

UNIX / XENIX

系统程序设计技术

钱培德 编 著

陕西电子编辑部

UNIX / XENIX

系统程序设计技术

钱 培 德 编著

陕西电子编辑部

内 容 简 介

本书详细介绍 UNIX / XENIX 系统程序设计技术, 以及系统内核的程序设计接口——系统调用, 并在此基础上介绍了系统提供的重要库例行程序, 旨在使读者能充分利用系统提供的丰富和强大的功能, 设计出功能齐全和效率高的系统程序的应用程序。书中着重介绍基本技术, 并给出大量有代表性的实例和习题, 有助于读者学习和理解。本书根据程序员的需求, 充分围绕实例展开论述, 向广大 C 语言程序设计者提供了开发和设计 UNIX / XENIX 程序的清晰指南。由于书中的实例尽可能选用系统中的有关部分, 故读者能从中进一步提高对 UNIX / XENIX 系统的认识。

本书可作为大专院校计算机专业的教材或参考书, 也是 UNIX / XENIX 系统研究者、开发者和用户的有用参考书。

作 者

前 言

UNIX 操作系统是一个通用分时系统，著名的美国 Bell 实验室于 1969 年开始研制 UNIX，该系统于 1970 年开始投入运行。起初，UNIX 系统并不引人注目。后来，由于它的先进技术和优越性能，首先引起大学界的重视，并逐渐在大学界推广使用。同时，其结构和能力不断得到优化和完善。以后，它又受到商业界和许多计算机厂商的青睐，UNIX 系统得到了迅速的发展和广泛的应用。

目前，UNIX 已成为计算机操作系统的佼佼者，它广泛应用于教学、科学、工业、商业和国防等领域。它已成为新一代多用户小型和微型计算机的事实上的标准操作系统。但是，其运行范围已由小型机和微型机渗透到中型机和大型机，甚至于巨型机。据统计，现在全世界大约有二百多万台计算机运行 UNIX 系统，这是一个多么惊人的数目。所以，UNIX 成为工业标准这是不容怀疑的。

随着 UNIX 系统的发展，UNIX 的变种也大量涌现出来。其中最著名的是 XENIX 系统，它是由美国 Microsoft 公司开发的。XENIX 系统在 UNIX 的基础上，进行了一系列的增强和优化后形成的，它广泛运行于 16 位和 32 位微型机上。在我国，它已成为拥有最多用户的微机多用户操作系统。

当前，UNIX / XENIX 系统已引起国内计算机界的普遍关注和重视，越来越多的计算机科研人员、计算机专业学生和计算机用户需要学习 UNIX / XENIX。同时我们还注意到，我国的 UNIX / XENIX 用户中仅有一小部分具有开发 UNIX / XENIX 软件的经验，这影响了用户的系统应用水平。我们认为，这与目前可获得的 UNIX / XENIX 资料有关。现在可以获得的这方面资料大多是介绍怎样使用 UNIX / XENIX 系统及其实用程序的，极少涉及到系统环境下的程序设计方法和开发技术。

本书的内容与目前的 UNIX / XENIX 资料有本质的不同，我们在书中重点介绍 UNIX / XENIX 系统环境下的程序设计技术，以及 UNIX / XENIX 系统和程序设计接口——系统调用和库例行程序。系统调用是系统向用户提供的最基本操作，所有的系统程序和应用程序均建立在此基础上。库例行程序能具有与系统调用类似的功能，并能在某些方面强于系统调用，向用户提供更友好的界面，它们运行在比系统调用更高的层次上。我们希望这本书能向读者提供开发 UNIX / XENIX 软件的经验，使读者掌握用 C 语言设计 UNIX / XENIX 系统软件和应用软件的技能，并对系统的内部工作有更进一步的了解。

现在，国内许多高校的计算机专业开设了“UNIX 与 C 语言”这门课，但是尚未有正式的教材。我们希望这本书能成为这门课的教材或教学参考书，为此我们在书中收入了大量典型的实例，并为每章编写了习题，以适应教学的需要。

全书共分九章，另外还有三个附录。第一章介绍 UNIX 的概况、发展历史、系统特点和标准化问题，并提出了系统中的一些基本概念；第二章介绍 C 语言的基本知识，为后面章节的讨论提供基础；第三章和第四章介绍文件系统的系统调用及其程序设计，包括文件基本操作、目录、文件卷和特殊文件等方面；第五章和第六章介绍进程控制的系统调用及其程序设计，包括进程控制操作、进程属性和进程通信机构等方面；第七章介绍终端驱动、终端控制和大型实例等；第八章介绍标准 I/O 库例行程序及其程序设计，包括流

操作、标准输入输出、格式输入输出和程序的执行等；第九章介绍屏幕处理库例行程序及其程序设计，包括 curses 库例行程序、terminfo 库例行程序、动态存贮管理和时间处理等例行程序；附录 A 给出 errno 出错代码与信息；附录 B 列出 UNIX 系统的主要标题文件；附录 C 为全书各章的习题。

本书由钱培德主持编写和组织全书内容，书中第二章、附录 A 和附录 B 由张元道执笔，其它各章均由钱培德执笔，并由钱培德对全部内容作最后修改和定稿。

本书的写作得到张福炎和陈世福老师的支持，本书的正式出版得到了张忠智先生的大力帮助，我们谨在此向他们深表谢意。多位专家和教授担任本书的审稿工作，我们由衷地感谢他们对书稿提出的意见和建议，使书稿更加完美。我们在编写过程中参考了国外和国内的许多资料，在此，我们向这些参考资料的作者表示敬意。

据我们所知，本书是国内第一本全面系统地介绍 UNIX / XENIX 系统程序设计的书籍，所以书中难免有不当和谬误之处。我们恳切希望广大读者不吝指正和批评，以在本书再版时纠正和充实。

钱培德

一九九〇年于苏州大学计算机工程系

目 录

第一章 绪论

第一节 UNIX概述	(1)
一、UNIX的历史与发展	(1)
二、UNIX的其它版本	(2)
三、UNIX的特点	(2)
四、UNIX的结构	(3)
第二节 UNIX的标准化	(5)
一、UNIX标准化的必要性	(5)
二、SVID 标准	(5)
三、POSIX 规范	(6)
第三节 基本概念	(6)
一、文件系统	(6)
二、进程	(8)
三、系统调用和库例行程序	(8)

第二章 C 程序设计语言

第一节 C 语言概述	(10)
一、C 语言的发展	(10)
二、C 语言程序实例	(11)
三、基本数据类型	(11)
四、运算符	(12)
五、数据输入与输出	(14)
第二节 基本语句	(17)
一、条件语句	(17)
二、循环语句	(17)
三、开关语句	(18)
四、转跳和继续语句	(19)
第三节 预处理程序	(20)
一、符号常数	(20)
二、宏代换	(20)
三、文件蕴含	(20)
四、条件编译	(21)
第四节 函数	(21)
一、函数的一般形式	(21)
二、函数的参数	(32)
三、函数的返回值	(24)
四、变量存贮类型	(25)

五、递归·····	(26)
第五节 指针与结构·····	(26)
一、指针·····	(26)
二、指针和数组·····	(27)
三、二维数组和指针·····	(28)
四、结构·····	(29)
五、指向结构的指针·····	(29)
第三章 文件操作	
第一节 文件的基本操作·····	(31)
一、概述·····	(31)
二、文件的打开·····	(32)
三、文件的建立与关闭·····	(35)
四、文件的读和写·····	(36)
五、文件的随机访问·····	(41)
六、系统调用unlink 和 fcntl ·····	(43)
第二节 标准输入、输出和标准 I/O 库·····	(45)
一、标准输入和输出·····	(45)
二、标准错误·····	(46)
三、标准 I/O 库概况·····	(47)
四、错误代码变量 errno ·····	(49)
第三节 多用户环境中的文件·····	(50)
一、文件主与有关标识符·····	(50)
二、文件权限和模式·····	(51)
三、文件权限与系统调用·····	(53)
四、文件可访问性的检测·····	(54)
五、文件权限和文件主的改变·····	(55)
六、文件建立屏蔽标记·····	(56)
第四节 文件的链接·····	(57)
一、概述·····	(57)
二、文件链的建立·····	(57)
三、文件链的删除·····	(58)
四、文件系统信息的获取·····	(59)
第四章 文件系统	
第一节 文件系统概述·····	(64)
第二节 文件目录·····	(64)
一、目录与 SVID ·····	(64)
二、目录的用户观点·····	(64)
三、目录的实现·····	(66)
第三节 目录的程序设计·····	(69)
一、目录项数据结构·····	(69)

二、当前目录的改变	(72)
三、目录的建立	(73)
四、根目录的改变	(75)
五、目录树的扫描	(76)
第四节 文件卷和特殊文件	(78)
一、文件卷概述	(78)
二、文件卷的装卸	(79)
三、缓冲存贮和系统调用sync	(80)
四、UNIX的特殊文件	(80)
五、块设备文件和字符设备文件	(81)
六、特殊文件与stat结构	(82)

第五章 进程控制

第一节 进程的建立与运行	(85)
一、进程的概念	(85)
二、进程的建立	(85)
三、进程的运行	(87)
四、数据和文件描述符的继承	(93)
第二节 进程的控制操作	(96)
一、进程的终止	(96)
二、进程的同步	(96)
三、进程终止的特殊情况	(98)
四、进程控制的实例	(98)
第三节 进程的属性	(103)
一、进程标识符	(103)
二、进程的组标识符	(105)
三、进程环境	(106)
四、进程的当前目录	(108)
五、进程的有效标识符	(109)
六、进程的文件长度限制	(109)
七、进程的优先数	(110)

第六章 进程通信

第一节 引言	(111)
一、进程之间的通信	(111)
二、新的进程通信机构	(111)
第二节 信号通信机构	(111)
一、信号及其类型	(111)
二、信号的处理	(114)
三、信号与系统调用的关系	(117)
四、信号的复位	(118)
五、用kill 发送信号	(119)

六、系统调用alarm 和 pause	(122)
七、例行程序setjmp和 longjmp	(125)
第三节 管道通信机构	(126)
一、管道概述	(126)
二、管道的程序设计	(127)
三、管道的长度和管道的关闭	(131)
四、管道中的read和 write	(133)
五、管道与系统调用exec	(136)
第四节 FIFO 通 信机构	(138)
一、FIFO及 其应用	(138)
二、FIFO的 程序设计	(140)
第五节 记录锁定	(142)
一、记录锁定的引出	(142)
二、用 lockf 实现记录锁定	(143)
三、用 fcntl 实现记录锁定	(148)
第六节 IPC通信机构	(151)
一、基本概念	(151)
二、消息的传送	(152)
三、信号量	(161)
四、共享存贮器	(168)
五、与IPC 有关的命令	(174)
第七章 终端管理	
第一节 引言	(175)
一、终端驱动程序	(175)
二、系统与终端之间的关系	(175)
第二节 UNIX中的终端	(176)
一、概述	(176)
二、控制终端	(177)
三、数据传输	(177)
四、正则模式和编辑功能	(178)
第三节 终端的程序设计	(179)
一、终端的打开与读写	(179)
二、例行程序ttyname和 isatty	(182)
三、终端特性的改变	(183)
四、参数 MIN和TIME	(186)
五、终端控制	(188)
六、终端与 SIGHUP 信号	(190)
第四节 程序 connect 的设计	(190)
一、总体描述	(190)
二、标题文件 和 main 函数	(191)

三、函数 connect	(194)
四、文件传输例行程序	(198)
五、connect 的使用	(202)
第五节 终端管理的发展	(203)
一、数据结构的变化	(203)
二、流的概念的提出	(203)
第八章 标准I/O库	(205)
第一节 引言	(205)
第二节 流与流操作	(205)
一、流和 FILE 结构	(205)
二、流的打开与关闭	(206)
三、流的单字符 I/O	(206)
四、恢复流的字符	(209)
第三节 标准输入输出与流	(211)
一、标准输入输出和标准错误	(211)
二、标准I/O 状态的查询	(212)
三、行输入与行输出	(213)
四、二进制输入与输出	(216)
五、随机文件访问	(218)
第四节 格式输入与输出	(219)
一、格式输出	(219)
二、格式输入	(224)
第五节 程序的执行与其它	(226)
一、程序的执行	(226)
二、其它标准I/O例行程序	(232)
第九章 屏幕处理和其它	
第一节 屏幕处理库概述	(234)
一、引言	(234)
二、curses库	(235)
三、curses程序的通用结构	(235)
四、terminfo库	(236)
第二节 curses例行程序	(237)
一、模式的设置	(237)
二、写字符和字符串	(238)
三、格式输出	(239)
四、光标的移动	(240)
五、键盘输入	(241)
六、屏幕字符的读入	(243)
七、屏幕编辑	(243)

八、视频属性的设置	(246)
九、窗口的建立和操作	(247)
十、程序 domenu	(247)
第三节 动态存贮管理	(251)
一、动态存贮器分配与释放	(251)
二、动态存贮器分配实例	(253)
三、系统调用brk和sbrk	(256)
第四节 时间和字符处理	(257)
一、时间处理	(257)
二、字符串和字符处理	(259)
附录A errno出错代码与信息	(263)
附录B UNIX系统的主要标题文件	(267)

第一章 绪 论

第一节 UNIX 概述

一、UNIX的历史与发展

1969年初,美国AT&T公司的Bell实验室退出了交互式分时系统 MULTICS (Multiplexed Information and Computing Service) 的设计,该系统运行在GE645 大型机上。1969年3月,该实验室的K·Thompson开始在DEC公司的PDP-7小型机上着手开发一个小型的通用分时系统,其目的在于开发一个廉价小型机上的良好程序设计环境,供Bell实验室内部使用。

1970年,这个命名为UNIX的系统开始在PDP-7机上运行,这是UNIX的最原始版本,它全部用PDP-7机的汇编语言写成。同年,它又被建立到较高档的PDP-11/20机上。

1971年,在对原始版本作适当修改后,正式形成了UNIX Version 1的程序设计文件,UNIX开始交Bell实验室内部使用。该版本中几乎包括了现代UNIX中除管道之外的所有重要思想。

1972年6月,UNIX的设计者为其增加了管道功能后,形成了UNIX Version 2。这时的系统仍然是用汇编语言编写的,所以系统的可移植性受到影响。

1973年,K·Thompson的伙伴D·M·Ritchie在B语言的基础上,重新设计了一种语言,首先在PDP-11机上实现。由于它是B语言的后继,因而取名为C语言。同年,K·Thompson和D·M·Ritchie用C语言改写了UNIX,由此形成了UNIX Version 5。事实证明,这次用C语言改写系统,对于UNIX的可移植性的提高,对于UNIX的迅速普及和推广是具有决定意义的一步。

1974年7月,K·Thompson和D·M·Ritchie合写的“UNIX分时系统”一文在美国的权威杂志——《ACM通讯》上正式发表,这是UNIX首次正式公布于众。

1975年3月,发表了UNIX Version 6。这种UNIX版本开始向外提供,并广泛配备于各大学的PDP-11机上。从此时开始,UNIX开始受到大学界的重视,不少学校对UNIX进行了扩充和再开发。

1978年,UNIX的设计者又对其进行了优化,形成了UNIX Version 7。该版本为UNIX进入商业界奠定了基础。1979年,在此版本的基础上产生了在VAX-11/780上运行的UNIX 32V。

1981年,AT&T公司发表了UNIX System III。从这时开始,UNIX不再采用版本(Version)号排列,而改为按系统(System)号排列。这说明了UNIX由Bell实验室内部使用,完全改为向社会提供。

1983年,AT&T公司又发表了UNIX System V。直至今,它仍是UNIX的最新版。值得一提的是,UNIX Version 8也在同年推出了,但是这个版本没有向外提供,仅

供Bell实验室内部使用。

二、UNIX的其它版本

除了上述由Bell实验室或其母公司AT&T提供的UNIX版本之外，还有许多其它版本。下面介绍其中最主要的几种。

美国California大学的Berkeley分校在UNIX的发展中起了重要作用。该校以UNIX Version 7为基础，推出了在VAX—11机上运行的Berkeley UNIX，该系统具有一系列以BSD (Berkeley Software Distribution) 为后缀命名为版本，其中最著名的有4.1BSD、4.2BSD和4.3BSD。这些版本对UNIX Version 7作了较大的扩充，它们在大学界非常受欢迎。它的文件存贮结构和进程通信等方面与当今的UNIX System V有较大的差别。但是，它所增强的一些内容，如curses屏幕处理库和vi编辑程序等，已被UNIX System V所采用。

美国的Microsoft公司意识到UNIX的巨大价值，1980年8月该公司宣布，它将在十六位微机上提供UNIX的变种——XENIX，作为UNIX的一种商业版本。XENIX运行的典型机种有：Altos 8600、586/986、1086/2086，Intel 86/330x，以及IBM PC/AT等。XENIX最早是基于UNIX Version 7开发的，为了使其能在十六位微机上运行并具有强大的竞争力，Microsoft公司对UNIX进行了一系列的更改和增强工作。添加了多种高级语言和实用程序，还配备了更多的网络通信软件和数据库管理系统，从而使XENIX不仅拥有高效率的软件开发环境，而且成了一个通用的应用系统。运行于IBM PC/AT、386机上的XENIX是根据UNIX System III和UNIX System V修改和扩充而成的，其功能更强大，性能也更完善。

三、UNIX的特点

UNIX系统能在不太长的时间内取得如此大的成功，其本身的性能和特点是决定性的因素。下面简要地分析UNIX系统的这些特点。

1. 吸取和发展了已有的先进技术

UNIX在设计思想和所采用的技术方面，对已有的较成熟的和先进的技术进行了精选、提炼和发展。UNIX从MULTICS及其开发工具CTSS (Compatible Time-Sharing System) 中吸取了极为丰富的内容，它删除或改造了这些系统中与基本功能关系不大的部分，吸收了它们的精彩部分。例如，UNIX的文件系统和用户界面等，都可以在这两个系统中找到它们的原型。

UNIX系统除了有选择地吸收已有的技术外，其总体设计思想也有所发展。这主要表现在操作系统的功能设计方面。在设计UNIX时，对系统功能作了仔细斟酌，着眼于向用户提供包括有许多工具而且便于综合应用它们的程序设计环境，也就是构成一个能够提供各种服务的基础。

2. 内核和核外程序的有机结合

UNIX系统在结构上分为两大层：内核和核外程序，即操作系统和实用程序两部分。UNIX的设计者对内核功能作了慎重考虑，对其数据结构和程序作了精心设计，使其非常精干简洁。因此内核体积很小，能够常驻内存，这就从根本上保证了系统能以较高的效率运行。

从内核中分离出来的部分,则以核外程序形式出现并在用户环境下运行。UNIX的系统设计特别注意了内核与核外程序的有机结合。内核向核外程序提供了充分而强有力的支持,核外程序则以内核为基础,灵活而恰到好处地运用了内核的支持。两者结合为一个向用户提供各种良好服务的整体,其功能可以与一些大系统相媲美。

3. 良好的用户界面

UNIX的用户界面是一种命令程序设计语言shell,它起着命令解释程序和全功能编程语言的双重作用。这种语言使用方便、灵活,并且易学。使用它不仅能提高在系统上进行工作的效率,而且便于扩充系统的功能。UNIX的有一些实用程序就是后来用户用shell语言编写的。

系统调用是用户在编写程序时可以使用的界面。UNIX不但在汇编语言中,而且在C语言中向用户提供这种界面。用户可以把它们看作是C语言的一部分加以应用。

4. 分级结构的可装卸的文件系统

UNIX具有一个分级结构的文件系统。整个文件系统形成从根目录表开始的树形分级结构。用户可以把自已的文件卷连入原有的系统上,以后还可以把它卸下来。这既能够扩大文件存储空间,又有利于安全与保密。

5. 文件和设备统一处理

在UNIX中,文件是无结构的字符序列。用户可以按需任意组织文件格式,对文件既可以进行顺序读写,也可以随机存取。UNIX把普通文件、文件目录表和外部设备都统一作为文件处理。它们在用户面前有相同的语法语义,使用相同的存取方法和保护机制。这样既简化了系统设计,又便于用户使用。

6. 丰富的核外系统程序

UNIX系统的核外部分包含有非常丰富的高级语言处理程序,软件开发工具、文本处理程序和系统实用程序。例如C、FORTRAN、APL、SNOBAL、YACC和LEX等。据AT&T公司称,至今在UNIX下运行的微计算机软件包已达1000多个。正是这些应用程序大大加强了UNIX的功能。

7. 具有良好的可移植性

UNIX的内核和核外程序基本上都用C语言编写,这使得系统易于理解、修改和扩充,并使它具有有良好的可移植性。这不仅意味着UNIX系统易于移植到别的硬件系统上,而且在UNIX系统下开发的应用程序也易于移植到其它配置了UNIX的系统上去,这正是UNIX得以普及和获得成功的关键之一。

四、UNIX的结构

1. 程序结构

我们以UNIX Version 7为例,来讨论UNIX内核源程序的结构。UNIX系统内核的源程序代码约18000余行(汇编程序约1050行),它们分为77个文件。这些文件按编译方式可分为以下三类:

(1) “.c”文件:这是C语言程序文件,共有41个文件。每个C语言程序文件中包含若干个子程序,这些文件可以独立编译。

(2) “.h”文件:这是标题文件,共有33个这样的文件。标题文件内含有C语言程序的全局变量、符号常数和数据结构说明,这种文件只能和C语言程序文件一起编译。

(3) “.s”文件：这是汇编语言程序文件，共有3个文件。它们约占内核总程序量的7%。这些汇编语言程序大部分与硬件关系密切，很难用C语言描述，或者是为了提高工作效率而没有用C语言描述。

2 系统结构

UNIX系统基本上分为内核和核外程序两部分。由于设计UNIX的时候正值对结构程序设计方法研究的开始，所以它的系统结构并不很理想，即它属于模块结构，模块间的关系比较复杂。图1—1为UNIX系统结构示意图。

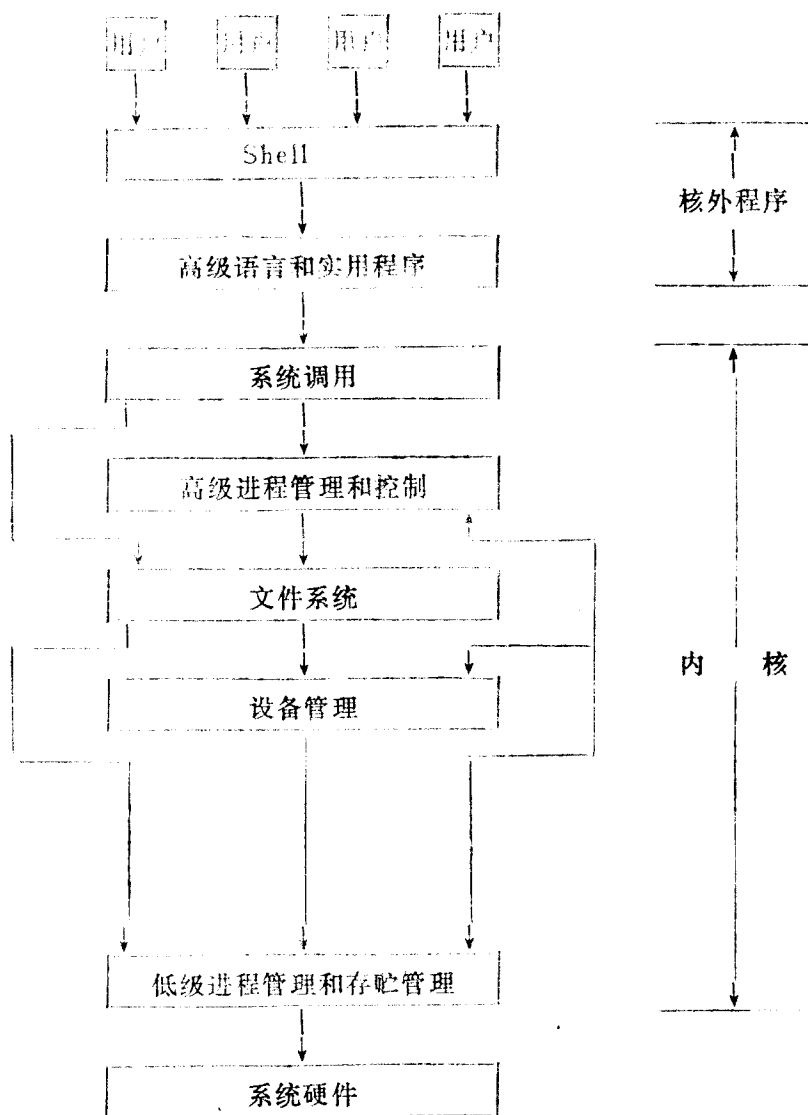


图1—1 UNIX系统结构示意图

从图1—1可知，UNIX系统中每个模块内的程序可以调用其它模块内的程序，这种调用不是绝对有序的，故还存在着一些循环调用关系。这就使得一个模块的正确性还要取决于其它模块的正确性。但是，由于模块结构具有系统效率高、代码紧凑等优点，使得UNIX内核的代码较少，模块之间的关系较易得到控制。

UNIX的内核部分包含了操作系统的主要功能。如存贮管理、进程与处理机管理、设备管理和文件管理等。虽然UNIX内核是模块结构，但从模块间的调用关系来看，除了少数模块外，大多数模块间仍然符合一定的层次关系。例如，文件管理处于设备管理之上，设备管理又处于存贮管理之上等。至于进程与处理机管理，从层次观点看，它们有的处于设备管理之下，如进程调度、进程通信等；有的处于较高层次，位于设备管理和文件管理之上，如进程管理和进程控制。内核的最外层是系统调用，它是UNIX内核的对外接口，是用户程序与内核之间唯一的接口。

UNIX的核外部分包括各种高级语言和实用程序。用户自编的可执行程序也属于这部分。所有核外程序都在shell的管理与控制下为用户服务。shell是用户与UNIX之间的接口。应当注意，在一般操作系统中，象shell这样的命令解释程序是内核的一部分，而在UNIX中，shell已不属于UNIX内核，而是在核外以用户态运行。shell根据用户输入的命令，找到相应模块中的程序，并为之建立进程执行之。

第二节 UNIX的标准化

一、UNIX标准化的必要性

在1983年之前，AT&T公司在UNIX的实现和开发方面几乎没有兴趣。在AT&T公司的许可下，许多硬件和软件厂商实现了它们各自的UNIX版本。

一方面，由于AT&T公司的不干涉态度，使得UNIX自行发展，并成为非专利的操作系统标准，而被广泛接受。另一方面，由于缺乏来自AT&T公司的控制，结果形成了许多不同的UNIX版本。例如，由于AT&T公司推出的一种UNIX版本缺少文件和记录的锁定功能，然而这些功能是商用系统所必需的。于是，硬件厂商和软件公司都在各自的UNIX版本中加入了这一功能，但是它们的实现互不兼容。结果是，在某个UNIX系统下设计的使用锁定功能的程序，只能在该系统下运行，若要在其它UNIX系统下运行，必须作相应的修改后才能正常运行。

由于以上原因，使得为UNIX开发的应用程序的移植工作发生越来越大的困难，以至影响UNIX的推广与发展。因此，UNIX的标准化工作已作为一个刻不容缓的问题，被提到议事日程上来了。

二、SVID标准

国际UNIX组织和AT&T公司于1985年宣布了一个文件，其名为AT&T System V Interface Definition，即SVID。这个文件在系统调用、例行程序库和实用程序等方面描述了UNIX System V。它给出了定义UNIX System V的基准，UNIX System V的所有提供者必须遵循这一基准。对于软件设计者来说，用SVID中所描述的基本操作编写的程序，其源代码可以移植到另一个UNIX System V系统上，而不管其支撑机种如何。

由于SVID的重要性，本书将基于SVID的第二版本。本书中所描述的绝大部分系统调用和例行程序均在SVID中有定义，书中的程序实例（除极少数专门说明的之外），都能运行于符合SVID标准的任何计算机系统。书中在描述到UNIX的实现时，将针对AT&T 3B2计算机上提供的UNIX System V Release 2.0进行描述。这个系统是AT&T公司开发的，

并完全符合SVID, 故它具有“正统性”。

三、POSIX规范

早在1980年, 美国的UNIX用户协会就开始了UNIX的标准化工作。与此同时, 美国的IEEE学会也成立了一个P1003标准化委员会。1985年, 这两个组织将UNIX的标准化工作合并起来共同开展。IEEE P1003标准化委员会制订出了一个可移植的操作系统规范, 称为POSIX规范。它包括可移植操作系统的规范, 命令语言shell和其它工具, 以及验证测试方法等。

1986年4月, IEEE提出了“试用POSIX规范”的标准草案。到1987年2月, 已有几十家大公司(包括IBM公司)和国际上很有影响的X/OPEN组织都表示支持这个规范。同时, 国际标准化组织ISO的TC97技术委员会正准备以IEEE的POSIX规范为基础, 制订一个操作系统的国际标准。

值得指出的是, SVID对UNIX System V的定义与POSIX规范是一致的。SVID是针对UNIX System V的, 而POSIX规范具有普遍性, 它是针对任何操作系统的, 所以两者之间没有矛盾。

第三节 基本概念

为了便于今后的叙述, 本节将集中介绍本书中要用到的一些UNIX的基本概念与有关专用术语。

一、文件系统

1. 文件

UNIX系统中的信息以文件来存放。UNIX的文件是由一串字符组成的信息的有序集合, 每个文件都由用户指定一文件名以资识别。

文件系统是负责存取和管理文件的信息处理机构。通常, 文件被存放在大容量磁盘上, 也可以把其它外部设备看作文件存储器。文件系统实现了按文件名存取文件中信息的功能。安放在一个磁盘驱动器上的一个盘组构成一个文件卷, 一个文件系统可由若干个文件卷组成。

2. 文件形成

UNIX文件从用户观点来看, 只有三类: 普通文件、目录文件和特殊文件。

(1) 普通文件 (ordinary file)

普通文件的正文是一个无特性的, 以512字节为一块顺序存取的字节序列。这种文件具有下列特点:

- 无记录概念, 一个文件相当于一个字符串, 但是换行符可插入文件, 以确定类似记录的子串。
- 文件的空间不预先确定, 即文件大小根据字符串的长度来定。
- 无明显的索引文件, 连续文件和串联文件等类型的区别。
- 几乎没有顺序和随机存取方式之区别, 用户可以移动文件读写指针, 预先确定下一次要读写文件之起始位置。