读卡机设备驱动程序中对卡片进行识别，若程序读的是控制卡，便认为程序出错，而将控制权返回给监控程序，监控程序因此可知发生了错误，进行处理。

问题是如果用户不使用系统提供的标准设备驱动程序，而是因某种考虑采用自己编制的设备驱动程序，此时便很难保证不发生类似前面所述的错误。容易理解，让用户任意修改监控程序和直接使用执行 I/O 操作的某些指令，很可能影响系统的正常工作，甚至导致系统瘫痪。为了提高系统的安全可靠性，一般的系统区分两种工作状态即用户态（目态）和内核态（管态）。在用户态运行的程序不能直接使用某些指令，这些指令通常包含所有的 I/O 指令，改变机器状态的指令，停机指令等等，统称为管态指令或特权指令。监控程序，也就是操作系统，以其特殊身份运行于管态，它可以使用包括管态指令在内的所有指令。

这种双状态的操作方式大大提高了系统的可靠性。回到前面的例子，由于用户程序只能使用系统提供的设备驱动程序完成读卡的I/O操作，就不会发生用户程序将控制卡作为数据读入的错误。

由于用户程序不能直接使用I/O指令，它必须请求监控程序为之完成所要求的I/O操作。监控程序以一种扩充指令作为它和用户程序的服务接口，而完成I/O操作的服务程序代码是监控程序本身的组成部分。现代计算机一般都在汇编语言一级提供这类扩充指令如IBM370中的SVC指令，DEC－10中的UUO指令，IBM－PC中的INT指令，PDP－11中的TRAP指令等等，为了方便，我们统一称之为系统调用。

用户程序要做I/O操作时，通过执行一条相关的系统调用，请求监控程序为之服务，随之控制返回监控程序，监控程序完成I/O服务后，将结果和控制又回送给用户程序。

系统调用更为重要的意义是使得硬件对程序员透明，因而极大地方便了用户编程。系统调用因此成为现代操作系统的重要组成部分。对于用户来说。熟练掌握和使用系统调用，也是理解操作系统的良好途径。

为了防止用户修改监控程序，还必须对存储器进行保护。我们知道，程序出错或程序执行系统调用时，一般是通过中断的方法将控制转交给监控程序，然后在管态下运行相应的服务程序。这些服务程序的地址一般由硬件定义和寻址，形成图1-2所示的中断向量表。如果用户修改向量地址，则控制就可能由用户程序掌握。为了防止监控程序失去控制，必须对监控程序所占用的内存进行保护。

此外，为了防止用户因某种程序错误，进入死循环，在计算机系统增设了定时器，监控程序通过对定时器设置时限，限制用户程序的运行时间，每当定时器值归零，便产生中断，使控制转交给监控程序。

综上所述，以监控程序为代表的简单批量系统，主要是解决了人工操作引入的人机矛盾和提高了系统的安全可靠性。

1.2.3 复杂批量系统

由简单批量系统发展到复杂的批量系统，主要是为了解决低速I/O设备和快速主机的矛盾。例如，一般读卡机的速度是每分钟1000张卡，而主机CPU的速度至少是微秒级，即每秒钟可执行百万次左右条指令，两者速度相差约三个数量级。因此，在进行I/O操作时，CPU闲置以等待缓慢的I/O操作完成，其利用率仍然很低。

解决上述矛盾方法之一是采用缓冲技术，其思想十分简单：当CPU当前正在处理某一作业时，便把下一作业从输入设备读入一个快速的缓冲器中，使得CPU和I/O设备并行工作，当CPU完成当前的作业处理，就可以从缓冲区迅速读入下一作业进行处理。

实际上，CPU和输入设备是不可能完全取得同步的，如果CPU先完成，仍得等待输入设备往缓冲区送信息。但比较无缓冲的情况，CPU等待的时间要少。缓冲技术类似地可应用于输出。

缓冲技术与中断处理直接相关，当I/O设备完成当前的I/O操作，便中断CPU。中断处理程序通过检测缓冲区是否满（对于输入设备），或是否空（对输出设备），决定是否启动该设备处理下一I/O操作。这一事实意味着缓冲技术是操作系统的基本功能。一般在设备驱动程序中包含对系统I/O缓冲区的管理。而用户程序的I/O请求一般只是导致系统缓冲区

中的数据传送。

缓冲技术表现的性能与作业的性质有关。如果作业中的I/O操作与计算操作的时间大致相等，则缓冲技术可以使CPU和I/O设备都保持繁忙，而提高主机利用率。如果作业属于I/O繁忙型（即I/O操作相对计算操作的时间多得多）。则由于CPU速度快于I/O设备，作业的处理速度因I/O速度而受到限制，大量的CPU时间用于等待I/O，此时缓冲技术作用很小。而对于计算量相当大的CPU繁忙型作业，则可能出现输入缓冲区总是满，输出缓冲区总是空的局面，CPU仍然不能与I/O设备保持协调关系。

因此，缓冲技术虽然可以改善CPU和I/O设备的利用率，但其作用并不十分明显。

随着磁带等快速I/O设备的出现，用磁带替代慢速的读卡机和行打机成为解决CPU和I/O速度不匹配的主要方法。早期的批量系统直接从读卡机输入和直接向行打机或卡片穿孔机输出，即这些慢速设备与主机联机操作。用磁带替代慢速的I/O设备，是把读卡机和行打机脱离主机，实际的输入和输出在这些设备与磁带之间进行，主机则通过快速的磁带输入。因此，主机不再受慢速设备的牵制，而只与磁带的速度有关。这一技术称之为脱机操作。实际上，通常利用一台专用的小型机控制慢速设备与磁带之间的操作，此小型机一般称为主机的卫星机，例如，早期的IBM就利用IBM 1401机作为主机IBM 7094的卫星机实际上脱机操作。

脱机操作对系统性能的改进，主要在于其可以使用多套读卡机和磁带输入，或多套磁带和行打机输出，从而使主机保持繁忙。但在另一方面，作业从其递交到在主机上得到处理之间的延时也增大，作业必须先从读卡机输入磁带，并等待磁带装满其它一些作业，然后重绕磁带，并卸下此带装入与主机相联的空闲磁带机上，才可能由主机开始处理。

磁盘的出现，改进了脱机操作。以磁带为基础的系统，由于磁带只能顺序访问，CPU不可能在读卡机向磁带输入的同时，也从磁带上输入，正是这种原因，前述的输入输出必须脱机处理。而磁盘由于其随机访问性质，可以迅速地从输入输出设备转接到CPU处理。在磁盘系统中，读卡机直接输入作业到磁盘，通过操作系统记录作业在磁盘上所处的空间，当作业要执行时，便从磁盘输入内存。类似地，当作业请求打印机输出时，通过系统缓冲区写入磁盘，作业完成后，输出在磁盘和打印机之间进行。由于此时的输入输出设备和磁盘都和主机直接相连，这种技术称为“外部设备同时联机操作”简称Spooling。又因为实际的I/O操作是通过磁盘和I/O设备进行的，也常称此技术为“假脱机操作”。

Spooling技术基本上是以磁盘作为大容量的缓冲区而实现的缓冲技术。所不同的是Spooling可使CPU的计算与若干作业的I/O操作并行处理，因此提高了CPU和I/O设备的利用率。采用Spooling技术后，脱机操作使用的卫星机不再成为必需的设备，也节省了因卸带和装带所花费的大量时间。此外，Spooling提供了一个十分重要的数据结构即作业井。通过Spooling技术，作业输入磁盘某一称之为作业井的分区中，操作系统可根据作业的某些性质选择其运行，以改善系统性能。显然，这种灵活的作业调度在磁带系统中是不可能实现的。这种环境下批量作业处理的含义不再局限于前述的将相类似作业分组，而是根据作业其它性质成批处理。

Spooling技术为以后发展起来的多道程序设计提供了基础，60年代初期，大多数计算机厂家一般都开发了两套相互独立的产品，一套产品是适用于大规模科学计算的大型机，如IBM的7694机；另一套则是面向事务处理的商品用小型机，如IBM的1401机。研制和维

护两套互不兼容的产品，对厂家来说造价昂贵；对用户来说，从小型机过渡到大型机，也十分不便。IBM 为了解决这一问题，引入了 OS/360 操作系统，360 是软件上可兼容的系列机，其包括 1401 到 7094 甚至更高档次的各档机器，这些机器主要区别在于价格和性能，由于所有这些系列机具有相同的体系结构及指令集，一种机器上开发的程序可在较高档次的机器上运行，从而实现软件兼容性。继 360 系列之后，IBM 利用更现代的技术，推出了与 360 兼容的 370、4300、3080 和 3090 系列机。系列兼容机的思想很快也被其它厂家所采纳。现在不少的计算中心仍然在使用这些更新的系列机。

系列产品也存在问题，兼容性要求使得所有软件（包括操作系统）的设计更为庞大和复杂。以 OS/360 操作系统为例，其设计者人数多达几千人，汇编语言程序多达几百万行。自然，如此庞大的软件也包含了许多无法调试和预测的错误。尽管存在这些问题。OS/360 以及类似的操作系统还是较好地满足了用户的要求。技术上，这些系统广泛地使用了多道程序设计。无多道程序设计的系统中，当主机所运行的作业等待 I/O 时，CPU 也闲置直到 I/O 完成。对于 CPU 繁忙的科学计算来说，由于 I/O 较少，CPU 闲置的时间尚不明显，但对于商业数据处理，这些 I/O 繁忙的作业，通常 I/O 活动高达 80～90%，浪费的 CPU 时间便十分可观。采用多道程序设计，可将内存分划成若干个分区，每一分区保存一个作业，当某一作业因等待 I/O 而暂停运行时，另一作业可以立即使用 CPU。如果内存可以保存足够多的作业，则 CPU 可以得到最充分的使用。360 系列机等相类似的机器一般都有专门的硬件保护内存中的作业互不干扰。多道程序设计环境下的操作系统变得十分复杂，由于各类资源被多个作业共享，必须提供适当的 CPU、内存和 I/O 设备等部分的管理。此外，多个作业在 CPU 上并行操作，也要求提供并发控制，这些内容形成了操作系统的基本理论部分。

至此，我们主要从提高系统效率的角度讨论了早期的裸机如何发展过渡到具有 Spooling 操作的多道程序设计批量系统。下面，我们将从用户的角度来回顾这一历史。

1.2.4 分时系统

批量系统中的作业，从其递交到完成并输出结果，所经历的延时一般较长，根据系统的负载以及用户作业的性质，其可能是几分钟、几小时，甚至一天，在这一过程中。用户一般无法知道作业的运行情况，也就是说，在作业执行期间用户和作业不存在交互性质，因此，批量系统比较适宜于大型计算作业和商业数据处理，而一些由若干小的事务性处理构成的作业，由于运行结果的不可预测性，往往要求用户和作业之间进行交互对话，以决定其如何继续执行，通常用户提交一个命令后，希望机器立即给予响应，批量系统显然不能满足这一要求。此外，在批量系统中，因用户不能与作业进行交互对话，用户就必须建立控制卡处理各种可能的情况，对于多作业步的作业来说，下一作业步可能依赖于前面作业步的结果，例如，程序能否运行，就取决于前一步的编译是否成功。因此，用户难以处理各种可能出现的情况。另外，批量系统中，程序员只能通过输出结果静态地调试程序，而不能在作业执行期间动态地研究其性能，这也给程序的调试带来了不便。

要求机器对用户作业响应迅速，导致了多道程序分时系统的出现。分时系统中，每一用户具有一台联机终端，用户通过终端上的键盘输入作业，CPU 迅速地在用户终端之间来回转换服务，其输出显示在终端的 CRT 屏幕上，因每一终端用户提交的命令只占用极短的 CPU 处理时间，一台计算机便可以同时为若干台终端服务，而给用户的印象是其独享该台

机器。第一个实验性的分时系统 CTSS 是 MIT 在 1962 年研制的，其主机为稍加修改的 7094 机，可支持 32 台终端同时工作。之后，在 CTSS 基础上又开发了 Multics 系统，此系统原来的设计思想是要像使用电力一样，提供计算服务，通过电话线把设置在办公室或家庭中的终端与计算中心的主机相连，以取得这种方便，但是，这一设想在实际研制过程中遇到了许多困难，以至该系统最后主要只用于 MIT 的某些环境，尽管如此，Multics 的许多富有创见的设计思想对以后的一些系统带来了巨大影响。

曾参与 Multics 设计的计算机科学家 Ken Thompson，后来在 PDP－7 机器上开发了一个单用户的 Multics 系统，命名为 UNIX。之后，他又与 Denis Ritchie 合作，用 C 语言重写了此系统。不到几年，UNIX 系统得到了相当多的人使用，因此，UNIX 迅速地被移植到许多别的机器上，成为最流行的操作系统，其版本有单用户系统，也有多用户分时系统。历史上，还没有任何其它的系统比 UNIX 具有更多的用户。

现代大型的计算系统一般都具有批量处理和分时处理的能力。通过在前台优先运行交互式作业，在后台运行批量作业的前后台处理方式，较好地结合了两者的特点。

1.2.5 新的发展趋势

70 年代初开始研制，80 年代得到迅速发展的个人计算机，使得计算资源空前丰富，特别是为个人计算机开发的各种软件，极大地方便了用户。即便用户不懂计算机知识，也十分容易使用它。这种用户友好性质的软件，比较 OS/360 等系统，有了根本的改变。

目前，个人计算机中占统治地位的两个操作系统是 Microsoft 公司为 IBM-PC 及其兼容机研制的 MS-DOS，以及前面提及的 UNIX。由于 Microsoft 公司同时也是 UNIX 商品化产品 XENIX 的主要厂家。MS-DOS 新的版本也越来越多地采纳了 UNIX 的许多特点。

另一值得提及的是 80 年代中期开始普及的计算机网络技术，它的出现产生了网络操作系统和分布式操作系统，网络操作系统中，用户意识到多台计算机的存在，本地用户可以录入远程机器，进行文件传送，电子邮件服务等操作，但每一台连网的机器仍然是在本机操作系统管理下工作；分布式操作系统则不同，尽管在实际上系统是由多台机器互连构成，但它在用户面前呈现的仍然好像是单处理机系统，也就是说，用户工作时，不知道自己的程序在哪一台机器上运行，也不知道它的文件存储在什么地方，这一切都是由一个统一的操作系统控制和处理的。网络操作系统通常可以不改变传统操作系统的结构，而是通过增添一些附加软件形成，分布式操作系统则在本质上与传统操作系统存在着区别。

80 年代中期，硬件产业创造了一类称之为精简指令集计算机（RISC）的处理器芯片，比较传统的一些称为复杂指令计算机（CISC）的处理器芯片（包括 80386/80486 等），RISC 技术由于简化了指令集，它可以在提高了时钟速度下运行，且速度很快。在 CISC 和 RISC 技术竞争中，微软公司认识到要充分利用硬件的先进性，就需要研制一种可以从一种硬件平台方便地移植到另一种硬件平台的新的操作系统，着眼于这一目标，微软公司于 1993 年正式推出了号称是 90 年代新技术的操作系统，命名为 Windows NT，NT 意指新技术。

Windows NT 是 32 位的操作系统，它并不是替代目前 DOS 上流行的 Windows，而是为了满足高档的单用户桌面工作站平台、局部网络中的服务器的需要。Windows NT 和 DOS 上的 Windows 都支持图形用户接口，但 NT 具有良好的可移植性、可扩充性，并从本质上克服了 DOS 上 640K 内存的瓶颈，NT 支持对称处理器结构；支持多线程程序，提供了性能优良、

可靠的文件系统。Window NT 目前正在市场上崭露头角。

1.3 操作系统的概念

操作系统提供的系统调用是它与用户的接口。为了真正地理解操作系统，必须熟练地掌握和使用系统调用，在此基础上研究它们的具体实现。仅管系统调用的设计随系统不同而不同，但在功能上有许多相同的地方，附录 1 给出了 UNIX 和 MS/DOS 的系统调用。

由于系统调用涉及操作系统许多内容，我们先介绍一些基本的概念。在理解这些概念的基础上，建议读者结合所学章节的内容上机实践系统调用。

1.3.1 进程

进程（Process）是操作系统中最基本的概念。简单地说，进程是执行中的程序，它由程序、程序使用的数据和堆栈、程序计数器、堆栈指针、其它寄存器和运行程序所需的各种信息组成。

在第二章里，我们将讨论进程的各种特性。在此之前，读者可以联系分时系统产生对进程的感性认识。分时系统中，各进程分时地轮流使用 CPU，由系统决定暂停某一进程的运行和调度另一进程运行。暂停运行的进程下次继续运行时，应从原来断点处开始执行，因此，系统在暂停某一进程时，应将此进程的所有信息保存起来，一般的操作系统都是把除程序外的其它进程信息存储在一个称之为进程控制表的数据结构中。因此，一个进程无论它当前是否在 CPU 上运行，都是由它在内存中的程序，以及进程控制表中对应的信息所组成。

与进程相关的系统调用中，最主要的是建立和取消进程两条调用，以命令解释器（shell）这个典型的进程为例，它负责从终端读入用户提交的命令。如果用户提交的是请求编译一个程序的命令，shell 进程便通过建立进程的系统调用，为运行编译程序建立一个新的进程，当此新进程完成编译后，再执行取消进程的系统调用终止自己。如果一个进程建立了多个新进程（也称子进程），这些新进程依次又建立新的子进程。如此便形成一个树形结构的进程家族，如图 1-3 所示。

此外，进程管理中的系统调用还包括申请和释放内存空间，等待子进程终止，用其它的程序替换子进程的程序等等操作。

某些情况下，用户或系统需要向进程传送一些信息，以指示它做特殊的事情。例如，某进程希望每隔 5 秒钟在终端上输出一次它当前的计算结果，此进程就会请求系统每隔 5 秒钟通知它一次。进程设置这种定时器后，便可做其它的计算工作，当定时区间到达后，系统产生一个定时信号发送给进程，并导致进程暂停当前的计算，保存现场到堆栈，转至事先编制的信号处理程序执行，也就是将当前计算结果在终端输出，信号处理程序执行完后，进程又恢复原来的状态，继续执行，上述发信号的过程，实际上是硬件中断的软件模拟实现。除定时信号外，还可以设置多种原因导致的信号，许多由硬件检测到的陷井，如执行非法指令或使用错误的地址，也都以发信号的形式通知进程进行处理。为此，UNIX 中也提供了设置信号的系统调用。

UNIX 中，每一用户由系统管理员分配一个用户标识符（UID），子进程具有与父进程

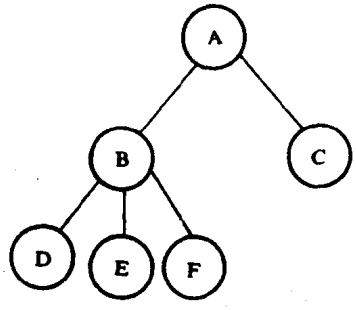

图 1-3 进程树

(建立此进程的进程)相同的 UID。此外,系统存在一个具有特殊权力的超级用户,它可以不受许多规则的约束。在系统安装过程中,仅有系统管理员知道超级用户的口令,因此他具有充当超级用户的特权。

1.3.2 文件

文件是操作系统中的另一基本概念,由于用户接触操作系统通常是从使用文件开始,比较进程,文件的概念就相当具体和直观。事实上,操作系统中与文件相关的系统调用也最多,其中直接涉及文件操作的有建立、删除、读和写文件等系统调用。此外,对文件进行读写操作,一般要求先打开文件,使用完后,应将其关闭,因此也必须提供打开和关闭文件的系统调用。

为了方便,迅速地查找文件,常用目录结构对文件分组管理。典型的目录结构是树形目录,它由一个唯一的根目录和若干层次和数目的子目录构成。每一层次中的目录,其目录项指向的可以是一个文件或是一个子目录。这样,为了查找某一指定文件,一般应给出文件的绝对路径名,即从根目录到达文件所遍历的所有子目录组成的列表信息(包括根目录和文件名本身在内)。绝对路径名以符号/开头,表示从根目录开始,之后各层子目录之间以符号/分隔。为了避免书写长路径名的麻烦,每一进程具有一个可指定的当前工作目录。这样,查找文件时,位于工作目录之上层次的目录名可以在路径名中省略。图 1-4 列举了一个树形目录。其中文件 file 的绝对路径名为/usr/ast/file 若当前工作目录为/usr,则它的路径也可简单地写成 ast/file。

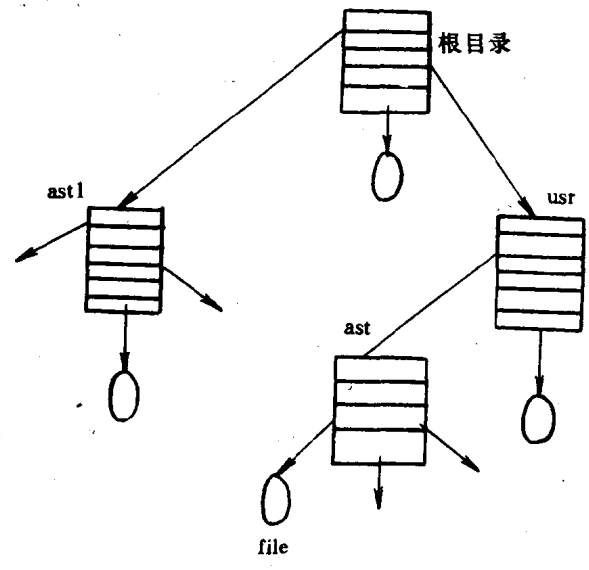

图 1-4 树形目录

为了对文件和目录进行保护,UNIX 采用一个九位的二进制码说明文件或目录的使用权限。此代码分为三组,依次说明文件主、伙伴用户和其它用户这三类用户对文件或目录的访问权限。文件主指的是文件的建立者,伙伴用户一般是指与文件主有某种合作关系的用户,它由系统管理员负责指派,其它用户指的是除上述两类用户以外的用户,每一组代码中的三位二进制位,分别说明文件的读、写和执行三个操作,通常简记为 rwx 位。例如,某文件的保护码若为 rwxr-x--x 则表示文件主可以读、写和执行此文件,其伙伴用户可以读和执行此文件,其它用户则只可以执行此文件。

读写文件之前,必须打开文件,此时系统对用户所要进行的文件操作做权限检查,若

对应的操作在保护码中是允许的，系统便回送一个称之为文件描述字的小整数指示此文件，此后对文件进行的操作。便可使用文件描述字来表示文件，如果系统检查权限时，发现要进行的文件操作在保护码中是禁止的，便会回送一个错误码。对于目录而言，“执行”(x)操作表示是否允许搜索目录。此外，保护码中的“－”号，表示对应的 rwx 操作是禁止的。

UNIX 中有关文件的另一重要概念是文件系统的可装卸性。因为一般的微机都有一个以上的软盘驱动器，为了提供一种简易的方法处理软盘这类可移动的存储介质，UNIX 允许把软盘上的文件系统连接到系统的主文件树上。图 1-5 说明这一安装过程，其中 (a) 图表示内存中原有的主文件树，或称为系统的根文件系统。图 (b) 表示驱动器中软盘上的文件系统。安装之前，由于 UNIX 不允许文件路径名中包含驱动器名，软盘上的文件无法访问。但通过安装文件系统的系统调用，UNIX 可以把软盘上的文件系统装入主文件树上，如图 1-5 (c) 所示，就是把软盘文件系统安装到主文件树上的 b 目录。此后访问软盘上的文件 x，只要给出路径名/b/x 就可进行，值得指出安装软盘的目录一般必须是空目录。

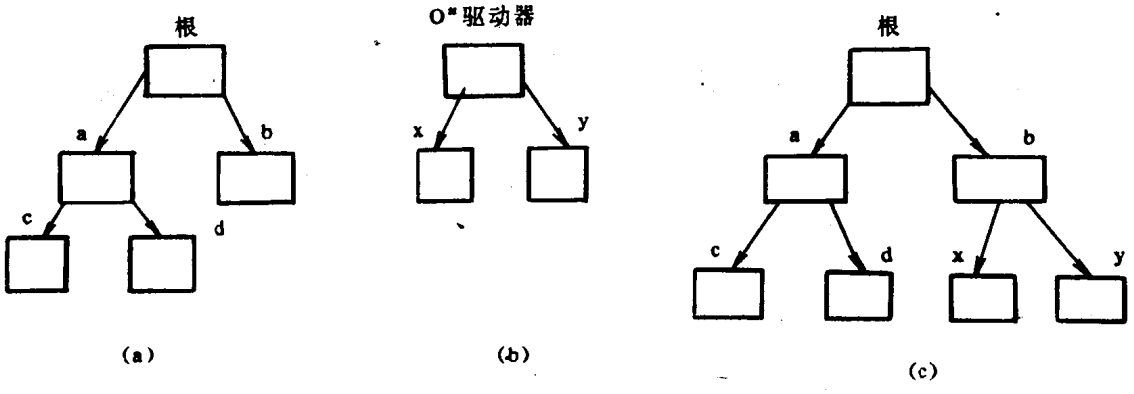

图 1-5　文件系统的安装

和许多操作系统一样，UNIX 把 I/O 设备抽象为一类特殊文件，因此可以把 I/O 设备的读写操作统一用文件系统相关的系统调用处理。特殊文件分为字符型特殊文件和块特殊文件，前者表示按字符流方式处理数据信息的设备，如终端、打印机和网络接口等，后者表示以信息块为单位进行读写的设备，如磁盘等设备。访问特殊文件同样是用 rwx 位的保护方式控制，因此可以把对 I/O 设备直接访问的权力只局限于超级用户。

许多操作系统中都把程序对用户终端的读写方法，设计成对字符特殊文件的读写，当一个进程建立之后，文件描述字 0，也称之为标准输入，专用于从终端读入信息，文件描述字 1，也称为标准输出，专用于向终端写信息，文件描述字 3，也称为标准错误，专用于向终端输出错误信息。

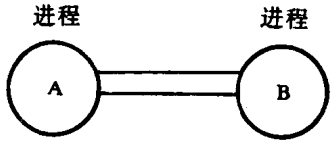

图 1-6　pipe 机制

UNIX 中，所有特殊文件具有一个主设备号和一个从设备号，主设备号说明设备类，如软盘、硬盘、终端；而从设备号用于说明同类设备中的哪一个设备。具有相同主设备号的设备共享操作系统中该类设备的驱动程序，从设备号通常作为一个参数传送给驱动程序，以告知它对哪一设备进行读写，UNIX 中的设备号分配可见/dev 目录。