HOPE

HOPE COMPUTER COMPANY LTD.

UNIX SHELL
程 序 设 计

Shell是UNIX环境中提高生产量的关键

北京希望电脑公司

UNIX SHELL 程序设计

赵庆林
俞　涛　　编译
陈晓龙

北京希望电脑公司
一九九一年十月

目　录

附录

前言

本书如同禁闭的鹦鹉螺的外壳，在你从 Shell 程序设计的新手到资深用户的成长过程中保护你。这样，你将学习 Shell 的基本点，并沿着这条通向知识和智慧的道路前进。在 Shell 中如同在生活中一样，成长过程也分成三个阶段：儿童，青少年，成年；学徒，熟练工人，大师；本书也一样——新手，用户，资深用户。十四套知识和经验将用各种实例来指导你。

Shell 解开了 UNIX 的威力。它假设用户能够而且将做各种非凡的事情。Shell 教你提高生产率：一行 Shell 代码可以做 100 行 C 代码的工作。Shell 通过组合已有程序来教程序设计，而不是通过编写复杂的用户软件。它用支持象面向对象程序设计这样的高级概念的方法来教重复使用(reuse)。

为了使每一个人都能从使用 Shell 中获得最大的好处，本书被设计为涉及从初始的 Shell 各种用法到资深用户状态的所有内容。主要有三个部分：

1. 新手的 Shell
2. 用户的 Shell 程序设计
3. 资深用户的 Shell 程序设计

第一部分介绍从文件到进程的各种成员，以及 Shell 的威力。它教你如何使用 Shell，从简单命令到十分复杂的程序设计。

第二部分教你如何用这些基本技能创建完整的应用。人们每天都做可以从自动化获益的事情，但并不要求一个复杂的数据库系统。本书的这一部分将教你如何掌握用 Shell 来自动处理日常工作。而且这也为以后精通 Shell、建立更为复杂的应用打下基础。

第三部分则步入软件开发和系统管理员的世界。这一部分详细讨论快速建立原型和其他重要的软件工程活动。重点放在 UNIX，以及 Shell 的各种变化。

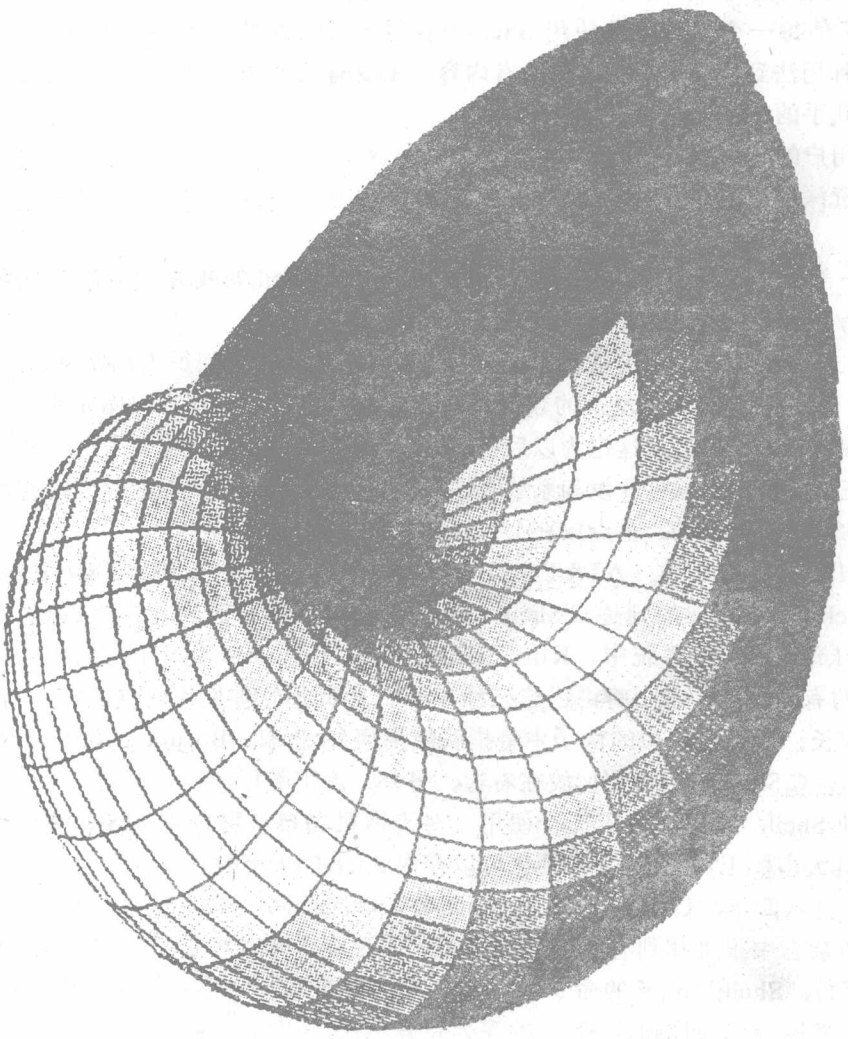
UNIX 主要有两种：AT&T 和 Berkeley。AT&T 贝尔实验室的有 Bourne Shell 和 Korn Shell。Berkeley 的则是 C Shell。Bourne Shell 是最普通的。C Shell 则广泛存在于所有的 Berkeley UNIX 系统中。Korn Shell 则在保留 Bourne Shell 的功能的同时，增加了 C Shell 的有用功能。本书将讲述这三种 Shell。因此不管你使用何种环境，本书总能适合于你的成长。而且在书中的例子中将指明使用何种 Shell。Borune / Kone Shell 的例子被放在左边，C Shell 的例子则被放在右边。另外，大多数厂商在他们的 UNIX 系统中都提供这三种 Shell，供用户按需选择使用。对于微机用户，可以在 MS-DOS 中使用 Korn Shell 及其大多数工具。PC 用户现在可以存取 Shell 的大多数功能。

在一九八四年，UNIX 还只是一个四个字母的单词，Shell 就更少有人知道了。当时在期刊和杂志中很难找到有关它们的介绍。而现在各种参考资料随处可见。随着 UNIX 的日益流行，Shell(UNIX 的命令语言)也逐渐流行起来。我将在本书中解释原因。

本书将逐步引导你到达 Shell 的深处及其用法。相信你通过阅读会从使用 Shell 获得很多好处，使你的工作更加高效。

第 一 部 分

新手的 Shell



第一章 Shell 的威力

Shell 是在 UNIX 环境中提高生产率和质量的关键．Shell 可以自动处理重复性的任务，找出何处留下了东西，在你午餐或休息时进行工作，同时也是其他节省时间的活动的源泉．Shell 的使用可以成倍地提高生产率和效率．Shell 是通过让你创建工具处理各种任务来达到这些结果的．它让你构造应用程序、程序、过程和工具的原型．可以用于提高建立工作原型的速度，使得可以提供确切满足要求的东西，或者全部扔掉重新开始，从而可以十分灵活地创建所需的适当工具或应用程序，而不必进行许多编码、编译、测试．几乎无需人工地处理各种任务，因为 Shell 几乎可以解决任何问题．

1.1 为什么要用 Shell?

在我们进入 Shell 的核心之前，首先让我们来看一下使用 UNIX Shell 的一些关键原因．

1. 越来越多的数据以机械化的格式存在于世界各地．但是对于那些想从该数据获益的人，则它必须被转换为信息、知识和智慧(见图 1.1)．数据只能以下列少数几种方法变成这些可理解的高级形式:
   - ☐ 数据的选择(selection)对数据进行分门别类．它演示信息的关键部分．Shell过滤程序完成这一功能．
   - ☐ 数据的组合(combination)为两组或两组以上数据创建collision course，同时信息被创建，知识被获取．其他 Shell 命令帮助清理信息和数据，以获取更进一步的知识．
   - ☐ 判定(decisions)和规则(rules)帮助我们进一步分析结果信息来获取知识．Shell控制结构帮助评价信息以创建知识．
   - ☐ 偶然运气(serendipity)使人以完全无法预计的方法组合思想、数据和信息．Shell 命令提供如此多的方法来观察和评价已有信息，这样偶然运气是可以保证的．
2. UNIX是第一个完全集成的CASE(计算计辅助软件工程)工具箱．工具的集成是 UNIX Shell 的基石．UNIX 将继续是 IPSE(integrated project support environment，集成项目支持环境)选择的平台．
3. 在运行单个程序的个人计算机平台中，不可能处理软件工程和开发中的复杂任务和其他与人类紧密相关的任务．
4. Shell是一个完整的程序设计语言，具有:
   - ☐ 变量
   - ☐ 条件和循环结构
   - ☐ 可剪裁的用户环境

   UNIX Shell是原始的快速建立原型工具，教你象模块、再使用和通过组合(com-

position)而不是编码来开发这样的关键概念。Shell 工具库是世界上除了一些 FORTRAN 的数学库以外最广泛地被重复使用的库。UNIX 的哲学是：在其他人的工作上开始。

5. UNIX Shell是早期的第四代程序设计语言(4GL)之一。所有应用程序都可以在 Shell 中迅速而有效地建立。比较一下 Shell 的威力：Shell 中的一行在 c++中可能需 10 行或更多，而在 C 或 COBOL 中则可能需要 100 行或更多。有些人抱怨 Shell 的语法太蠢笨，但是实际上并不比其他第四代程序设计语言多。

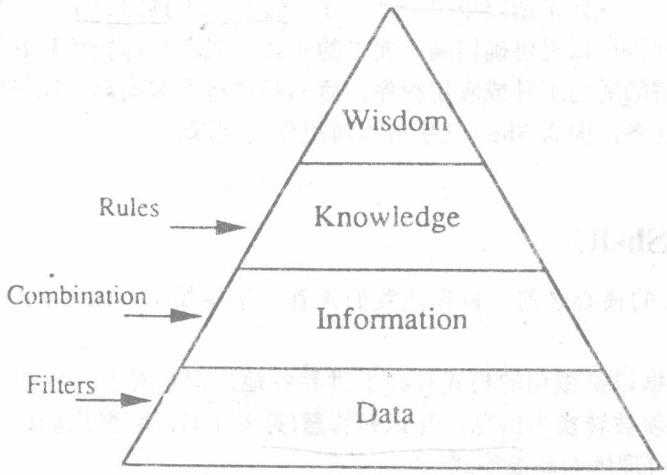


图 1.1：数据的发展

6. 机器比人更为便宜。可以通过为新用户创建看上去类似的环境并且开发随着用户的成长同时成长的环境，来优化对人员的投资。

1.2 Shell 有什么东西适合我?

你可能会说“那又怎么样呢？我以前也听说过这些要求。”或者你是用户而不是系统开发者。“Shell 里面有什么东西能适合我呢?”请看下面几点：

1. 你是否想更为成功、自信和高生产率；受到更多赞许；在工作中更为自豪？Shell 是个人满足和成功的关键。它使得你可以用几小时或几天来组合应用程序，而在常规环境中则可能需要几个月或几年。项目的完成是提供精神收入——自尊，承认和奖励——的主要源泉。
2. 你是否要更为高效和采用各种优点？Shell给予你迅速而高效地解决问题的能力。程序员倾向于认为他们必须用 C 或 COBOL 或其他语言编程。其实这并不是必需的。从 Shell 程序组合就几乎可以完成任何任务。不必再等数月以得到“完美的解答”，原型 Shell 应用可以被开发、精炼和实现。如果需要，它们可以被用作“实际”系统开发的需求说明。把 UNIX Shell 工具箱的全部作用用于即时问题的能力超过使用 Shell 的效率损失。

3. 你是否需要更多时间？每一个作业都有激动人心的部分和索然无味的部分。Shell把你从索然无味、典型的重复任务中解放出来。
4. 你是否需要更多的及时信息？Shell可以抽取极大量的信息来满足你的需要。当Shell可以为你扫描和检索重要信息时为什么还要请求整个原始报告呢？
5. 是否需要从各种已有系统集成信息？使用UNIX的通讯机制和Shell的威力，集成可以很容易地完成。
6. 是否为自己或用户创建确实需要的应用程序的时间十分紧迫？用UNIX Shell，可以迅速地建立应用程序的原型，这样可以保证在建立产品系统之前需求是正确的。即使在这一早期阶段，Shell版本也可能是十分令人满意的。
7. 你是否要更有创造性？工作中是否有足够的乐趣？用Shell程序设计进行修补对于新手和专家都是十分有趣的。Shell的巨大的工具库可以被容易地组合起来，处理许多重复性的活动。
8. 是否要避免重新培训来使用UNIX？是否宁愿使用熟悉的命令名，而不愿使用UNIX命令？Shell允许你仿真以前使用过的任何环境。它可以很方便地仿真DOS(我把它称为"小"UNIX)或IBM MVS TSO环境。用Shell工具的组合，可以创建仿真大多数系统的功能的命令。无需重新培训。

UNIX Shell如何完成所有这些工作？它一定是很复杂的！其实正好相反。处理、接口和数据管理都十分简单，这也是人们乐于使用Shell的原因。

1.3 Shell的简单性

Shell命令相互之间通过一个被称作管道(pipe)的简单而又连贯的接口对话。Shell使用UNIX的文件系统(类似组织图的等级表)，允许以目录形式组织文件。目录和文件的这一简单等级给出系统中所有信息的简单清晰的视图。

UNIX系统是Shell能力的另外一个关键。UNIX是可移植的，它可以在当今生产的几乎所有计算机上运行。你在Shell程序的培训、教育和开发方面的投资也可以移植到各个系统。因为UNIX支持多用户和多任务，所以你在Shell编程上的投资可以支持其他用户，而且可以在你工作的同时在后台完成成千上万的重复性任务。

Shell与听上去基本相似，是在用户之间起保护作用的友好环境。它允许用户做其想做的事情，而不影响其他用户。当一个用户登录到UNIX系统中时，操作系统就自动启动Shell的一个唯一的副本，在其之下用户可以执行任何已有的功能。这一被保护的有力环境给每一个用户以提高生产率的能力。

1.4 UNIX Shell

UNIX主要有两种：Berkeley(BSD 4.3)和AT&T System V。随着POSIX逐渐成为标准，最好就是两者同时出现以构成一个较强的环境。在这两个UNIX环境中，有三个Shell：Bourne，Korn和C(见表1.1)。所有这些都支持进程——前台的和后台的，管道，过滤程序，以及其他类似的UNIX标准功能。最早的Shell是S.R.Bourne大约在1975年

重写的．Bill Joy 和学生在加大 Berkeley 分校创建了另一个被称作 C Shell 的 Shell 程序，它对于 C 程序员十分有用．它在 Berkeley 的 UNIX(BSD 4.3)下运行．David Korn 在 AT&T 创建了 Korn Shell，它保留了 Bourne Shell 的功能，同时又联合了 C Shell 的许多有用功能．这些 Shell 中的哪一个可以主宰市场现在还看不清；但我预测将是 Korn Shell．我预计在将来的 UNIX 版本中可以同时看到这三个 Shell，因为支持用户的已有知识十分容易且费用很低，而不必限制用户．本书将描述所有三种 Shell．

表 1.1: UNIX Shell

Shell	开发者	系统名称	提示符
Bourne	S. R. Bourne	sh	$
Korn	David Korn	Ksh	$
C	Bill Joy	csh	%

1.4.1 Bourne Shell

Bourne Shell 是这三个 Shell 中最普遍的(见表 1.2)．几乎所有的 UNIX 实现都把 Bourne Shell 作为它们的标准配置的一部分．它比其他两个 Shell 都要小，因此对于大多数 Shell 进程都比较高效．但是它的交互性要比 C Shell 和 Korn Shell 差．

表 1.2 Shell 功能

功能	Bourne	Korn	C
可用性	最大	最小	Berkeley UNIX; 有些系统 V
变量	局部	局部	局部(set)
	全局(export)	全局(export)	全局(setenv)
控制命令	if–then–else–fi	if–then–else–fi	if–then–else–endif
	case–esac	case–esac	switch–case–endsw
		select	
	for–do–done	for–do–done	foreach–end
	xargs	xargs	repeat
	while–do–done	while–do–done	while–end
	until–do–done	until–do–done	
条件求值	test，expr	直接	直接
交互	——	history	history
别名	函数	函数	alias
信号	trapo	trap	
效率	快	中	中

Bourne Shell 允许用 trap 命令进行异常处理，这对于 Shell 是唯一的．输入／输出重定向十分广泛．例如，它允许标准输入和输出的重定向到完整的控制结构中，而不象 C Shell 那样．Bourne Shell 也可以利用 System V 的命名管道(named pipe)的优点．

Bourne Shell 支持局部和全局变量。全局变量必须被输出。Bourne Shell 提供 if-then-else，case，for，while 和 until 控制结构。但它依赖 UNIX 的实用工具 test 和 expr 来求条件表达式的值，而不象 C Shell 和 Korn Shell 那样直接地求表达式的值。

1.4.2 C Shell

加大 Berkeley 分校开发的 C Shell 提供一些超过 Bourne Shell 的优点：history 以及条件求值和内部命令。C Shell 的 history 功能交互地跟踪你输入的命令，并允许你返回以再次执行它们而不必重新输入命令。或者根据需要，可以重新调用它们，作些修改，然后执行新命令。

C Shell 提供别名(aliasing)，允许用户创建命令名的别名。同时 C Shell 也提供对后台和前台任务的控制。在 Bourne Shell 中，如果在后台或前台启动了一个命令，则只能等到它结束。在 C Shell 中你则可以根据需要把命令从前台执行移到后台执行。

C Shell 提供两种变量，普通(局部)和环境(全局)，分别用 set 和 seteny 来建立。

C Shell 的语法与 C 语言程序设计十分相似，并且提供所有的 C 条件运算符(==，>，等)，这些 C 程序员可能认为十分有用。它提供 if-then-else，switch，foreach，repeat 和 while 控制结构，并且直接在这些控制结构中求表达式的值。

1.4.3 Korn Shell

Korn Shell 保留了 Bourne Shell 的所有功能，并且组合了 C Shell 的许多关键功能。对于大多数进程，Korn Shell 要比 C Shell 快，但比 Bourne Shell 慢。

Korn Shell 提供改进了的 history 管理，可以对以前的命令进行直接存取。类似 Bourne Shell，Korn Shell 也提供使用 Shell 函数的别名功能。为了效率 Korn Shell 直接求条件表达式的值，而且给菜单驱动的 Shell 增加了 select 控制结构(类似 case)。

1.4.4 选择 Shell

对于初学者最好选择 Korn Shell。因为 Korn / Bourne Shell 比 C Shell 简单。而熟悉 Bourne 或 CShell 的有经验用户也会从其已有的经验获益。而 Bourne Shell 用户可以从 Korn Shell 的交互功能获益。在全书中，对于各个 Shell 的例子将分别予以说明，另外 C Shell 的例子"csh:"开头，以便于用户查找。

1.5 何时使用 Shell

无论何时你输入一个 UNIX 命令，你就正在使用 Shell。为了提高生产率，每当你面临下列情况时就应使用 Shell:

1. 对许多文件做些事情。
2. 重复地做同一任务。
3. 按日程表自动地做任务。

人们根据需要执行重复性的任务:

- 一组文档中的日期或名字必须被修改，
- 每月必须做报表，
- 状态必须每天输入，每月报告。

这些任务中的每一个以及其他类似的都可以用一个 Shell 程序来自动进行。

你也可以在终端前交互地使用 Shell，自动处理一次性的任务。许多情况可以从 Shell 的强大功能的使用中获益，但它们并不需要建立一个单独的 Shell 程序。这些问题可以通过 Shell 的交互使用来解决。我们将在第四章"Shell 控制结构"中更深入地来看 Shell 的交互用法。

你不应使用 Shell，当任务:

- 太复杂，
- 要求高效率，
- 要求不同的硬件环境，或者
- 要求不同的软件工具。

用 Shell 来自动处理要求文本操作的任何事情：选择数据，数相加，打印统计结果，等等。在一大堆报告中找出正确的信息对于 Shell 十分简单，而对人来说就不太可能了。处理数据并以可打印的形式放置它们也是十分枯燥无味和不可靠的。正如你很快能看到，Shell 可以迅速而可靠地做所有这些工作。

1.6 高生产率和 Shell

研究表明(Thadhani 1984)一个程序员平均每周在终端前化 20 到 25 小时。而 95%的时间化在编辑和数据处理上。随着家庭、办公室和商务变得日益自动化，20–25 小时可能会变成 30–35。为了进一步提高生产率:

1. 响应时间必须为最小(在一秒种以下)，而且
2. 人们必须可以自动处理他们的耗时活动。

Shell 及其工具被设计和优化为自动处理这些活动中的许多。它要求一些对 Shell 及其用法的深入了解以导出这些好处，但它只是利用一点儿独特性而变得更为高效。

因为 Shell 可以自动处理大多数重复性的任务，它占人们耗时活动的 50–80%，所以毫无疑问 Shell 可以成倍地提高生产率。UNIX 文件和文件系统设计的简化使其成为可能。

第二章 Shell 基础

在当今科技迅猛发展的环境中，UNIX 肯定是你所考虑的．无法将 Shell 和 UNIX 分开讨论(虽然某些单任务版本在 MS-DOS 下运行)．如果没有 UNIX 结构的简单性，Shell 也肯定无法存在．Shell 本身从 UNIX 获得了许多威力．

2.1 UNIX 是什么?

UNIX 是一个操作系统？一种哲学？在从个人计算机到 Cray 超级计算机的每一个硬件平台上运行的为了提高个人的生产率的热门环境？回答是：所有这些都是．

UNIX 是一个分时操作系统．UNIX 与大多数传统的操作系统方法的最大区别是它简化了文件的读写——输入／输出(I／O)．所有文件和设备对于操作它们的命令都是相同的．文件包含字符流形式的数据——没有记录，没有可变记录大小，因此也没有与之相关的问题．这一独特的设计使得每一个程序都可以从任何其他程序接受输入．每一个程序都可以执行单独一个独有的功能，而且可以通过操作系统与其他程序、设备或文件相连结．这一简单设计给 UNIX 和 Shell 赋以了它们的威力．

随着 UNIX 的发展，许多软件开发人员开发了许多其他面向用户的工具．最早作为 UNIX 的扩充而封装的被称作程序员工作台(Programmer's Work Bench，PWB)的工具，现在已被包括到 UNIX 的标准软件包中．

2.2 UNIX 文件

UNIX 文件是非常独特的，因为它们基本上没有格式．每一个文件都只是一个字符序列(见图 2.1)．行或记录被用换行符(\n 或 nl)分界．文件的结束被用文件结束符(\Q 或 EOF)或磁带结束符(EOT)标识．由于每一个文件都可以被逐字符地读取和输出，每一个 Shell 工具都被设计为处理这一简单的文件结构．这一设计选择使得 UNIX 的设计者可以创建简单的、模块化的程序来执行单一的任务．虽然每一个函数被看作单独一个实体时都是十分平常的，但是当与其他单一化的函数组合起来时就变得十分重要的，几乎可以完成任何类型的任务．

```
The UNIX Shell is the key to improving your productivity\n
and quality in a typical UNIX environment.\n
The Shell can automate repetitive tasks,\n
find where you left things,\n
do things while you are at lunch,\n
and a host of other time-saving activities.\n\Q
```

图 2.1：一个 UNIX 文件

UNIX 文件存在于等级化的文件系统或倒置树(类似于结构图)中，如图 2.2 所示。要实现这一结构，UNIX 使用被称作目录的特殊文件。你可以把目录看作文件柜、文件抽屉或其他装文件的器具。每一个目录是等级图中的一个分叉，分支可能从它那儿生长。这一机制对于组织文件和信息十分有用。在图 2.2 中，用户标识 lja 位于文件系统 / enduser 之下。在它之下还有 Shell 命令的目录(bin)，文档(doc)和源代码(src)。这些名字都十分简单，因为大多数人都是很糟糕的打字员。象 source code 或 documentation 之类的冗长名字很难无错误地键入，而且也十分耗费时间。src 之下的目录是 C 语言(c.d)和 COBOL(cobol.d)的。该约定用后缀".d"很清楚地表明它们是目录。有些人用大写字母(如 C 或 COBOL)表示目录。这些目录的每一个下面都是大量的文件(如图中的矩阵所示)。通过使用目录和它们的名字，一般就可以很快地找到它们。要在 UNIX 文件系统中找你留下的东西通常是一个很大的工作，尤其当在一个目录有 30 到 100 个文件时。

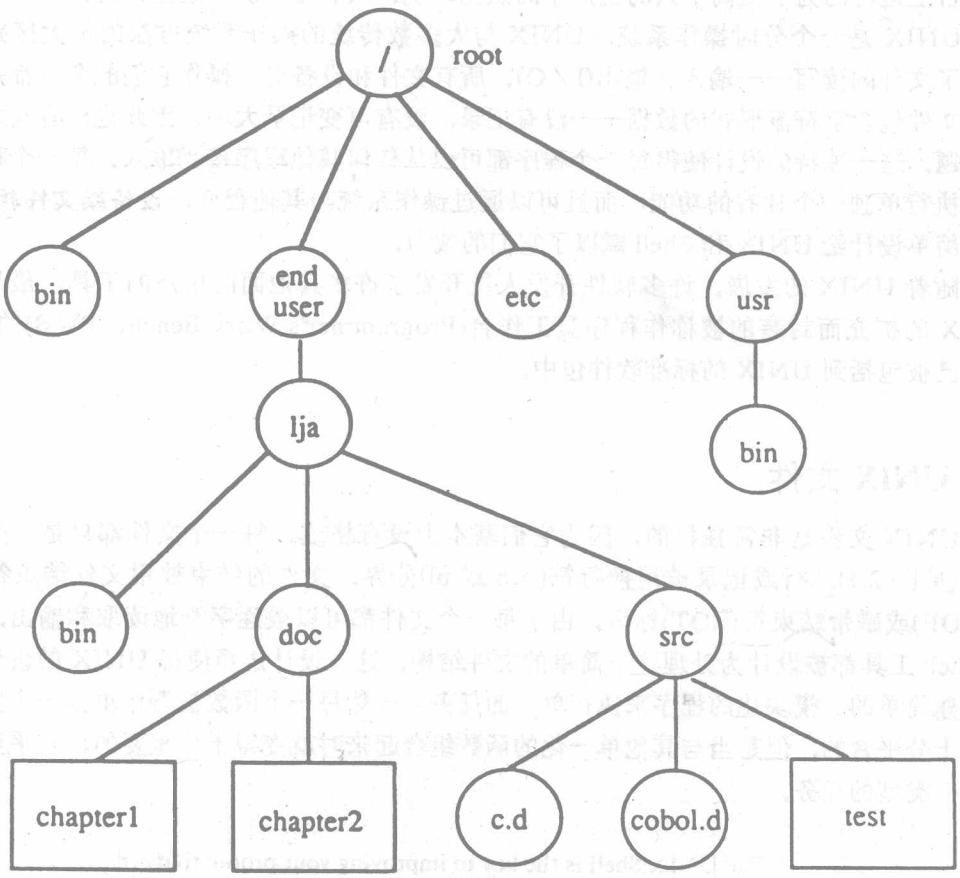


图 2.2 一个 UNIX 文件系统

目录和数据文件并不是 UNIX 提供的唯一文件类型。另外一些"特殊"的文件其实根本不是真正的文件，其实是象终端处理器、磁盘驱动器、磁带驱动器等的设备。这些将在高级的材料中详细讨论。这些文件中的任何一个都可以用一个 Shell 命令处理以过滤或增