PEARSON

UNIX编程环境

The UNIX Programming Environment

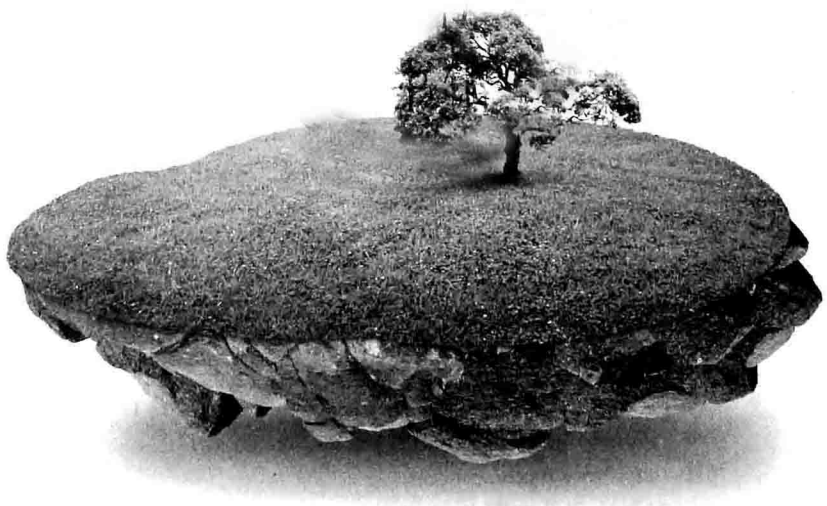
- 经久不衰的UNIX经典教程
- 两位UNIX大师合力之作，浸透了UNIX的设计思想
- 启发读者体会编程方法、思想以及环境的奥秘

[美] Brian W. Kernighan　Rob Pike　著
陈向群　等　译

人民邮电出版社
POSTS & TELECOM PRESS

UNIX编程环境

[美] Brian W. Kernighan　Rob Pike 著
陈向群 等 译



人民邮电出版社
北京

图书在版编目（CIP）数据

UNIX编程环境 /（美）克尼汉（Kernighan,B.W.），（美）派克（Pike,R.）著；陈向群等译. -- 北京：人民邮电出版社，2014.4
ISBN 978-7-115-33835-8

Ⅰ. ①U… Ⅱ. ①克… ②派… ③陈… Ⅲ. ①UNIX操作系统－程序设计 Ⅳ. ①TP316.81

中国版本图书馆CIP数据核字(2013)第283960号

内 容 提 要

本书是关于在 UNIX 环境下进行程序设计的一本经典教科书。书中引用了大量编程实例，由浅入深地讲解了如何使用 UNIX 及其各种工具，以及如何用 C 语言在 UNIX 环境下写出高质量的程序。

本书共 9 章 3 个附录，第 1 章为系统基础入门，第 2 章讨论 UNIX 文件系统，第 3 章讲述怎样按要求使用 shell，第 4 章介绍过滤程序，第 5 章讨论如何使用 shell 编写程序，第 6 章讲述运用标准 I/O 库编写程序，第 7 章涉及系统调用，第 8 章讨论有关程序开发的工具，第 9 章讨论文档准备工具，附录 A 概括了标准编辑器 ed，附录 B 列出了一个编程实例——计算器的语言参考手册，附录 C 是编程实例——计算器程序的最后源代码版本。

本书适合作为大学院校相关专业的教科书，对于想深入掌握 UNIX 和 C 语言的程序设计人员是一本很好的参考书。本书也适合想学习和掌握 Linux 的人员阅读。

◆ 著　[美] Brian W. Kernighan　Rob Pike
译　陈向群 等
责任编辑　杨海玲
责任印制　程彦红　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：800×1000　1/16
印张：20.75
字数：446 千字　2014 年 4 月第 1 版
印数：1－3 000 册　2014 年 4 月河北第 1 次印刷

著作权合同登记号　图字：01-2012-8863 号

定价：59.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

序

“UNIX 安装的数量已经增加了 10 倍，预期还将更多。”

——UNIX 程序员手册，1972 年 6 月第 2 版

UNIX[①]操作系统是 1969 年首次在贝尔实验室的一台丢弃的 DEC PDP-7 计算机上启用的。当时 Ken Thompson 从 Rudd Canaday、Doug McIlroy、Joe Ossanna 和 Dennis Ritchie 那里获得理念和支持，编写了小型通用分时系统，其适用性能良好，足以吸引热心的用户，并最终为一台较大的计算机——PDP-11/20 的购买提供了充分的可靠性。系统早期的用户之一是 Ritchie，他在 1970 年曾帮助将该系统转到 PDP-11 计算机上。Ritchie 也曾设计和编写了 C 程序语言用的编译程序。1973 年，Ritchie 和 Thompson 用 C 语言重写了 UNIX 核心，打破了系统软件要用汇编语言编写的传统。经过此次重写，该系统最终成为今天必不可少的操作系统。

大约在 1974 年，大学获得了“用于教育目的”的 UNIX 使用许可，而且，几年后 UNIX 有效地开拓了商业用途。在此期间，UNIX 系统在贝尔实验室取得了成功，从而找到了进入实验室、软件开发项目、字处理中心和电话公司运行支持系统的途径。从那时起，UNIX 已遍布整个世界，从微型计算机到最大的主机，已安装了数以万计的 UNIX 系统。

是什么促使 UNIX 系统如此成功呢？可以阐明几个原因。首先，由于是用 C 语言编写，因此它是可移植的，UNIX 系统的运行范围可以从微处理系统到最大的主机。这是一个强大的商业优势。其次，源代码非常有效并且是用高级语言编写的，从而使系统容易适应特殊的需求。最后，也是最重要的一点，它是一个良好的操作系统，特别是对程序员而言，UNIX 程序环境通常是丰富而有成果的。

尽管 UNIX 系统引入了很多创新的程序和技术，但是一个单独的程序或想法并不能使工作完美无缺。相反，是程序设计的处理方式才使系统有成效，这也是应用计算机的一个基本原则。虽然这个基本原则不能用一句话概括，但其核心就是，系统的能力更多地来自程序之间的关系，而不是程序本身。许多 UNIX 程序虽然单独做相当简单的工作，但是，在与其他

---

① UNIX 是贝尔实验室的商标。“UNIX”不是首字母缩拼词，它是一个同 MULTICS 稍微有联系的双关语，在 UNIX 之前，Thompson 和 Ritchie 工作于 MULTICS 操作系统之上。

程序结合起来使用时，它们就成为全面而有益的工具。

本书的目的是传播 UNIX 程序设计的基本原则，由于这个基本原则是以程序间的关系为基础的，因此我们必须用大量的篇幅专门对单个的工具进行讨论，但同时贯穿了程序组合的主题和运用程序进行创建程序的内容。要想很好地使用 UNIX 系统及其组成部件，不仅必须了解如何使用程序，还要了解怎样使其与环境相匹配。

随着 UNIX 系统的广泛应用，能够熟练掌握 UNIX 系统的用户越来越难找到。好几次，我们发现富有经验的用户，包括我们自己在内，只能笨拙地找到问题的解法，或者只会写程序去做现有的工具易于处理的工作。当然，如果没有一定的经验和理解，就不易了解精巧的解法。我们希望无论是新手还是老用户，通过阅读本书，都能够开发理解力，从而使系统的使用更加有效和有趣。希望读者更好地使用 UNIX。

我们把目光对准程序员们，希望通过本书使其工作更有效，从而也可以使程序组的工作更有成果。尽管我们的主要对象是程序员，但前 4 章或前 5 章并不需要具有编程经验就可以理解，因此，对于其他用户来说，本书也是很有帮助的。

无论如何，我们已经尽可能尝试使用真实的、而不是人为的例子来表述我们的观点。尽管有一些程序在本书中是作为例子开始的，但它们就此而成为常用程序的一部分。所有实例都经过了机器可读格式文本的直接测试。

本书的编排如下：第 1 章为系统使用的基础入门。其中包括登录、邮件、文件系统、常用命令和命令解释的基本原理。有经验的用户可以略过此章。

第 2 章讨论 UNIX 文件系统。文件系统是系统操作和使用的核心，因此必须充分理解才能更好地使用。这一章讲述文件和目录，容许权限和文件模式，以及 i 接点。其中包括文件系统层次的浏览和设备文件的解释。

命令解释程序，即 shell，是一个基本工具，不仅用于执行程序，也用于写程序。第 3 章讲述怎样按要求使用 shell：创建新命令，使用命令变量、shell 变量、基本控制流和输入输出再定向。

第 4 章介绍过滤程序：当数据流通过时，对数据执行简单转换的程序。第一部分涉及 grep 模式搜索命令和其对应程序；接下来讨论诸如 sort 的一些更多的常用过滤程序；其余部分重点讨论两个通用数据转换程序，称为 sed 和 awk。sed 是流编辑程序，是当数据流通过时，做出编辑更改的一个程序。awk 是用于简单信息检索和报告生成任务的一种程序设计语言。使用这些程序，有时再加上与 shell 的配合，常常就能完全避免流于常规的程序设计。

第 5 章讨论如何使用 shell 进行程序编写，并能够经得起其他人的使用。主要内容包括更高级的控制流以及变量、陷阱和中断处理。本章的实例充分利用了 sed、awk 和 shell。

人们最终会达到运用 shell 和其他现有的程序进行工作所能达到的极限。第 6 章讲述有关

运用标准 I/O 库写新程序的内容。这些程序用 C 语言编写，我们假设读者了解或至少最近学过 C 语言。我们会展示用于设计和组织新程序的实用策略，以及怎样用易于管理的步骤创建程序和如何利用现有的工具。

第 7 章涉及 UNIX 系统调用，这是其他所有软件层的基础，主要内容包括输入/输出、文件创建、出错处理、目录、i 接点、进程和信号。

第 8 章讨论有关程序开发的工具：yacc，一种语法分析程序生成器；make，可以控制编译大程序的过程；lex，它可以生成词法分析程序。程序评注是以一个大程序的开发为基础的，该大程序是一种类似 C 语言的可编程计算器。

第 9 章讨论文档处理工具，可用一种用户级的叙述以及手册的一张页面来说明第 8 章的计算器。本章可以独立于其他章节进行阅读。

附录 A 概括了标准编辑器 ed。尽管许多读者在日常工作中喜欢运用其他编辑器，但 ed 是非常通用、有益而且高效的。它的正则表达式是其他的程序，如 grep 和 sed 的核心，而且只为此就值得一读。

附录 B 包括用于第 8 章计算器语言的参考手册。

附录 C 是计算器程序最后版本的目录，在同一处列示了所有的编码以便于阅读。

另外，还有一些实用的内容。首先，UNIX 系统的使用已经很普遍，有多种广泛应用的版本。例如，第 7 版出自 UNIX 系统的发源地——贝尔实验室计算机科学研究中心。系统 III 和系统 V 是贝尔实验室支持的正式版本。Berkeley 是加利福尼亚大学提供的系统，源自第 7 版，通常称为 UCB 4.xBSD。另外，特别是对于小型计算机，有大量变种 UNIX 系统，它们也是从第 7 版本派生的。

我们尽可能地把完全一致的内容放在一起，来对付这些版本差异。尽管我们将要讲授的课程是独立于任何特定版本的，但对于我们所选定的细节内容而言，会和第 7 版中有关内容一致，因为它是以广泛应用的绝大多数 UNIX 系统为基础的。我们也已经在贝尔实验室的系统 V 上和 Berkeley 4.1BSD 中运用了这些例子，只需做一些轻微的改动，而且改动只限于少数例子。不管机器运用何种版本，读者会发现其间的差异是很小的。

其次，虽然本书中有很多参考资料，但它不是一本参考手册。我们知道讲授处理和使用风格要比仅仅讲授细节重要得多。《UNIX 程序员手册》是信息的标准源泉。读者需要使用手册去解决那些没有包含在本书中的问题，或者检测读者的系统与本书作者的系统之间的不同之处。

再次，我们相信，实践出真知。本书应在终端旁阅读。这样便于对所讲的内容进行实验、检查或者反驳，探索其极限和变化。边阅读，边实践，回顾小结，然后再更多地阅读。

我们相信，尽管 UNIX 系统不是完美无缺的，但它是一个相当优秀的计算机环境；我们希望，对本书的阅读能帮助读者也得出这一结论。

衷心感谢提出建设性意见和建议的人们，以及他们对改进代码所做的协助。特别要感谢 Jon Bentley、John Linderman、Doug McIlroy 和 Peter Weinberger，他们以极大的热情阅读了大量的文稿。我们非常感激 Al Aho、Ed Bradford、Bob Flandrena、Dave Hanson、Ron Hardin、Marion Harris、Gerard Holzmann、Steve Johnson、Nico Lomuto、Bob Martin、Larry Rosler、Chris Van Wyk 和 Jim Weythman 对初稿所提的意见，同时我们也感谢 Mike Bianchi、Elizabeth Bimmler、Joe Carfagno、Don Carter、Tom De Marco、Tom Duff、David Gay、Steve Mahaney、Ron Pinter、Dennis Ritchie、Ed Sitar、Ken Thompson、Mike Tilson、Paul Tukey 和 Larry Wehr 的宝贵建议。

Brian Kernighan
Rob Pike

目录

第 1 章　初学 UNIX …… 1
1.1　起步 …… 2
1.1.1　有关终端和输入的一些预备知识 …… 2
1.1.2　与 UNIX 会话 …… 2
1.1.3　登录 …… 3
1.1.4　键入命令 …… 4
1.1.5　异常的终端行为 …… 5
1.1.6　键入错误 …… 5
1.1.7　继续键入 …… 7
1.1.8　中止程序 …… 7
1.1.9　注销 …… 7
1.1.10　邮件 …… 7
1.1.11　用户间通信 …… 8
1.1.12　新闻 …… 9
1.1.13　手册 …… 9
1.1.14　计算机辅助教学 …… 10
1.1.15　游戏 …… 10
1.2　文件和常用命令 …… 10
1.2.1　创建文件 …… 10
1.2.2　列出文件 …… 11
1.2.3　显示文件 …… 13
1.2.4　移动、复制和删除文件 …… 15
1.2.5　文件名 …… 16
1.2.6　有用的命令 …… 16
1.2.7　文件系统命令小结 …… 19
1.3　目录 …… 20
1.4　shell …… 24
1.4.1　文件名简写 …… 24
1.4.2　I/O 重定向 …… 26
1.4.3　管道 …… 29
1.4.4　进程 …… 30
1.4.5　剪裁环境 …… 33
1.5　UNIX 系统的其余部分 …… 35
相关历史和文献 …… 36

第 2 章　文件系统 …… 37
2.1　文件系统的基础 …… 37
2.2　文件结构 …… 41
2.3　目录和文件名 …… 43
2.4　权限 …… 47
2.5　i 节点 …… 52
2.6　目录层次 …… 57
2.7　设备 …… 59
相关历史和文献 …… 63

第 3 章　shell 的使用 …… 64
3.1　命令行结构 …… 64
3.2　元字符 …… 67
3.3　创建新命令 …… 72
3.4　命令参数 …… 74
3.5　程序输出作为参数 …… 77
3.6　shell 变量 …… 79
3.7　进一步讨论 I/O 重定向 …… 83
3.8　shell 程序里的循环 …… 85
3.9　bundle 合并 …… 88
3.10　为什么说 shell 是可编程的 …… 89

相关历史和文献 …… 90

第 4 章　过滤程序 …… 91
4.1　grep 家族 …… 92
4.2　其他过滤程序 …… 95
4.3　流编辑程序 sed …… 97
4.4　模式扫描与处理语言 awk …… 103
4.4.1　字段 …… 104
4.4.2　打印 …… 105
4.4.3　模式 …… 106
4.4.4　BEGIN 与 END 模式 …… 107
4.4.5　算术运算与变量 …… 107
4.4.6　控制流 …… 109
4.4.7　数组 …… 111
4.4.8　关联数组 …… 112
4.4.9　字符串 …… 113
4.4.10　与 shell 的交互作用 …… 115
4.4.11　基于 awk 的日历服务 …… 116
4.4.12　附注 …… 118
4.5　好的文件与好的过滤程序 …… 119
相关历史和文献 …… 120

第 5 章　shell 程序设计 …… 121
5.1　定制 cal 命令 …… 121
5.2　which …… 126
5.3　while 和 until 循环：观察情况 …… 131
5.4　trap：捕获中断 …… 136
5.5　overwrite：改写文件 …… 139
5.6　zap：使用名字终止进程 …… 143
5.7　pick 命令：空格和参数 …… 145
5.8　news 命令：社团服务信息 …… 148
5.9　get 和 put：追踪文件变动 …… 150
5.10　后记 …… 155
相关历史和文献 …… 156

第 6 章　使用标准 I/O 编程 …… 157
6.1　vis：标准 I/O …… 158
6.2　vis 第 2 版：程序参数 …… 160
6.3　vis 第 3 版：访问文件 …… 162
6.4　p：一次显示一屏 …… 166
6.5　示例：pick …… 171
6.6　错误与调试 …… 172
6.7　示例：zap …… 174
6.8　idiff：交互式文件比较程序 …… 177
6.9　获取环境变量 …… 182
相关历史和文献 …… 183

第 7 章　UNIX 系统调用 …… 184
7.1　低级 I/O …… 184
7.1.1　文件描述符 …… 184
7.1.2　文件 I/O：read 和 write …… 185
7.1.3　创建文件：open、creat、close、unlink …… 187
7.1.4　错误处理：errno …… 189
7.1.5　随机访问：lseek …… 190
7.2　文件系统：目录 …… 191
7.3　文件系统：i 节点 …… 196
7.4　进程 …… 201
7.4.1　创建低级进程：execlp 和 execvp …… 201
7.4.2　控制进程：fork 和 wait …… 203
7.5　信号和中断 …… 205
相关历史和文献 …… 210

第 8 章　程序开发 …… 212
8.1　第一阶段：四则运算器 …… 213
8.1.1　文法 …… 213
8.1.2　yacc 概述 …… 214

8.1.3 第一阶段的程序……215
8.1.4 修改程序——增加一元减……219
8.1.5 关于 make……220
8.2 第二阶段：变量和错误恢复……220
8.3 第三阶段：任意变量名和内部函数……224
8.3.1 再谈 make……232
8.3.2 关于 lex……233
8.4 第四阶段：编译到机器……235
8.5 第五阶段：控制流和关系运算符……242
8.6 第六阶段：函数、过程和 I/O……248
8.7 性能评价……257
8.8 小结……259
相关历史和文献……260

第 9 章 文档处理……261
9.1 宏程序包 ms……262
9.1.1 显示……264
9.1.2 改变字体……266
9.1.3 其他命令……267
9.1.4 宏程序包 mm……268
9.2 troff……269
9.2.1 字符名……269
9.2.2 改变字体和尺寸……271
9.2.3 基本 troff 命令……272
9.2.4 定义宏……273
9.3 tbl 与 eqn 预处理器……273
9.3.1 表格……274
9.3.2 数学表达式……275
9.3.3 输出……277
9.4 排印手册……279
9.5 其他文档处理工具……283
相关历史和文献……284
结束语……286
附录 A 编辑器概述……288
附录 B hoc 手册……299
附录 C hoc 清单……304

第 1 章

初学 UNIX

什么是 UNIX？狭义地看，它是一个分时操作系统内核，即一个控制计算机的资源并将其分配给用户的程序。它让用户运行其程序，并控制与机器连接的外围设备（硬盘、终端、打印机等），提供一个文件系统用以管理诸如程序、数据及文档等长期存储的信息。

广义地看，UNIX 通常不仅包含内核，还包括一些基本程序，如编译器、编辑器、命令语言、用以复制和显示文件的程序等。

从更广的角度来看，UNIX 可以包括由用户开发的、运行于用户的 UNIX 操作系统上的程序，如文档处理工具、统计分析程序以及图像软件包等。

这些有关 UNIX 的解释究竟正确与否，取决于读者所面对的系统的应用级别。本书其他部分提到 UNIX 时，会在上下文指示其内在含义。

UNIX 系统有时看起来比实际上更复杂——对于新用户而言，很难充分利用可用的资源。所幸它并不难入门——只要了解很少的几个程序，就可以开始工作了。本章会帮助读者尽快地学会使用 UNIX 系统。本章是概述，不是手册；后续的章节将详细介绍各种内容。本章涉及以下主要内容。

- 基本操作——登录和退出、简单的命令、纠正输入错误、邮件及终端间通信。
- 日常使用——文件及文件系统、文件显示、目录以及常用命令。
- 命令解释器或 shell——文件名缩写、输入输出重定向、管道、删字符及消行符设置、命令查询路径定义。

如果你用过 UNIX，那么对本章的多数内容应该是熟悉的，可以直接跳至第 2 章。

即使阅读本章，也需要一份《UNIX 程序员手册》。对作者而言，告诉你翻阅手册中的某些内容，比在书中重复这些内容更为方便。本书不是为了替代手册，而是为了教会你充分利

用手册中的命令。另外，本书中所叙述的内容可能同你的系统上的内容有所差别。在手册中的开始处有索引，可以协助找出解决某一问题的程序，应学会如何使用它。

最后，有一句忠告：不要害怕实践。如果你是初学者，请放心，不会出现伤害自身或其他用户的事。实践出真知。本章篇幅较长，最好的方法是一次读几页，并在学习过程中不断实践。

1.1　起步

1.1.1　有关终端和输入的一些预备知识

为了避免繁冗解释有关计算机使用的所有事项，作者假定你已经熟悉计算机终端，并知道使用方法。如果下面的叙述使你迷惑不解，请询问身边的专家们。

UNIX 系统是全双工的：你在键盘上敲入的字符送至系统，然后系统回送给终端并在屏幕上显示出来。通常 echo 进程把字符直接复制到屏幕上，这样就可以看到输入的是什么。但是有的时侯，比如在键入密码时，echo 关闭，字符就不会在屏幕上显示出来。

多数键盘字符是普通字符，没有什么特殊的含义。但有些字符是通知计算机怎样解释所键入的内容的。到目前为止，这些键中最重要的键是回车键（RETURN）。回车键说明一行输入的结束，系统做出反应把屏幕上的光标移到下一行的起始处。必须按下回车键后，系统才会对键入的字符做出解释。

回车键是一种控制字符，即不可见的字符，它控制着终端上某些特殊的输入输出。在任一台终端上，回车键都有自己的按键，但是其他大多数控制字符并非如此。相反，它们必须通过按住 Ctrl 键（有时称为 CONTROL 键、CNTL 键或 CTL 键）的同时按下另一个键（通常是一个字母）来输入。例如，回车键可以通过按下回车键输入，或者按住 CONTROL 键同时按下 m 键。所以回车键又可称为 CONTROL+m 键，也可以写成 Ctrl + m。其他的控制字符有：Ctrl + d，表示程序输入到此结束；Ctrl + g，终端上的振铃鸣响；Ctrl + h，通常称为（退格键）Backspace，它可以用于纠正错误的输入；还有 Ctrl + i，通常称为 Tab，它使光标往前跳一个 Tab 间隔，这一点非常类似于普通的打字机。在 UNIX 系统中，Tab 间隔为 8 个空格字符。在多数终端上 Backspace 和 Tab 都有各自的键。

另外两个键也具有特殊的含义：一个是 Delete，有时又称为 Rubout 或某些其他缩写；另一个是 Break，有时又称为 Interrupt。在多数 UNIX 系统中，按 Delete 键可以立即中止程序，而不等待程序完成。在某些系统中，Ctrl + c 提供这项功能。而在有些系统中，根据终端的连接方式，Break 与 Delete 或 Ctrl + c 的功能一样。

1.1.2　与 UNIX 会话

让我们从与 UNIX 系统的会话开始。在本书的例子中，你键入的内容用等宽斜体字符表

示，计算机回应的内容用等宽正体字符表示，说明性的文字用楷体表示。

建立连接：必要时拨号或打开电源。
系统应该显示：

```
login: you                          键入用户名后按回车
Password:                           口令在键入时不会显示
You have mail.                      登录后可看邮件
$                                   系统等待用户命令
$                                   按几次回车
$ date                              日期和时间
Sun Sep 25 23:02:57 EDT 1983
$ who                               谁在使用机器
jlb      tty0    Sep 25 13:59
you      tty2    Sep 25 23:01
mary     tty4    Sep 25 19:03
doug     tty5    Sep 25 19:22
egb      tty7    Sep 25 17:17
bob      tty8    Sep 25 20:48
$ mail                              读邮件
From doug Sun Sep 25 20:53 EDT 1983
give me a call sometime monday

?                                   按回车键跳到下一条消息
From mary Sun Sep 25 19:07 EDT 1983  下一条消息
Lunch at noon tomorrow?

? d                                 删除该消息
$                                   没有邮件了
$ mail mary                         向 mary 发邮件
lunch at 12 is fine
ctl-d                               邮件结束
$                                   挂断电话或关闭终端，全部结束
```

上述内容就是一段会话的全部，当然，有时人们也做一些其他的工作。本节的下面部分将讨论这段会话，顺便分析一些其他有用的程序段。

1.1.3 登录

登录必须有登录名和口令，这些可以从系统管理员处得到。UNIX 系统可以适用于多种类型的终端，但它更倾向于有小写字体的设备，因为 UNIX 区分大小写！如果你的终端只有大写字符（类似某些可视终端和便携式终端），那么使用起来会很困难，应该另外换一台终端。

要确认设备的开关设置为大小写、全双工，以及身边专家们建议的其他设置，诸如传输速率，即波特率。要为终端建立连接，这也许要拨通电话，或者只是拨一下开关。不论哪种情形，系统都会出现：

```
login:
```

如果出现乱码，则可能是速率不对，请检查速率和其他设置。如果仍不成功，按几次 Break 键或 Interrupt 键。如果仍不出现登录消息，那就只好请人帮忙了。

出现 login:登录提示后，用小写字母键入登录名，然后按回车键。如果要求密码，系统会有提示，而且屏幕不会显示所输入的密码。

登录一旦成功，系统便会显示一个提示符，通常是一个字符，表示系统已经准备好接收用户的命令。提示符多半是美元符号（$）或百分比符号（%），但是你可以将其改为任意的符号，这一点以后再讨论。提示符实际是由一个名为命令解释器或称为 shell 的程序显示出来的，它是系统对用户的主界面。

在提示符之前可能会有日期消息，或者有关于电子邮件的通知。有时系统会询问你正在使用的终端类型，这有助于系统利用终端所具有的特性。

1.1.4　键入命令

一旦有了提示符，就可以键入命令了，命令就是请系统完成某项工作的要求。我们使用“程序”这一词作为命令的代名词。当看到提示符（假设是$）时，键入 date 并按下回车键。系统应该回应日期和时间，然后显示下一个提示符，所以整个处理过程在终端上看起来如下所示：

```
$ date
Mon Sep 26 12:20:57 EDT 1983
$
```

不要忘了回车键，也不要键入$。如果系统没有反应，可以按回车键，应该会有响应。以后不再按回车键，但是在每一行的结尾都需要它。

下一个要尝试的命令是 who，它表明当前有哪些人在登录上机：

```
$ who
rlm      tty0     Sep 26 11:17
pjw      tty4     Sep 26 11:30
gerard   tty7     Sep 26 10:27
mark     tty9     Sep 26 07:59
you      ttya     Sep 26 12:20
$
```

首列是用户名。第二列是连接的终端的名称（tty 即 teletype，终端的一个学名）。其余信息是登录的日期和时间。读者还可试验：

```
$ who am i
you      ttya     Sep 26 12:20
$
```

如果键入出错，输入了一条不存在的命令，会被告知没有发现该命令：

```
$ whom                          拼错命令名……
whom: not found                 ……所以系统不知如何执行
$
```

当然，如果你不恰当地键入了一条实际存在的命令，它会运行，也许会出现奇怪的结果。

1.1.5 异常的终端行为

有时终端会出现一些奇怪的行为，例如，一个字符也许会显示两次，或者回车键可能不把光标置到下一行的首列。通常可以把终端关闭再开启，或者退出登录然后再次登录以消除这些现象。也可以阅读手册第 1 节中有关 stty（set terminal options）命令的叙述。如果终端没有 Tab 键，要想巧妙地处理该键，可键入命令

```
$ stty -tabs
```

接着系统会把 tabs 转换成正确数量的空格。如果终端的 Tab 间隔可以设置，命令 tabs 会正确地为用户设置跳格位置。这时，需要键入

```
$ tabs 终端类型
```

这样即可使系统正确工作——请参阅手册中有关 tabs 命令的说明。

1.1.6 键入错误

如果出现键入错误，并且在按回车键前发现了它，可有两个方法纠正：一次删去一个字符或者删行后重新键入。

如果键入消行符，其默认值是@，会取消整行，就好像没有键入该行一样，从而可以开始新的一行：

```
$ ddtae@                        命令输入错，整行删除；开始新的一行
date
Mon Sep 26 12:23:39 EDT 1983
$
```

字符#删除所键入的最末一个字符，每个#字符删去一个字符，一直到行的起始处（但不会超越它）。如果键入出错太多，可以进行如下纠正：

```
$ dd#atte##e                    纠正错误
Mon Sep 26 12:24:02 EDT 1983
$
```

专用删字符和消行符是与系统密切相关的。在很多系统中（包括作者使用的系统），删字符都已改为 Backspace 键，这在可视终端上很成功。你可以键入如下内容迅速检查所使用的系统是哪一种情况。

```
$ datee←                          试一下 Backspace 键
datee←: not found                 不是 Backspace 键
$ datee#                          试一下#
Mon Sep 26 12:26:08 EDT 1983 正是#
$
```

（我们把 Backspace 键印刷为←，这样就可以看到它。）消行符的另一种常见形式是 Ctrl + u。

在本节的其余部分，将使用字符#作为删字符，因为它较为清晰可见，如果系统不一致，请做好思想准备。在 1.4.5 节将介绍如何设置你所喜欢的删字符和消行符。

假如必须把删字符或者消行符作为文本的一部分输入，该怎么办呢？在#或@之前加入反斜杠（\），它们就失去了特有的含义。所以要输入#或@，只需键入\#或\@。系统会先把终端光标移在@之后的下一行，即使它前面有反斜杠。不过不用担心，该@符号已被记录下来了。

反斜杠有时被称为转义字符，它主要用于指明后续字符是特殊字符。要消除反斜杠，需要键入两个删除字符，即\##。知道这是为什么吗？

键入的字符在到达最终目的地之前，要经过一系列程序的检查和解释，至于如何解释它们，则完全取决于它们如何结束以及如何到达终点。键入的每一字符立即在终端显示，除非回应功能被关闭（而这是极少见的）。在按回车键之前，字符还未被内核保留，所以键入的错误可以用删字符或消行符纠正。当删字符或消行符前端有反斜杠时，内核会去掉反斜杠，并且不加解释地保留后续字符。

按下回车键之后，保留的字符被送到从终端读取字符的程序。程序也许会以特殊的方式解释字符，例如，假若字符的前面有反斜杠，shell 就关闭特定的字符解释器。这在第 3 章会进一步探讨。总之应该认识到，内核处理删字符或消行符，而反斜杠只有在删字符或消行符之前，才会被内核处理；其他剩下的字符都可以被别的程序解释。

练习 1-1　说明下面命令的结果：

```
$ date\@
```

练习 1-2　多数 shell（尽管第 7 版 shell 并非如此）把#解释为注释，并忽略从#开始至行尾的全部文字。按照这一点，说明下列的文本，假设删字符也是#。

```
$ date
Mon Sep 26 12:39:56 EDT 1983
$ #date
Mon Sep 26 12:40:21 EDT 1983
$ \#date
$ \\#date
#date: not found
$
```

1.1.7 继续键入

在键入的同时，内核读取所键入的内容，即使系统在忙于其他事务时也是如此，所以你可以用最快的速度键入想输入的内容，即使有命令正在显示也没有关系。如果此时系统正在输出，则输入字符会同输出字符混合在一起，但它们会另行存储并以正确的次序解释。你可以一条接一条地输入命令，而无需等待它们完成，甚至不需要等待它们开始。

1.1.8 中止程序

通过按 Delete 键，可以中止大多数命令。在某些终端上，Break 键也起作用。当然这是与系统相关的。在少数程序中，如文本编辑器，Delete 键中止了程序的执行过程但仍停留在该程序中。关闭终端或挂断电话会中止大多数的程序。

如果只打算暂停输出，如在要避免某些关键信息从屏幕上消失时，可以键入 Ctrl + s，输出会立即停下来，程序会被挂起直到再次启动。要继续输出，可键入 Ctrl + q。

1.1.9 注销

注销的正确方法是按下 Ctrl + d，而不用输入命令。这样就通知 shell，输入中止了（至于它到底是如何起作用，下一章再作说明）。实际上，可关闭终端或挂断电话来注销，但这样做是否真正注销了，取决于不同的系统。

1.1.10 邮件

系统提供一套邮件系统用于与其他用户通信，所以有时在登录时，在第一个提示符之前会看到下列消息：

```
You have mail.
```

要阅读邮件，请键入

```
$ mail
```

邮件消息会显示出来，一次一条消息，最新的消息首先出现。在每一项消息后面，邮件等待用户的行动指示。一般有两种基本响应方式：d，删除该条消息；回车键，不删除该条消息（消息会保留，下次可以阅读消息）。其他处理包括：p，显示消息；s *filename*（文件名），以你起的名称保存消息；q，从 mail 中退出。（假如不知道文件这个概念，简单地把它看成一个用选定的名字保存信息的地方，以备今后再使用。文件是 1.2 节的主题，而且本书很多部分都在讨论它。）