计 算 机 科 学 丛 书

# UNIX 编程环境

# The UNIX Programming Environment

(美) Brian W. Kernighan Rob Pike 著
陈向群 等译

机械工业出版社
China Machine Press
Prentice Hall

计算机科学丛书

# UNIX编程环境

(美) Brian W.Kernighan
Rob Pike 著
陈向群 等译

机械工业出版社
China Machine Press

本书对UNIX操作系统的编程环境做了详细而深入的讨论，内容包括UNIX的文件系统、Shell、过滤程序、I/O编程、系统调用等，并对UNIX中的程序开发方法做了有针对性的指导。本书内容深入浅出，实例丰富，无论是UNIX系统的初学者还是专业人员都可从本书受益。本书亦可作为大学生、研究生学习UNIX的教材。



**本书版权登记号：图字：01-1999-1567**

**图书在版编目(CIP)数据**

UNIX编程环境/(美)柯尼汉(Kernighan, B.W.)，(美)派克(Pike, R.)著；陈向群等译.
－北京：机械工业出版社，1999.10
(计算机科学丛书)
书名原文：The UNIX Programming Environment
ISBN 7-111-07115-8

TP316-62/01

Ⅰ.U… Ⅱ.①柯… ②派… ③陈… Ⅲ.UNIX操作系统－程序设计 Ⅳ.TP316

中国版本图书馆CIP数据核字(1999)第23881号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码100037）
责任编辑：陈 谊
北京牛山世兴印刷厂印刷·新华书店北京发行所发行
1999年10月第1版·2001年9月第4次印刷
787mm×1092mm 1/16 · 16.5印张
印数：11001-12000册
定价：24.00元

# 译 者 序

UNIX是当代计算机操作系统的一个成功典范。从计算机操作系统的发展史来看，UNIX超过了以往所有的操作系统，取得了前所未有的成功。

UNIX系统的诞生已经有30年了。许多操作系统在这期间早已寿终正寝，然而没有一位计算机工作者会预言UNIX在可见的今后年代中走下坡路。UNIX不仅在操作系统发展的历史上具有里程碑作用，而且，它在学术上和教学中也是人们首先研究和探讨的为数不多的操作系统之一。UNIX极大地推动了操作系统理论的发展，是所有计算机院校、系所必须开设和研讨的一门课程。

UNIX不仅在学术上取得了成功，多年来，它在商业上的成功也是极为出色的。成百万台的机器在运行着它。而近年来，随着互联网的迅猛发展，它的装机量更是可观，而更重要的是，凡是重要的互联网上的站点，几乎都把UNIX作为首选的Web服务器操作系统。从这些事实来看，UNIX的重要性是不言而喻的。

与其他系统不同的一点是，UNIX的成功和C语言的成功密不可分。UNIX自身就是C高级程序设计语言的编程典范。UNIX推动了C语言的应用和普及，而C语言又反过来促进了UNIX在更广阔领域里的成功。

如今，世界正处于世纪之交，随着Linux操作系统在Internet上的出现，人们又一次看到UNIX操作系统的新的高峰出现在我们面前。Linux是“free UNIX”，即免费的UNIX。Linux遵从国际上相关组织制定的UNIX标准，即POSIX。它的结构、功能以及界面都与经典的UNIX并无二致。Linux不但继承了UNIX的全部优点，而且还增加了一条其他操作系统不曾具备的优点，即Linux操作系统源码全部开放。从某种意义上讲，Linux是UNIX在国际互联网环境下的一次新生。想要学习90年代的Linux吗？请从UNIX开始，请从这本书开始。

应该说本书不是一本普通的有关UNIX的书籍。本书的写作日期是1984年，十多年岁月的考验更显露其华彩。在书籍的海洋中，本书是有关UNIX编程环境的一本举世公认的经典著作。迄今为止，在这个领域中，还没有其他的书能够替代它。这是一本深入浅出地讲解如何在UNIX环境下进行程序设计的优秀书籍。书中引用大量编程实例，由浅入深地讲解如何使用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来。正因为本书的重要价值，所以在国际上，许多一流大学的相关专业都采用本书作为首选教科书。

想要成为一名UNIX系统上的程序设计高手吗？请仔细阅读本书，并上机实践。想要掌握C语言的精妙技巧吗？请在机器上试验本书中的实例并尝试做出改进。

本书对初学者也同样适用。跟随本书的讲解，一边学习，一边上机实践，初学者就一定会成为编程好手。

本书适合作为大专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，以及想学习和掌握Linux的人员，本书也不失为一本难得的参考书。

参加本书翻译工作的还有：张乃琳、李茹、汤敏、王梦秋、叶松和翁念龙。由于时间有限，书中难免有误，请读者指出，以便再版时改正。

1999年2月于北京

# 序　言

“UNIX安装的数量已经增至10台，超过了预期。”

《UNIX程序员手册》，第2版，1972年6月

UNIX[㊀]操作系统于1969年首次在贝尔实验室一台闲置的DEC PDP-7计算机上启用。当时，Ken Thompson从Rudd Canaday、Doug McIlroy、Joe Ossanna和Dennis Ritchie处得到启示，并在他们的支持下，编写了小型通用分时系统，其适用性好、可信度高，吸引了热心的用户，并最终装备在一台较大的计算机——PDP-11/20上。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，UNIX获准进入大学“用于教育目的”，而且，几年之后便进入了商业领域。在此期间，UNIX系统在贝尔实验室日益发展状大，成功地进入到实验室、软件开发项目、字处理中心和电话公司运行支持系统。从那时起，UNIX开始向全球扩散，从微型计算机到大型机，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？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型机到大型机，这是一个强大的商业优势。第二，源代码是用高级语言编写的，且很容易得到，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编程环境非常丰富而且容易取得成效。

尽管UNIX系统引入了很多创新的程序和技术，但并非其中某个单独的程序或想法使得工作完美无缺。相反，是程序设计的方法，即应用计算机的基本原则使系统富有成效。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。许多单独的UNIX程序只能完成相当简单的工作，然而，在与其他程序结合起来使用时，它们就成为通用而有效的工具。

本书的目的是传授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，除了程序组合和运用程序进行创建程序的主题外，我们还用了大量篇幅对单个工具进行讨论。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，还要了解它们是怎样与环境相匹配的。

尽管UNIX系统得到了广泛应用，但精通的用户却越来越难找到。我们经常发现富有经验的用户(包括我们自己在内)只能找到问题的笨解法，或者只会写程序做现有的工具易于处理的工作。当然，没有一定的经验和对其透彻的理解，更不易获得精巧的解法。我们希望无论是新手还是老用户，通过阅读本书，都能够增强理解力，从而使系统的使用更加有效和有趣。

---

㊀ 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使用MULTICS操作系统。

希望读者更好地使用UNIX。

我们把焦点对准程序员，希望通过本书的学习，能使其工作更有效，从而也使程序组的工作更有成果。尽管我们的主要对象是程序员，但学习本书的前4章或前5章并不需要具有编程经验，因此，对于其他读者来说，本书也是很有帮助的。

我们尽可能尝试使用真实的而不是假设的例子，来表述我们的观点。有些程序尽管在书中作为实例，但也可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节点。章末是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，也用于编写程序。第3章讲述怎样按要求使用shell：创建新命令、命令变量、shell变量、基本控制流和输入输出重定向。

第4章介绍过滤程序，它是当数据流通过时，对数据执行简单转换的程序。首先涉及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，作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编程语言。使用这些程序，有时再加以与shell的配合，常常能完全避免流于常规的程序设计。

第5章讨论如何使用shell编程，以供他人使用。主要内容包括更高级的控制流和变量、陷阱和中断处理。本章的实例充分利用了sed、awk和shell。

至此读者将完全领会运用shell和其他现有程序所能进行的工作。第6章讲述有关运用标准I/O库编写新程序的内容。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，介绍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节点、进程和信号。

第8章讨论程序开发工具。yacc是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可以生成词法分析程序。上述讨论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，本章可以独立于其他章节阅读。

附录A概括了标准编辑器ed。尽管许多读者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程序(如grep和sed)的核心，而且只为此理由就值得一读。

附录B是用于第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，列出了所有的源代码以便于阅读。

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。加利福尼亚大学Berkeley分校提供的系统源自第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版派生的。

我们尽可能地把不同版本中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，但对于特定的细节，我们选择了和第7版中有关内容一致的描述，因为它构成了广泛应用的绝大多数UNIX系统的基础。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。我们知道讲授使用、方法和风格要比仅仅讲授细节重要得多。《UNIX程序员手册》是信息的标准来源，书中的许多参见内容是指参见这个手册。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。

最后，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、检查或者指正，探索其局限和变化。请边阅读、边实践，再回顾小结，然后进一步地阅读。

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，我们希望通过对本书的阅读，读者也得出这一结论。

Brian Kernighan
Rob Pike

# 目　录

# 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分配给用户的程序。它让用户运行其程序；它控制与机器连接的外围设备(硬盘、终端、打印机等等)；它提供一个文件系统用以管理诸如程序、数据及文档一类信息的长期存储。

广义地看，UNIX通常不仅包含内核，还包括一些基本程序，诸如编译器、编辑器、命令语言、用于复制和打印文件的程序等。

从更广的角度来看，UNIX可以包括由用户开发的、用于运行用户定制系统的程序，如文档处理工具、统计分析程序以及图像软件包。

这些有关UNIX的解释究竟正确与否，取决于读者所面对系统的应用级别。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，很难以最佳方式使用可用资源。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。本章帮助读者尽快地学会使用UNIX系统，是概述，而不是手册；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、文件打印、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，而是为了教会你更好地使用手册中的命令。另外，本书中所叙述的内容可能同你所使用的系统有所差别。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：不要害怕做试验。如果你是初学者，那么放心，不会出现伤害自身或其他用户的事。要通过试验了解事物原理。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试验。

## 1.1　起步

### 1.1.1　有关终端和击键的一些预备知识

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。如果对下面的叙述迷惑不解，请询问身旁的专家们。

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，这样就可以看到所输入的是什么。但是有的时候，比如在键入保密口令时，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，这些键中最重要的是Return键。Return键说明一行输入的结束；系统作出反应并把屏幕上的光标移到下一行的起始处。必须按下Return键后，系统才会对键入的字符作出解释。

Return键是一种控制字符——即不可见的字符，它控制着终端上输入输出的某种状况。在任一台终端上，Return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Control键(Control键有时被称作为Ctl键、Cntl键或Ctrl键)，同时按下另一个键，通常是一个字母来输入。例如，Return可以通过按下Return键输入，或者，按住Control键并键入m键。所以Return键又可称为Control-m键，它可以写成Ctl-m。其他的控制字符有：Ctl-d，表示程序输入到此结束；Ctl-g，终端上的振铃鸣响；Ctl-h，通常称为Backspace(退格键)，它可在纠正错误的输入时使用；还有Ctl-i，通常称作Tab，它使光标往前跳到下一个Tab的停止位，这一点非常类似于普通的打字机。在UNIX系统中Tab间隔为8个空字符。在多数终端上Backspace和Tab都有各自的键符。

有两个键具有特殊的含义：一个是Delete，有时又称作为Rubout或某些缩写；另一个是Break，有时又称作为Interrupt。在多数UNIX系统中，按Delete键可立即中止程序，而不等待程序完成。在某些系统中，Ctl-c提供这项功能。而在有些系统中，根据终端的连接方式，Break可能是Delete或Ctl-c的同义词。

### 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，你键入的内容用斜体字符表示，计算机回应的用正体表示。

```
建立连接：必要时拨号或打开电源。
系统应该显示
login: you                      键入名称后按回车
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

?                               回车以到下一条消息
From mary Sun Sep 25 19:07 EDT 1983   下一条消息
Lunch at noon tomorrow?

? d                             删除该消息
$                               没有消息了
$ 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，有时人们也进行一些其他任务。本节的下面部分将讨论这段会话，顺便分析其他一些有用的程序段。

### 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，但最习惯于小写字体的设备；它区分大小写！如果你的终端只有大写字符(类似某些视频和便携式终端)，那么日子就难过了，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。如果仍旧不成功，轻按几次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: 登录消息后，用小写字母键入登录名，然后按回车键。如果要求口令，系统会有提示，而且在输入口令时，屏幕不会显示所输入的口令。

登录一旦成功，系统便响应一个提示符，通常是一个单个字符，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的程序打印出来的，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，命令是请系统完成某项工作的要求。我们将使用“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Return键。系统应该回应日期和时间，然后显示下一个提示符，所以整个的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Return，不要键入$。如果你觉得系统没有反应，可以按Return键；应该会有响应。以后不再提Return键，但是在每一行的结尾都需要它。

下一个要试用的命令是who，它说明当前有哪些人登录上机了：

```
$ 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表明teletype，终端的一个学名)。其余信

息告知登录的日期和时间。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

有时终端的行为会是很奇怪的，比如，一个字符也许会显示两次，或者Return键可能不把光标置到下一行的首列。通常可以把终端关闭后再开启，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。如果终端具有计算机可设置的Tab跳格键功能，命令tabs会正确地为用户设置跳格位置。(这样读者就需要键入

```
$ tabs 终端类型
```

从而使系统正确工作——请参阅手册中有关tabs命令的说明)

### 1.1.6 键入错误

如果出现了键入错误，并且在按Return键前发现了它，可有两个方法纠正：一次删除一个字符或者删行重新键入。

如果键入消行符，其缺省值是@，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)。所以如果键入出错太多，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，这在视频终端上很成功。你可以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。稍后，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？如果在#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到在@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。

反斜杠有时被称作escape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：\##。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和消行符纠正。当删字符和消行符前端有反斜杠时，内核取消反斜杠，并且不加解释地保留后续字符。

按下Return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依次解释字符，例如，假若字符的前面有反斜杠，shell就关闭专用的解释器。这在第3章会进一步探讨。总之应该认识到，内核处理删字符和消行符，而反斜杠只有在删字符和消行符之前，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，即使系统正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，那么输入字符会同输出字符混在一起，但它们会另行存储起来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或是开始。

### 1.1.8 中止程序

通过键入字符Delete，可以中止大多数命令。在某些终端上，Break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中止了程序的执行过程但仍留在该程序中。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，例如，在要避免某些关键信息出现在屏幕上时，可以键入Ctl-s，输出会立即停下来，程序被挂起来直到再次启动。要打算继续输出，可键入Ctl-q。

### 1.1.9 注销

注销的正确方法是按下Ctl-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，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消息

```
You have mail.
```

要阅读邮件，请键入

```
$ mail
```

邮件消息会打印出来，一次一条消息，最新的消息首先出现。在每一项后面，邮件等待用户的行动指示。有两种基本响应方式：d，删除该条消息；Return，不删除(消息会保留，下次可以阅读消息)。其他处理包括：p，打印消息；s filename(文件名)，以规定的名字保存消息；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。)

同已经介绍的程序相比，mail可能有所不同，它有许多变量。请阅读手册以了解细节。

给他人发邮件很简单。假设要邮往登录名为nico的人，最方便的方法是这样的：

```
$ mail nico
在此键入邮件文本，可键入任意长度的文本，输入完毕后按
Ctl-d
$
```

Ctl-d指明信件的结尾，通知mail没有输入了。如果在编辑信件的中途改变了主意，可以按Delete键，而不须使用Ctl-d。写到一半的信件将存放在名为dead.letter的文件中而不是邮出。

作为练习，给自己发封邮件，然后键入mail读取它(它不像听起来那么奇怪——这是一个简便的提醒机制)。

发邮件还有其他一些方法——发出一封事先准备好的信件，或一次给许多人同时发信，也可以发信给使用其他机器的人们。有关细节请参阅《UNIX程序员手册》第1节中的有关mail的叙述。今后我们将用标记mail(1)来表征手册第一节中有关mail的叙述。本章讨论的所有命令都可以在第1节中找到。

系统还有日历服务(参见(calendar(1))；如果你没有使用过，我们会在第4章介绍如何设置它。

### 1.1.11 用户间通信

如果你的UNIX系统是多用户的，某一天，蓝色屏幕上可能会出现如下类似的语句

```
Message from mary tty7 . . .
```

并伴随着一声蜂鸣声。Mary打算写信给你，但是除非采取明确的行动，否则不能写回信。若要回应，请键入

```
$ write mary
```

这样，就建立了一条双向通信途径。现在Mary在她的终端上键入的内容，会出现在你的终端上，反之也一样，尽管通信很慢，但毕竟不象是同月球通话。

如果你正在进行某项工作，那么必须进入可以键入命令的状态。通常，正在运行的程序必须中止，但是有些程序，诸如编辑器以及write本身，利用“!”命令可暂时切换到shell——请参阅附录A中的表A-2。

write命令没有规则，所以要有一个协议，以免你键入的内容同Mary键入的内容混到一起。通常的做法是，每一方以(o)结束，它表明“完毕”，打算退出的信号是“oo”，其含义是“完毕并退出”。

| Mary的终端: | 你的终端: |
| --- | --- |
| **$ write you** | |
| | **$ Message from mary tty7...** |
| | **write mary** |
| **Message from you ttya...** | |
| ***did you forget lunch? (o)*** | |
| | **did you forget lunch? (o)** |
| | ***five@*** |
| | ***ten minutes (o)*** |
| **ten minutes (o)** | |
| ***ok (oo)*** | |
| | **ok (oo)** |
| | ***ctl-d*** |
| **EOF** | |
| ***ctl-d*** | |
| | **$ EOF** |
| **$** | |

要退出write也可以按Delete键。注意，键入的错误不会在Mary的终端上出现。

如果要写信给某个未登录的人，或者收件人不想被打扰，系统会通知你。如果目标登录了，但是在一个间隔后没有回答，也许对方太忙或者不在终端旁，只要按Ctl-d或Delete键就可退出。如果你不想被打扰，可利用mesg(1)。

### 1.1.12 新闻

许多UNIX系统提供了一种新的服务，可使用户随时了解一些有趣无趣的事件。试试键入

```
$ news
```

也有大量的UNIX系统网络存在，它们可以通过电话拨号连接，有关netnews和USENET，请询问你身边的专家。

### 1.1.13 手册

《UNIX程序员手册》描述了大量读者需要了解的系统信息。第1节介绍命令，包括本章讨论的内容。第2节讨论系统调用，这是本书第7章的主题，而第6节是关于游戏的内容。其余章节讨论C程序员使用的函数、文件格式，以及系统维护(这些章节的编号随着系统的不同而变

化)。在开始使用手册时，不要忘记使用各种形式的手册索引。可以快速地翻阅手册索引，以了解相关命令的内容。还有关于UNIX系统运行的原理。

通常手册用在线形式存放，这样可以在终端上阅读。如果你不知道做法，又找不到专家协助，可以在终端上用命令man打印出有关页面。下面例子是读取who命令：

```
$ man who
```

而

```
$ man man
```

则说明了man命令。

### 1.1.14 计算机辅助教学

你的系统中也许有learn命令，这个命令提供了有关文件系统、基本命令、编辑器、文档准备，甚至包括C语言程序设计的计算机辅助教学。请试一下

```
$ learn
```

它会告诉你从哪儿开始，可以做什么。如果不行，可以再试一下teach。

### 1.1.15 游戏

游戏往往不被正式承认，但熟悉计算机和终端的最好方式之一，就是玩游戏。UNIX系统提供了适量的游戏，通常在本地机上就有。问问周围的人，或者查看手册的第6节。

## 1.2 文件和常用命令

在UNIX系统中信息存储在文件中，它很像日常的办公室文件。每个文件有名字，内容，存放地点以及某些管理信息，诸如所有者以及文件大小等。文件可能是一封信，或者是人名及地址清单，或者是源程序，或者是供某个程序用的数据，甚至是程序的可执行形式以及其他的非文本类型材料。

UNIX文件组织结构使你可以维护自己的文件而不会妨碍其他人的文件，并且也防止他人干预你的文件。有大量的程序可对文件进行操作，但是现在，我们只介绍最频繁使用的那些命令。第2章是关于文件系统的具体讨论，并引入了许多与文件有关的其他命令。

### 1.2.1 创建文件

如果想录入一篇文章、一封信或一个程序，那么怎样把这些信息存放在机器中呢？这类任务大多是用文本编辑器完成的，它是一个在计算机中存放和操作信息的程序。在每个UNIX系统中几乎都有一个屏幕编辑器，它利用现代终端的特点，显示你对文件所进行的编辑效果。两个最流行的文本编辑器是vi和emacs。在这里我们将不介绍任何屏幕编辑器，其原因部分是由于篇幅所限，部分是由于不存在一个标准的文本编辑器。

不过，有一个名为ed的编辑器肯定在你的系统中存在。它不需要特定的终端功能，因此可以在任何终端上工作。它也构成了其他基本程序的基础(包括一些屏幕编辑器)，所以这个程