

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

# UNIX 应用教程

## UNIX Operating System

须德 许宏丽 编著

- 简明扼要阐述UNIX原理
- 全面准确介绍Shell编程技巧
- 讲解清晰详尽，配有大量示例



高校系列

 人民邮电出版社  
POSTS & TELECOM PRESS

TP316.81/235

21世纪高等

2009

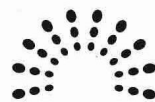
教材

21st Century University Planned Textbooks of Computer Science

# UNIX 应用教程

## UNIX Operating System

须德 许宏丽 编著



高校系列

人民邮电出版社

北京

## 图书在版编目 (C I P) 数据

UNIX应用教程 / 须德, 许宏丽编著. -- 北京: 人民邮电出版社, 2009. 11

21世纪高等学校计算机规划教材  
ISBN 978-7-115-21346-4

I. ①U… II. ①须… ②许… III. ①  
UNIX操作系统—高等学校—教材 IV. ①TP316.81

中国版本图书馆CIP数据核字(2009)第180950号

## 内 容 提 要

本书的编写以实用性为主要原则。首先有选择地讲解了 UNIX 操作系统的原理, 主要是讲解一般 UNIX 教科书上讲得不多或不够详细的内容; 然后重点是准确、清晰和详尽地阐述每条 Shell 命令的功能。通过大量的示例讲解 Shell 命令的使用方法, 目的是使读者能尽快地掌握 Shell 编程技巧。

全书共 8 章, 分别介绍 UNIX 操作系统的基本概念, UNIX 进程的组成、存储管理和调度管理, 文件系统的组织、实现和使用, vi 编辑器, 远程通信, 重点是详解 B Shell 和 C Shell 的各种命令、控制结构及编程应用示例。

本书可作为计算机科学技术专业的本科和研究生教材, 也可作为工程技术人员的自学参考书或查阅手册。

21 世纪高等学校计算机规划教材

## UNIX 应用教程

- 
- ◆ 编 著 须 德 许宏丽  
责任编辑 刘 博
  - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号  
邮编 100061 电子函件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
北京铭成印刷有限公司印刷
  - ◆ 开本: 787×1092 1/16  
印张: 16.5  
字数: 430 千字  
印数: 1—3 000 册
  - 2009 年 11 月第 1 版  
2009 年 11 月北京第 1 次印刷

---

ISBN 978-7-115-21346-4

定价: 29.00 元

读者服务热线: (010) 67170985 印装质量热线: (010) 67129223

反盗版热线: (010) 67171154

现今社会对人才的基本要求之一就是应用计算机的能力。在高等学校,培养学生应用计算机的能力,主要是通过计算机课程的体制改革,即计算机教学分层、分类规划与实施;密切联系实际,恰当体现与各专业其他课程配合;教学必须以市场需求为导向,目的是培养高素质创新型人才。

人民邮电出版社经过对教学改革新形势充分的调查研究,依据目前比较成熟的教学大纲,组织国内优秀的有丰富教学经验的教师编写一套体现教学改革最新形势的“高校系列计算机教材”。在本套教材的出版过程中,我社多次召开教材研讨会,广泛听取了一线教师的意见,也邀请众多专家对大纲和书稿做了认真的审读与研讨。本套教材具有以下特点。

### 1. 覆盖面广,突出教改特色

本套教材主要面向普通高等学校(包括计算机专业和非计算机专业),是在经过大量充分的调研基础上开发的计算机系列教材,涉及计算机教育领域中的所有课程(包括专业核心骨干课程与选修课程),适应了目前经济、社会对计算机教育的新要求、新动向,尤其适合于各专业计算机教学改革的特点特色。

### 2. 注重整体性、系统性

针对各专业的特点,同一门课程规划了组织结构与内容不同的几本教材,以适应不同教学需求,即分别满足不同层次计算机专业与非计算机专业(如工、理、管、文等)的课程安排。同时本套教材注重整体性的策划,在教材内容的选择上避免重叠与交叉,内容系统完善。学校可根据教学计划从中选择教材的各种组合,使其适合本校的教学特点。

### 3. 掌握基础知识,侧重培养应用能力

目前社会对人才的需要更侧重于其应用能力。培养应用能力,须具备计算机基础理论、良好的综合素质和实践能力。理论知识作为基础必须掌握,本套教材通过实践教学与实例教学培养解决实际问题的能力和知识综合运用能力。

### 4. 教学经验丰富的作者队伍

高等学校在计算机教学和教材改革上已经做了大量的工作,很多教师在计算机教育与科研方面积累了相当多的宝贵经验。本套教材均由有丰富教学经验的教师编写,并将这些宝贵经验渗透到教材中,使教材独具特色。

### 5. 配套资源完善

所有教材均配有PPT电子教案,部分教材配有实践教程、题库、教师手册、学习指南、习题解答、程序源代码、演示软件、素材、图书出版后要更新的内容等,以方便教与学。

我社致力于优秀教材的出版,恳请大家在使用的过程中,将发现的问题与提出的意见反馈给我们,以便再版时修改。

人民邮电出版社

[illegible]

本书是根据作者十几年来讲授 UNIX 操作系统的讲稿整理并扩充而成的。书中首先有选择地扼要讲解了 UNIX 的原理,而且主要是讲解一般 UNIX 教科书上讲得不多或不够详细的内容;然后重点讲解 Shell 的使用。对于简单易学的 Shell 命令仅仅是提示;对于难理解、不易掌握的 Shell 命令,则通过大量的示例,详解不同参数环境下的执行过程和执行结果,使读者能解除疑惑,清晰地了解该 Shell 命令的全部功能。对容易忽略或不注意的细节,我们给出了特别的提醒,目的是使读者能全面、准确地掌握 Shell 编程技巧。此外,在阐述过程中,我们还加入了一些评述或看法,这些观点是否恰当,请读者指正。

本书既可作为计算机科学技术专业的本科和研究生教材，也可作为工程技术人员的自学参考书或查阅手册。

作者  
2009年8月

# 目 录

|                      |    |                    |     |
|----------------------|----|--------------------|-----|
| 第 1 章 引论             | 1  | 2.5.2 创建新进程        | 50  |
| 1.1 操作系统的相关概念        | 1  | 2.5.3 初始化进程        | 54  |
| 1.1.1 计算机系统          | 1  | 2.5.4 进程的同步与终止     | 57  |
| 1.1.2 操作系统           | 3  | 习题                 | 59  |
| 1.2 UNIX 简介          | 7  | 第 3 章 文件系统         | 61  |
| 1.2.1 UNIX 的发展简史     | 7  | 3.1 概述             | 61  |
| 1.2.2 UNIX 的特点和性能    | 9  | 3.2 UNIX 文件系统的组织   | 65  |
| 1.2.3 UNIX 的结构       | 10 | 3.2.1 文件系统框架       | 65  |
| 1.3 UNIX 的优缺点        | 12 | 3.2.2 目录           | 69  |
| 1.3.1 UNIX 的优点       | 12 | 3.2.3 文件和目录的命名     | 72  |
| 1.3.2 UNIX 的缺点       | 13 | 3.2.4 文件和目录的显示     | 72  |
| 1.4 UNIX 使用入门        | 13 | 3.3 文件的内存映像        | 74  |
| 1.4.1 UNIX 的启动过程     | 13 | 3.3.1 索引节点         | 74  |
| 1.4.2 登录与退出          | 16 | 3.3.2 多重索引结构       | 75  |
| 1.4.3 常见命令           | 18 | 3.3.3 UNIX 打开文件机构  | 75  |
| 习题                   | 25 | 3.4 文件系统的实现        | 78  |
| 第 2 章 UNIX 中的进程      | 26 | 3.4.1 文件系统的磁盘管理    | 78  |
| 2.1 进程的引入            | 26 | 3.4.2 文件链接         | 80  |
| 2.1.1 引入原因           | 26 | 3.5 文件的使用          | 84  |
| 2.1.2 进程的定义          | 27 | 3.5.1 文件的创建        | 84  |
| 2.2 UNIX 进程          | 28 | 3.5.2 文件的打开/关闭     | 86  |
| 2.2.1 映像的概念          | 29 | 3.5.3 文件的读写        | 88  |
| 2.2.2 proc 结构        | 30 | 3.5.4 文件的随机存取      | 90  |
| 2.2.3 user 结构        | 32 | 3.6 管道通信           | 91  |
| 2.2.4 共享正文段          | 35 | 3.6.1 管道通信机制的引入    | 91  |
| 2.3 进程的存储管理          | 36 | 3.6.2 系统调用 pipe    | 91  |
| 2.3.1 进程在核心态下的虚拟地址空间 | 36 | 3.6.3 pipe 文件的读写操作 | 93  |
| 2.3.2 进程在用户态下的虚拟地址空间 | 37 | 3.6.4 举例           | 94  |
| 2.3.3 内存中进程映像的分布     | 38 | 习题                 | 97  |
| 2.3.4 虚—实地址映射        | 38 | 第 4 章 vi 和 Shell   | 99  |
| 2.4 进程调度管理           | 42 | 4.1 vi 编辑器         | 99  |
| 2.4.1 调度状态           | 42 | 4.1.1 vi 操作模式      | 99  |
| 2.4.2 UNIX 进程调度      | 45 | 4.1.2 基本编辑命令       | 102 |
| 2.4.3 进程状态列表         | 47 | 4.1.3 vi 小结        | 108 |
| 2.5 进程的创建            | 49 | 4.2 Shell 简介       | 109 |
| 2.5.1 进程树            | 49 | 4.2.1 Shell 家族     | 109 |

|                            |            |                        |            |
|----------------------------|------------|------------------------|------------|
| 4.2.2 常用 Shell             | 110        | 6.3.2 流控制语句的应用         | 197        |
| 4.2.3 Shell 间的转换           | 111        | 6.3.3 一个监视系统用户的程序      | 203        |
| 4.3 Shell 简单命令             | 111        | 6.3.4 Shell 解释程序的分析与设计 | 204        |
| 4.3.1 基本命令                 | 111        | 习题                     | 208        |
| 4.3.2 目录操作命令               | 117        | <b>第 7 章 C Shell</b>   | <b>209</b> |
| 4.3.3 文件操作命令               | 122        | 7.1 Csh 的启动与退出         | 209        |
| 4.3.4 操作权限设置               | 129        | 7.1.1 C Shell 的启动      | 209        |
| 4.3.5 命令分隔符                | 131        | 7.1.2 C Shell 的退出      | 210        |
| 4.3.6 输入/输出重定向             | 133        | 7.2 C Shell 变量         | 210        |
| 习题                         | 135        | 7.2.1 环境变量             | 210        |
| <b>第 5 章 B Shell</b>       | <b>137</b> | 7.2.2 用户定义变量           | 213        |
| 5.1 B Shell 程序概述           | 137        | 7.2.3 全局变量和局部变量        | 215        |
| 5.1.1 程序文件                 | 137        | 7.3 命令查询和别名            | 217        |
| 5.1.2 程序的执行                | 138        | 7.3.1 历史命令             | 217        |
| 5.1.3 程序的调试                | 138        | 7.3.2 别名               | 222        |
| 5.2 B Shell 变量             | 141        | 7.4 C Shell 程序         | 223        |
| 5.2.1 变量的命名与赋值             | 141        | 7.4.1 C Shell 程序结构     | 223        |
| 5.2.2 变量分类                 | 142        | 7.4.2 数值型数据处理          | 224        |
| 5.2.3 B Shell 特殊字符         | 150        | 7.4.3 分支结构             | 226        |
| 5.2.4 变量的替换                | 151        | 7.4.4 循环结构             | 229        |
| 5.3 输入与输出                  | 152        | 习题                     | 231        |
| 5.3.1 标准输入/输出              | 152        | <b>第 8 章 远程通信</b>      | <b>232</b> |
| 5.3.2 输入/输出命令              | 152        | 8.1 概述                 | 232        |
| 5.4 控制结构                   | 161        | 8.1.1 finger 命令        | 232        |
| 5.4.1 测试语句                 | 161        | 8.1.2 网络、主机和路由参数的设置    | 232        |
| 5.4.2 计算表达式值               | 163        | 8.2 用户终端间的通信           | 234        |
| 5.4.3 条件判断                 | 165        | 8.2.1 write 命令         | 235        |
| 5.4.4 循环结构                 | 169        | 8.2.2 talk 命令          | 235        |
| 习题                         | 174        | 8.2.3 wall 命令          | 236        |
| <b>第 6 章 高级 B Shell 编程</b> | <b>177</b> | 8.2.4 msg 命令           | 236        |
| 6.1 B Shell 函数             | 177        | 8.3 电子邮件               | 236        |
| 6.1.1 函数的定义                | 177        | 8.3.1 mail 邮件系统        | 236        |
| 6.1.2 函数的使用                | 178        | 8.3.2 pine 电子邮件        | 243        |
| 6.1.3 函数的编辑                | 182        | 8.4 ftp                | 246        |
| 6.1.4 函数应用举例               | 183        | 8.4.1 ftp 命令           | 247        |
| 6.2 选项和参数                  | 191        | 8.4.2 ftp 使用实例         | 249        |
| 6.2.1 case 处理选项和参数         | 191        | 8.5 telnet             | 251        |
| 6.2.2 getopt 处理选项和参数       | 192        | 8.5.1 telnet 命令        | 251        |
| 6.3 综合应用                   | 195        | 8.5.2 telnet 使用实例      | 252        |
| 6.3.1 个性化环境设置              | 195        | 习题                     | 253        |
|                            |            | <b>参考文献</b>            | <b>255</b> |

# 第 1 章

## 引论

UNIX 操作系统自 20 世纪 70 年代诞生以来,经历了数十年的风风雨雨,以它日臻完善的系统管理、调度技术、功能强大的命令系统和各种实用的编辑工具,以及系统运行的可靠性和稳定性,成为大、中、小型计算机和 workstation 各类计算机系统中的一个重要的操作系统。本章从不同角度理解操作系统概念,介绍 UNIX 操作系统的基本概念和命令。当阅读完本章以后,将能够:

- 了解计算机系统的相关概念;
- 了解 UNIX 的发展历史;
- 掌握 UNIX 的性能和特点;
- 掌握现代 UNIX 的结构;
- 掌握 Shell 的启动过程;
- 掌握 UNIX 基本的系统命令。

## 1.1 操作系统的相关概念

了解事物的过程大都是由大到小,由宏观到微观,由功能(原理)到具体算法和操作。因此,在学习操作系统具体内容之前,首先了解一下操作系统的相关概念。这里先介绍计算机系统,然后介绍计算机系统的组成部分——软件,最后概述软件中最重要、最基础的软件——操作系统。这样能使读者对操作系统在计算机系统的位置和作用有一个清晰的认识。

### 1.1.1 计算机系统

计算机系统是能被用户使用的全部硬件和软件资源的总称,即计算机系统是硬件资源和软件资源的总和(传统定义)。硬件资源如 CPU、存储器、I/O 等;软件资源如操作系统、数据库管理系统、应用程序(Word, Photoshop, ……)等。硬件是计算机系统的物质基础,软件是对硬件的功能扩充和完善。裸机只识别机器代码,即汇编指令,使用的格式和硬件的物理特性相关,而且功能有限。如执行一个循环,需要执行很多条机器指令,用户使用不方便,而且用机器指令编辑的程序不能移植。用户使用软件(如 C, SQL, PB, ……)后,不必再用机器指令,这时用户使用的不再是裸机,而是计算机系统。这是一个经过一层层功能扩充,功能很强的计算机系统。而操作系统就是对裸机的第一层扩充。由此可知,软件的最大效用是:把复杂的技术掩盖起来。例如,电视技术很复杂,但是有了控制面板,用户不必懂得这些复杂技术就能使用(操作)电视机,控制面板就是电视机的“软件”。图 1-1 所示为计算机系统与中国传统计算器算盘的对应关系。其



中，计算机裸机可类比于算盘，而计算机软件系统可类比于算盘的运算口诀。

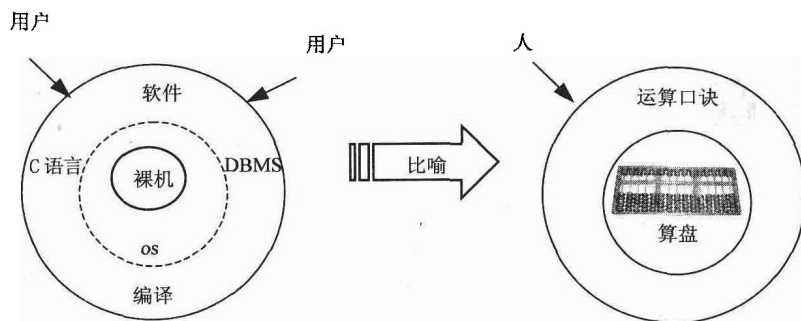


图 1-1 计算机系统概念图示

从图 1-1 可看出，操作系统是配置在硬件上的第一层软件，是对硬件功能的第一次扩充。其他系统软件如编译器、数据库管理系统以及大量的应用程序都是在操作系统支持下运行的。操作系统为不同用户的不同应用程序控制和协调资源的使用。

在计算机问世初期，“计算机”一词实际上只是指“计算机硬件”。进入 20 世纪 60 年代，由于程序设计技术的进步，才形成了“计算机硬件”和“计算机软件”的概念。硬件是计算机的“躯体”，软件是计算机的“灵魂”。因为程序、数据存放在柔软的纸带上，所以相对于硬邦邦的机器，人们称程序为软件，这是初期的概念，不准确。软件源于程序，但是慢慢地人们认识到文档的重要性，对软件的理解又深了一层，认为软件是程序和文档的总和。程序是让机器读的，文档是给人看的，只有完善的文档才能保证开发者，甚至非开发者能够修改已开发好的程序或软件。如果文档不完善，细节描述不清楚，将会给软件的维护（修改）带来很大的困难，花费巨额的工作量。例如，千年虫就是一个教训，仅仅把二位年份表示简单地修改为四位，如原来的 98 年修改成为 1998 年，就花费了几千万美元。

### 1. 软件

从学术的角度，软件存在个体、整体和学科 3 层含义。

#### (1) 个体含义

个体含义为计算机系统中某个程序及其文档，如文字处理软件 Word、售票软件、图像处理软件 Photoshop 等。

#### (2) 整体含义

整体含义为计算机系统中所有个体软件的总称。如某台计算机中所有具体软件：Windows、游戏软件、Word、Excel 等的总称。

#### (3) 学科含义

在研究、开发、维护和使用中，上述两种含义下的软件所涉及的理论、方法、技术所构成的学科，规范的名字应为软件学，但日常情况下常被简称为软件。如：你搞什么的？搞软件的。回答中所提到的软件是软件学，是对一个学科的描述。

### 2. 软件分类

软件可以分为系统软件和应用软件。

#### (1) 系统软件

系统软件是靠近硬件的一层软件，其他软件通过系统软件才能发挥作用。系统软件的特点是与具体应用领域无关，如操作系统、编译器。如前所述，操作系统在系统软件中是最靠近硬件的一层。

## (2) 应用软件

应用软件是特定应用领域的专用软件,如人口普查软件、教学管理软件、火车售票软件等。

## 3. 软件的开发技术

依据软件开发方法,软件不仅仅是程序的集合,还包括分析、设计、测试和质量控制等一系列文档。软件开发应考虑以下几个原则。

(1) 文档量高于代码量。一般为 7:3 或更高。

(2) 开发过程要规范统一。最基本的原则(或标准、要求)有:

- 代码+大量注释。注释包括程序功能说明,变量用途说明,重要的语句功能说明,注释最好能详细到算法中哪几条语句对应什么功能。

- 增量法(不是瀑布法)。即程序功能逐步增加,每增加一部分功能都在原代码的基础上增加部分代码,不存在原有代码的废弃。

- 测试驱动,先写测试程序,再写程序代码。

- 先写文档后编代码。

(3) 清晰的逻辑控制结构远比编程技巧更重要。

(4) 开发一个新软件,要尽量提高重用的代码比例,如达到 50%以上。要达到这一目标,平时应积累大量经过全面测试的功能子程序(或称为库函数),这样能降低开发难度,缩短开发周期,大幅度地减少新软件的错误。

开发软件一定要遵循良好的规则。这一思想的出发点是,软件的质量保证主要不是靠产品成型后的测试,而是要靠合理、正确的开发过程(步骤),只要按照这种开发过程开发软件,最后得到的将是质量高的软件产品。

必须提醒读者的是,计算机科学与技术(简称计算机)本质上是一门技术学科,技术概念会不断更新、充实(即与时俱进地变化)。比如对计算机系统的定义的变迁:从计算机系统是硬件和软件的总和,发展到计算机系统是硬件、软件和用户的总和,计算机系统的含义发生了很大变化。其中,用户的概念现在又包括:人、机器(计算机控制系统中的用户)、计算机(联网使用时的用户)。又如,在计算机领域关于媒体概念的变迁:20 世纪 80 年代前媒体指纸带、磁带、磁盘;20 世纪 90 年代,除纸带、磁带、磁盘外,又增加了感觉媒体:声音、图像、图形、文字的含义;现在,人们常使用的多媒体概念,包括感觉媒体、表示媒体、表现媒体、存储媒体、传输媒体和转换媒体。由上述可知,对一个技术概念的理解会随时间不断深化,与最初的定义比较,会发生很大变化。因此,在学习过程中,只有反复地从不同角度去认识、理解一个概念,才能真正掌握其精髓,才能有所发明和创造。

## 1.1.2 操作系统

### 1. 设置操作系统的目标

设计操作系统的目的是为了用户操作计算机时更加方便,计算机的运行效率更高,可更方便地扩充计算机的功能以及为计算机程序提供良好的移植性,即操作系统应具有方便性、有效性、可扩充性和开放性。

#### (1) 方便性

裸机配置了操作系统后,使计算机系统更容易使用。假如一台计算机没有装载操作系统,那么用户必须使用机器语言进行程序设计,因为计算机硬件只能识别 0 和 1 这样的机器代码,而这是大多数用户无法胜任的工作。安装了操作系统的计算机,用户可以通过操作系统提供的各种命

令使用计算机系统。比如编译命令提供了将高级语言编写的程序翻译成机器代码的功能，这就使得计算机变得更加易学易用，大大方便了用户。

因此，从用户的观点看，操作系统提供了良好的、一致的用户接口，隐藏（掩盖）了硬件系统的类型和机器命令的差别。通过使用 PC 可知，大家不必使用机器指令，可以使用高级语言编写程序，还可以用操作系统提供的命令，或在操作系统上运行软件来使用计算机。也就是说，操作系统向用户提供了一个方便的使用环境。

### （2）有效性

在计算机系统中包含有各类资源（如 CPU、内存、I/O 设备等），如何合理有效地组织这些资源是设计操作系统的主要目标之一。在未配置操作系统的计算机中，由于 CPU 运行速度远远高于 I/O 设备，使得 CPU 常常处于空闲状态而得不到充分利用；内外存中存放的数据也会由于缺少管理而处于无序状态，浪费存储空间。配置了操作系统，可以使 CPU 和 I/O 设备尽可能保持忙碌状态来提高运行效率，数据在内外存中的有序存放将提高存储空间的利用率。

因此，从资源管理的角度看，操作系统应该能够合理地分配和回收各种资源，使资源得到有效的使用，使程序有条不紊地运行，提高 CPU 和 I/O 效率（这是最主要的两项），当然还包括存储设备的效率。因此设置操作系统的目的就是充分、合理地使用计算机系统内包含的各种软件和硬件资源，按照需要和一定规则对它们进行分配、控制和回收，以便高效地向用户提供各种性能优良的服务。

### （3）可扩充性

随着计算机技术的高速发展，计算机硬件和体系结构也随之得到迅速发展。这就要求操作系统的设计必须具有很好的扩充性以适应这种发展。这就是说，操作系统应采用模块化的结构，以便于增加新的功能模块和修改老的功能模块。目前的操作系统是模块化的结构，且大部分用 C 语言编写，而 C 语言具有方便、高效、程序代码紧凑等特点，更方便了对系统的阅读和修改。

### （4）开放性

为了使不同厂家生产的计算机设备能够通过网络加以集成化并能正确、有效地协同工作，实现应用程序的可移植性和互操作性，就要求具有统一的开放环境，其中首要的就是操作系统的设计应具有开放性。

操作系统的开放性就是实现应用程序的可移植性和互操作性。由于大部分操作系统程序都是用 C 语言编写，虽然在效率上 C 语言比汇编语言稍差，但具有很多汇编语言所无法比拟的优点，它隐藏了具体机器的结构，即 C 语言程序不依赖于具体机器，从而使得操作系统易于移植到各种机器上。

操作系统最主要的目标是方便性和有效性。但实际上它们是矛盾的，过去由于计算机资源非常昂贵，因此主要强调有效性。近十几年来随着计算机技术的飞速发展，计算机硬件价格大幅度下降，因此更加重视方便性（特别是微型计算机）。

## 2. 什么是操作系统

操作系统目前已是常用词，在朗曼词典（Longman Dictionary）中的定义为“A set of programs inside a computer that controls the way it works and helps it to handle other programs.”。这表明操作系统是计算机用户和计算机硬件之间起媒介作用的程序。操作系统的目的是提供用户运行程序的一种环境，使用户在此环境下能方便地使用计算机系统，有效地使用计算机硬件。

操作系统是计算机系统的一个重要组成部分，其结构如图 1-2 所示，其中计算机软件系统可以分为操作系统和应用软件两层，也可以分为操作系统、系统软件 and 应用程序三层。

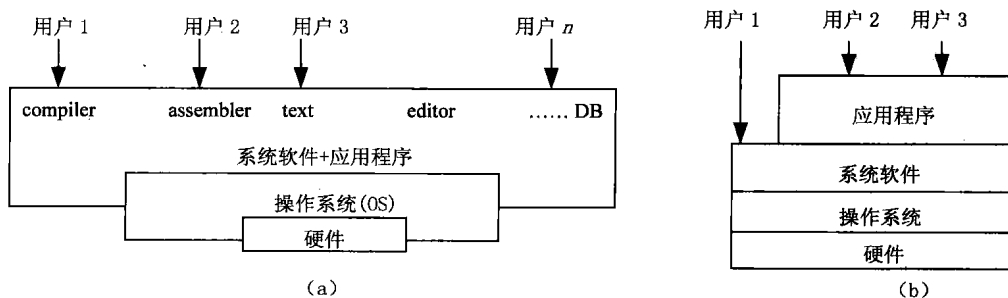


图 1-2 计算机系统结构

从不同的角度来看操作系统，可以获得对操作系统的不同的认识。

#### (1) 资源的观点

从资源的观点看，操作系统被看做是资源分配器。计算机系统拥有解决用户问题的各种资源：CPU、存储器、磁盘、I/O 接口、数据等。操作系统是这些资源的管理者，按照需求和资源的可用状态，将资源分配给某个具体用户和程序。

管理内容包括：

- 资源的当前状态：数量和使用情况。
- 对资源的操作：分配/回收和访问、管理策略。

管理目标包括：

- 为用户提供简单的操作方法，如用户只需提出所需空间的大小，不必指明存储地址，不必编写硬件驱动程序等。
- 有效、公平、安全地运行。因为资源需求总是大于资源量，使用资源还存在冲突，所以要决定哪些请求能满足（仲裁），哪些请求需要等待处理，以保证计算机系统能正常地运行。
- 防止死锁，使程序有条不紊地运行。公平、安全在多用户系统中尤为重要。

从资源的观点出发，操作系统可分为 5 大功能：作业管理、CPU 管理、存储管理、I/O 管理和文件系统管理。传统的教科书都是按照这一思路来编写的。近期一些教材书中已不再讲述作业管理的内容。

#### (2) 控制的观点

从控制的观点看，操作系统可以被当作设备、程序运行的监控器。早期的操作系统亦称为“监控程序（监督程序）”。监控程序的目的是为了防止出现程序运行错误和计算机使用错误。其中很重要的是 I/O 设备（因为特性各异，使用复杂）的监控和 CPU 的运行控制（多用户运行环境下）。

#### (3) 服务的观点

操作系统为用户提供最基本的、公共的服务，包括编辑功能、空间分配、编译源程序、文件读写等服务。从这一观点出发，操作系统也称为计算机的管理者（或计算机系统的经理）。例如，旅馆的经理为客人提供最基本的公共服务，如找空房、打扫卫生、打水、结算等。旅馆经理的管理功能方便了旅客，提高了空房利用率。从服务的观点看，操作系统也像旅馆的经理一样管理和控制计算机的运行，达到方便用户、提高资源利用率的目的。

#### (4) 软件的观点

把操作系统看做是一个软件，包含了软件的内在和外在特性。

外在特性：操作系统用什么语言实现（汇编语言、C 语言等），提供何种用户界面（命令

界面或图形界面)。

内在特性(即结构):操作系统由几部分组成,每部分的功能是什么,各部分之间如何联系等。

### (5) 虚拟机(Virtual Machine)的观点

从操作系统内部结构来看,操作系统分成若干层,每层完成一个特定功能,并对上一层提供支持,逐层扩充功能组成一个整体,最终显示给用户的是功能强大的虚拟机。

### (6) 其他观点

一个操作系统的最基本目标是使用户能够运行它们的程序,并让用户解决问题比较容易。裸机不易操作,当开发应用程序时,需要公共操作,如 I/O、edit 这些公共功能,这些公共功能分解成一个个系统调用,所组成的系统调用组成一个软件即操作系统。所以也有人称操作系统为系统调用的集合。

操作系统没有完善、准确的定义,应包括哪些部分也没有公认的定义,通常只是从操作系统在整个计算机系统中的地位 and 所起的作用给操作系统一个非形式化的描述。一个比较流行的操作系统的定义是计算机上任何时候都要运行的程序(通常称为 kernel)。这是一种不严格的说法。比较准确地说,操作系统可以定义为:控制和管理计算机资源,合理地各类作业(程序)进行调度(组织计算机工作流程)以及方便用户的程序集合。

当选购一个操作系统时,必须考虑经销商能提供什么功能,有何资源限制。例如,各操作系统所需空间和提供的性能差异很大,有的要求空间小于 1MB,但没有全屏编辑,有的需要几百兆存储空间,配有图形窗口系统。计算机的价格是否合理,要由应用程序的价值来判定。买计算机是为了解决某领域的具体问题,除了解决该问题以外的计算机功能都是系统开销。因此,计算机系统的功能刚好满足用户的需求是最低标准,如果计算机系统的功能超过需求太多又太浪费,需求和功能之间需要权衡一下。操作系统由哪些部分构成正成为一个重大的问题,某国司法部起诉某公司的操作系统包括太多功能,侵害了应用程序经销商的利益,是不公平竞争,就是一个例证。

## 3. 资源抽象

资源抽象和共享是操作系统的两个主要方面,尤其是资源抽象,下面从磁盘操作入手描述这一概念。如果从内存复制信息到磁盘,用户需要编写 3 条命令:

```
load(block,length,device)    //从内存(内存地址为 block)复制到磁盘缓冲区(device buffer)
seek(device, track)           //磁头移到磁盘的指定区(track)
out(device, sector)           //从 device buffer 写入磁盘指定区(track)的扇区(sector)
```

对于确定的磁盘进行上述操作,则写为如下命令形式:

```
load(block, length,D)
seek(D,236)
out(D,9)                      //数据写入磁盘 D 的 236 磁道的 9 扇区
```

将上述操作简单抽象写成一个过程,如下:

```
void write(ch *block, int len, int device, int track, int sector)
{
    load(block, len, devicehi);
    seek(device, track);        // track is physical number
    out(device, sector);        // sector is physical number
    .....
}
```

```
}
```

使用上述过程定义，完成从内存复制信息到磁盘的操作可以描述为：

```
write(block, 100, device, 236,9)
```

再进一步抽象：(track, sector)用非负整数表示，磁盘用逻辑地址表示，则上述操作表示为：

```
write(block, 100, device, 3788) //3788 为逻辑数，由系统转换为 (236, 9)
```

再对此描述作进一步抽象：磁盘抽象表示成文件，内存中要复制的信息存放在变量 datum 中，写入磁盘的首地址是文件指针(offset)指示的地址，offset 转换为物理地址（如 3788 转换为 (236, 9)）由系统完成，则上述操作被进一步简化描述为：

```
fprintf(fileID, "%d", datum)
```

#### 4. 资源共享

共享有两种方式，一是同时共享，二是分时共享。分时共享，即轮流共享。因为计算机运行非常快，它迅速地从一個作业转换到另一个作业，造成一种假象：计算机正在同时执行多个任务，这种“同时执行”称为“并发”。并发使得用户感觉到提交的任务被及时处理。对一个分时系统来说：

- 微观上——程序轮流占有 CPU。
- 宏观上——多个程序并发运行。
- 局部看——每个程序按程序指令串行、顺序执行。
- 整体看——多个程序穿插运行。

操作系统从宏观、整体角度讨论问题。因此，操作系统讨论的是分时共享，并发操作。分时系统中，CPU 轮流执行每个作业，因为轮换很快，所以一个用户的使用不影响其他用户使用，每个用户在感觉不到等待了很长时间的情况下，和自己的程序实现了交互。分时系统能得到成功应用的原因是：计算机交互主要是键盘/鼠标，比如用户从键盘每秒键入 7 个字符，已经很快了，但 CPU 速度为几百兆，用户动作和命令执行速度相比很慢，CPU 处理能很快地从一个用户转到另一个用户，响应时间通常不超过 1 秒，用户的感受是独占了 CPU。

分时操作系统是使计算机得到广泛应用的基础，是操作系统发展史上的重大里程碑。分时是多道程序设计的一个逻辑扩展。多道的概念是操作系统在内存同时存放多个作业 (jobs)，磁盘上开辟一个作业池 (job pool)，存放大量的作业，内存中的作业是其子集。操作系统选择内存中的一个作业（如 job1）运行，当 job1 执行 I/O 时，操作系统选择另一个作业(如 job2)运行。当 job1 完成 I/O 操作后，再投入运行，这样 CPU 总是处于忙碌状态。多道的概念在日常生活中也有类似的例子，如律师同时接几个案子，当一个案子等待审判时，律师就处理另一个案子。多道操作系统比较复杂，涉及作业调度、内存管理、多道程序的存放和 CPU 调度等多个方面。

## 1.2 UNIX 简介

### 1.2.1 UNIX 的发展简史

UNIX 操作系统是一个通用的、交互式的分时系统。它是美国贝尔实验室的 K.Thompson 和 D.M.Ritchie（两人是 1983 年图灵奖得主）共同研制成功的。在 20 世纪 60 年代后期，贝尔实验室

为 GE (通用电气) 和 MIT (美国麻省理工学院) 研制了一个多用户操作系统, 称为 Multics。1969 年 3 月, Multics 项目被中止, 一些主要开发人员开始寻找其他有意义的项目继续发展下去。其中之一就是 Thompson, 他编写了一个游戏程序 Space Travel, 并将其安装到一个废弃的 PDP-7 上运行。PDP-7 没有程序开发环境, Thompson 必须在 Honeywell635 上编译, 然后再装在纸带上输入 PDP-7。为了使 Space Travel 的开发方便, 他和 Dennis Ritchie 开始在 PDP-7 上研制运行环境, 第一个部件是简单的文件系统 (FS, File System), 后来发展为 s5fs 的系统。接着 Thompson 又研制了进程子系统、一个简单的命令解释器 Shell (即后来发展为 B Shell) 和很少的一组使用程序 (utility), 他们将其命名为 UNIX (所以早期的 UNIX = FS + processer + Shell + 少量的 utility)。1974 年, Thompson 和 Ritchie 发表了第一篇 UNIX 论文“The UNIX Time Sharing System”, 在 ACM Symposium on OS(1973.10)上介绍, 在 Communication of the ACM 上发表 (1974.7), 这是 UNIX 第一次向外界介绍。

从 1969 年开始, UNIX 被安装在几十种硬件平台上, 厂家、大学和研究机构推出了多种版本, 它从很小的程序集合已变成使用在广泛环境的通用操作系统, 运行平台从很小的嵌入式处理器、工作站、台式机, 一直到高性能的多 CPU 系统。最早的 UNIX 版本是美国贝尔实验室研制的, 现在有了多种版本, 虽然大多都支持核心的接口, 但是内部实现、接口的详细定义和各自增加的功能有所差别。UNIX 可以分为基础版本(Baseline Release)和商业版本 (Commercial Implementation) 两大类。基础版本(Baseline Release)包括: Novell 的 SVR4(System V Release 4)、Berkeley 的 4.xBSD (Berkeley Software Distribution) 和 Carnegie - Mellon 的 Mach。商业版本(Commercial implementation)包括: SunOS(4.2BSD based)、Solaris(SVR4 based)、Digital UNIX、HP-UX(HP)、AIX(IBM)等。

### 1. BSD

BSD 版本最早是伯克利 (Berkeley) 大学的一些研究生 (如 Bill Joy, 他是 C Shell 的创始人) 为 UNIX 开发实用程序 (如 Vi), 并把这些实用程序作为附加功能和 UNIX 捆绑成的一个包, 称为 BSD。BSD 的一大贡献是虚拟机 (VM) 的定义。UNIX 首先移植在 VAX 机, VAX-32 位允许 4GB 寻址, 但机器实际只有 2GB 内存; 由此引发了设计基于页交换的 VM。BSD (Berkeley System Distribution) 主要用于科学、工程计算, 对 UNIX 的主要贡献: 虚存、Vi、B Shell。变种 BSD UNIX 在 UNIX 的历史发展中具有相当大的影响力, 被很多商业厂家采用, 成为很多商用 UNIX 的基础。最新版本是 1993 年 6 月研制成功的 BSD4.4, 因为研究人员解散了, BSD 没有再发展。研究小组结束 UNIX 的研究工作的主要原因是:

- 资金不足。
- BSD 特点、功能在许多商用系统中已有, 存在的必要性日趋减弱。
- UNIX 系统太大、太复杂, 一个小组无法再维护它。

### 2. System V

AT&T 开发了许多版本的 UNIX, 其中 System V 最著名, 很多商业版本的 UNIX 都基于 System V, 因为它有许多新功能, 如 VM、进程间的通信功能 (如内存共享、信号量、消息队列)、远程文件共享、共享库、Net 规程。

XENIX 是 Microsoft 在微机上开发的 UNIX 版本。AT&T 为了解决不同版本、不兼容等问题, 和 Microsoft 达成协议, 把 XENIX 版本的功能移植到 System V, 变成 System V/386 R3-2。1983 年正式公布 UNIX System V, 1989 年 11 月的 SVR (System V release) 4.0 是以前的 System V、Sun OS、BSD4.3 和 XENIX 融为一体的产物。1991 年推出安全性增强版 SVR4.0 ES, 1992 年秋推出 SVR4.2 版。随后, System V 成为事实上的 UNIX 工业标准。

### 3. Mach

UNIX 的流行吸引了许多计算机公司, 这些公司纷纷推出自己的商业化版本, 它们或基于 AT&T 的 System V, 或基于 Berkeley 的 BSD, 并在此基础上增加自己特有的功能, 移植到自己的硬件上。

UNIX 流行的主要原因是它虽然规模小、简单, 却提供了许多有用的实用程序。但是当加入了越来越多的功能时, 内核变大、变复杂, 日益笨拙, 许多人感到 UNIX 偏离了它最初成功的原则。20 世纪 80 年代中期, Carnegie-Mellon 的研究人员开始研制新的 OS-Mach, 目标是开发微内核 (内核的精简版本), 提供小的、简单的服务和移植, 在用户层实现操作系统的其他功能。Mach 支持 UNIX 编程接口, 适用于单 CPU、多 CPU 和分布式环境, 因为是从头开始研制, 所以希望规避使 UNIX 苦恼的许多问题。Mach 和 AT&T License 无关, 是商业版 OSF/1 NextStep 的原型基础。

现有的各种 UNIX 版本都不可能不加修改地在各种机型上运行。System V 和 BSD 在许多方面不同, 公司商业化时再增加新功能, 使两者更不可移植。这种形式促成了标准的建立, 但是, 各公司各自建立自己的版本, 形成了多个标准, 比较公认的是 IEEE POSIX (portable OS based on UNIX), 主要原因是它近似地是 SVR3 和 4.3BSD 的核心部分的混合物, 不偏向某个特定的 UNIX 版本。

## 1.2.2 UNIX 的特点和性能

### 1. UNIX 是对已有技术的精选、提炼和发展

UNIX 操作系统并没有很多标新立异的东西, 而是吸取了 MULTICS 和 CTSS 的经验和成果, 这是两个在操作系统发展史上起过重大影响的操作系统, Thompson 和 Ritchie 参加了此项工作, 例如, UNIX 中的文件系统、用户界面的原型就是这两个系统的相应部分。

过去的操作系统希望包罗万象, 为用户做每一件事, 使系统庞大、复杂, 使得操作系统的理解和维护十分困难。这一现象和开发 MIS (管理信息系统) 类似, 用户希望什么都自动化, 菜单一层又一层地展开, 用户无需学习就能使用, 结果使得 MIS 系统庞杂, 用法烦琐, 最后导致因为难以使用、效率太低而废弃。UNIX 吸取了这一教训, 对系统功能进行了仔细斟酌, 着眼于向用户提供开发工具, 为应用建立开发环境, 即提供基本的、公共的服务基础, 便于用户综合运用。UNIX 核心系统小、易学, 将使用的灵活性留给用户。核心系统和运行库虽然足够小, 但是功能很强, 允许用户在此基础上建立复杂系统。

### 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。

#### (1) 内核

内核是 UNIX 系统的核心, 包括文件系统、进程管理、存储管理和设备管理。系统的其他部分调用内核来执行服务。

#### (2) 核外程序

凡是能从内核分离的都归结为核外。

内核小、高效等特点使得内核程序能常驻内存。内核为系统调用提供了充分、强有力的支持。核外程序在用户环境下运行, 内核与核外程序联合向用户提供服务。

内核的功能是经过仔细斟酌的, 采用的算法也被反复推敲, 采用的数据结构、程序都是精心设计的, 符合软件设计规律——即常用软件必须高效, 而一次性软件只需要功能满足需求即可, 不用太多考虑运行效率。



### 3. 良好的用户界面

用户界面功能齐备，易于扩充、修改。UNIX 提供两种界面：①用户界面 Shell，Shell 是命令程序设计语言，可在终端使用，也可以编程。②系统调用，可以在汇编语言和 C 语言程序中使用，其他操作系统中的系统调用只能在汇编级使用。

### 4. 文件系统可随意装卸

UNIX 拥有一种简单、小巧、组织灵活的文件和目录系统，支持用户高效、有序地组织和管理文件。UNIX 文件系统由基本文件系统和子文件系统（对应某个硬件存储设备）构成。子文件系统可以随意拆卸和安装。

### 5. 文件和设备的统一处理

UNIX 将输入输出设备作为文件处理，使用输入输出设备的方式（语法和保护机制）同普通文件是一样的。这意味着程序的输入可以来自任何设备和文件，而程序的输出也可以写入任何设备和文件，借助 UNIX 的这种特性简化了外部设备的使用，可以方便地控制文件的输入和输出。

### 6. 丰富的核外程序

UNIX 拥有大量的实用程序和用于开发软件的工具性软件，如 Lex 可用来编写词法分析程序，YACC（Yet Another Compiler - Compiler）可用来编写语法分析程序。

### 7. 可移植性良好

UNIX 内核和核外程序基本上用 C 语言编写，系统易于理解、修改、扩充，可移植性好。可移植性有两个含义：①UNIX 内核易于移植到别的硬件系统。②UNIX 下开发的应用程序易于移植到其他配有 UNIX 的计算机上。正是因为 UNIX 有很好的可移植性，使得 UNIX 广泛地运行在各种品牌的大型、中型、小型计算机和工作站上。

## 1.2.3 UNIX 的结构

UNIX 操作系统结构主要由 3 大部分组成。

#### (1) 内核 (Kernel)

Kernel (UNIX 内核) 是 UNIX 操作系统的核心，指挥调度 UNIX 机器的运行，直接控制计算机的资源，保护用户程序不受错综复杂的硬件细节的影响。

#### (2) 外壳 (Shell)

Shell (UNIX 外壳) 是一个 UNIX 的特殊程序，是 UNIX 内核和用户的接口，是 UNIX 的命令解释器。

目前常见的 Shell 有 3 种。

- Bourne Shell (B Shell、bsh 或 sh) 是最老的，使用最广泛的，每个 UNIX 都提供。
- Korn Shell (K Shell 或 ksh) 是对 B Shell 的扩充，兼容 B Shell。
- C Shell (csh) 格式像 C 语言。功能强，命令记忆稍难，在大学和学院中较为流行。

#### (3) 工具及应用程序

工具及应用程序包括各种语言处理程序，以及用户自己编写并经编译、连接处理而形成的各种可执行目标程序等。如常见的文本编辑器：Vi、Emacs 和 UNIX 的电子邮件系统 pine 等。UNIX 这些实用程序都在 Shell 命令语言解释程序的管理、控制下为用户服务。

Kernel 是 UNIX 中唯一不能由用户任意变化的部分，它大致分成存储管理、进程管理、设备管理和文件管理等几部分。20 世纪 70 年代到 80 年代初，UNIX 内核不是非常灵活，只支持一种类型的文件系统、一种调度策略、一种执行文件格式 (a.out)。灵活性主要体现在，不同的外设可