



第二版
上卷



UNIX 超级工具

Jerry Peek, Tim O'Reilly, and Mike Loukides 著

张志龙 李波 赵宏刚 李海燕 译

O'REILLY® 中国电力出版社

UNIX 超级工具

(第二版) 上卷

Jerry Peek, Tim O'Reilly, and Mike Loukides 著

~~张圣波~~ 李波 赵宏刚 李海燕 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly & Associates, Inc. 授权中国电力出版社出版

中国电力出版社

UNIX 超级工具

(第二版) 下卷

Jerry Peek, Tim O'Reilly, and Mike Loukides 著

张志龙 李波 赵宏刚 李海燕 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly & Associates, Inc. 授权中国电力出版社出版

中国电力出版社

图书在版编目 (CIP) 数据

UNIX 超级工具 (第二版) / (美) 皮克 (Peek, J.) 等编著; 张志龙等译. - 北京: 中国电力出版社, 2002.8

书名原文: UNIX Power Tools, Second Edition

ISBN 7-5083-0947-2

I. U... II. ①皮... ②张... III. UNIX 操作系统 IV. TP316.81

中国版本图书馆 CIP 数据核字 (2002) 第 036754 号

北京市版权局著作权合同登记

图字: 01-2002-1211 号

©1997 by O'Reilly & Associates, Inc.

Simplified Chinese Edition, jointly published by O'Reilly & Associates, Inc. and China Electric Power Press, 2002. Authorized translation of the English edition, 1997 O'Reilly & Associates, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly & Associates, Inc. 出版 1997。

简体中文版由中国电力出版社出版 2002。英文原版的翻译得到 O'Reilly & Associates, Inc. 的授权。此简体中文版的出版和销售得到出版权和销售权的所有者——O'Reilly & Associates, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式重制。

书 名 / UNIX 超级工具 (第二版)

书 号 / ISBN 7-5083-0947-2

责任编辑 / 夏平

封面设计 / Edie Freedman, 张健

出版发行 / 中国电力出版社 (www.infopower.com.cn)

地 址 / 北京三里河路 6 号 (邮政编码 100044)

经 销 / 全国新华书店

印 刷 / 北京市地矿印刷厂

开 本 / 787 毫米 × 1092 毫米 16 开本 41 印张 595 千字

版 次 / 2002 年 12 月第一版 2002 年 12 月第一次印刷

印 数 / 0001-3000 册

全套定价 / 128.00 元

UNIX 超级工具

上卷

O'Reilly & Associates 公司介绍

为了满足读者对网络 and 软件技术知识的迫切需求,世界著名计算机图书出版机构 O'Reilly & Associates 公司授权中国电力出版社,翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly & Associates 公司是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司,同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》(被纽约公共图书馆评为二十世纪最重要的 50 本书之一)到 GNN (最早的 Internet 门户和商业网站),再到 WebSite (第一个桌面 PC 的 Web 服务器软件),O'Reilly & Associates 一直处于 Internet 发展的最前沿。

许多书店的反馈表明,O'Reilly & Associates 是最稳定的计算机图书出版商——每一本书都一版再版。与大多数计算机图书出版商相比,O'Reilly & Associates 公司具有深厚的计算机专业背景,这使得 O'Reilly & Associates 形成了一个非常不同于其他出版商的出版方针。O'Reilly & Associates 所有的编辑人员以前都是程序员,或者是顶尖级的技术专家。O'Reilly & Associates 还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家,而现在编写著作,O'Reilly & Associates 依靠他们及时地推出图书。因为 O'Reilly & Associates 紧密地与计算机业界联系着,所以 O'Reilly & Associates 知道市场上真正需要什么图书。

如何使用本书

章节号

前两个数字表示章号，后两个数字表示节号。章节号用于全书的交叉参考引用。

字典风格的页眉

有时在当前页小节的标题不可见，因此使用此条目表示每页的开始和结束。在双页上，该数字位于左上角，表示本页第一个小节的章节号（此节标题可能在前一页上）；在单页上，该数字位于右上角，表示本页最后一个小节的章节号。

CD-ROM

如果你不想自己输入此脚本，或者我们正在讨论一个书中未列出的C程序，那么你可以从本书附带的CD-ROM来安装它。此图标下给出了安装程序名。请参考第52.05节中完整的安装指令。

螺丝钉图标

小心这个特性，否则你将会陷入麻烦。

页脚

在单页内侧的页脚处是章的标题，这样读者总可以知道章的标题。在双页的页脚处是章号。

21.11 比较安全的删除、移动和拷贝



默认情况下，UNIX中的`rm`、`mv`和`cp`实用程序完成用户告诉它们要完成的任务而不会去询问用户，即使这个任务是要重写目录中的已经存在的文件或者删除其中的所有文件，它们也会照做不误。如果用户希望能够更安全些，那么可以将如下的这些别名添加到用户的shell启动文件（2.02）中：

```
C shell      ksh, bash
alias rm /bin/rm -i      alias rm="/bin/rm -i"
alias cp /bin/cp -i      alias cp="/bin/cp -i"
alias mv /bin/mv -i      alias mv="/bin/mv -i"
```

（在用户的系统上，路径名`/bin/`可能不同，用户可以完全地忽略它，但是在忽略它的时候要当心别名循环。）那么，当用户在命令行键入`cp`或者`mv`时，这些命令就会在破坏一个已经存在的文件前提示用户。并且`rm`命令总会在删除每个文件之前提示用户（使用一个通配符，这可能要花上一些时间！）。这些别名会设置`-i`（交互式）选项，许多版本的`cp`、`mv`和`rm`都有该选项。



`cp`和`mv`的GNU版本有`-b`选项。它们还有另外一个选择：`-b`选项，该选项会自动地为任何将被重写的文件生成一个备份。



如果没有这些别名，它们就会很危险。如果用户在沒有这些别名时改变到一个新的shell或者使用另外一个账户，那么在用户意识到没有这些别名保护之前可能会破坏一些文件。因此我在账户上从不使用这些别名中的任何一个。相反，在按RETURN之前我会检查输入的内容。

用户可以输入下面的命令来覆盖这些别名：

```
% rm *.o      C shell
$ command rm *.o
$ /bin/rm *.o  bash
                 Bourne shell
```

代码例子中的交叉参考

这是代码中的交叉参考。
当交叉参考出现在例子中时，所引用的内容和其所所在的章节号显示在左边的页边空白处。

炸弹

如果你看到了炸弹图标，
就请阅读有关的内容，其
章节号在炸弹图标的下
面。此章节给出了你在使
用当前章节中的提示或脚
本时所遇到问题的解释。

概要框

本书中有许多概要框。它
们就某个主题进行了总
结，并向你指出包含有关
例子和解释的章节。

正文中的交叉参考

如果你想了解一些术语的更多信
息，就请阅读与该术语有关的章
节中的内容，其章节号以灰色字
体显示在术语后面的小括号内。

作者姓名的首字母缩写

在前言中列出了作者的全名。

17.09

题，用户想寻找在那之后被修改过的文件，从而可以检查它们。很明显，用户希望的是比“给我在最近 24 小时内被修改的所有文件”更精确的内容。

较新版本的 *touch* (21.07) 以及其他可以免费获取的类似命令，可以创建有任何时间标记的文件，即用户可以用 *touch* 让一个文件回溯到过去任何一点（或者用它把日期推到将来的某一点）。这个特性，如果组合 *find* 的 *-newer* 操作符，就可以让用户进行精确到秒甚至更小的比较。

例如，为了创建日期为 3 月 20 日的 4 p.m. 的文件，给出这个命令：

```
hmp 21.02 % touch -t 03201600 /tmp/4PMYesterday
```

(*touch* 的一些版本不需要 *-t* 选项) 然后寻找创建时间比它晚的文件，给出这个命令：

```
% find . -newer /tmp/4PMYesterday -print
```

那么“陈旧的”文件呢？陈旧的文件是“不断的”文件，而 *find* 有一个方便的 NOT 操作符 (!)，刚好用于这个目的。因此假设用户想寻找创建时间介于 1982 年 7 月 3 日 10:46 a.m. 和 1985 年 8 月 4 日 9:37 p.m. 之间的文件，则可以使用下面的命令：

```
% find -t 0703104602 /tmp/file1  
% touch -t 0804213709 /tmp/file2  
% find . -newer /tmp/file1 ! -newer /tmp/file2 -print  
% rm /tmp/file1[2]
```

—ML



25.09

- 对于其他作业，像 *awk* (33.11) 和 *sed* (34.24) 这样的实用程序可以完成用户所希望的工作。在使用之前，用户必须理解如何对它们进行编程。

—JP

25.10 压缩额外的空白行

阅读有许多空白行的输出是很浪费屏幕空间的。比如，有些系统 V 版本的 *man* (50.01) 显示帮助文档中间的所有空白行。要结束这样的情况，可以通过 *cat* *-s* 阅读文件或者用管道输出文件。（许多 *more* (25.03) 版本都有一个相似的 *-s* 选项。）*-s* 选项用单个空白行代替多个空白行。（如果你的 *cat* 没有 *-s*，就请参考本节末尾的 *sed* 方法。）

cat -s 可能不总是有效的。问题通常是在“空”行中有 SPACE、TAB 或者 CTRL-m 字符。改进的方法是让 *sed* 用行中那些不可见的字符“删除”行：

```
% sed 's/[SPACE TAB CTRL-V CTRL-M]*$//1' -e 'f,/1/d' file | cat -s
```

在 *vi* (31.06) 和许多终端驱动程序 (42.01) 中，CTRL-V 字符引用 CTRL-m (RETURN)，这样该字符就不会结束当前行。

如果用户没有 *cat -s*，那么 *sed* 可以完成这两个作业：

```
% sed -e 's/[SPACE TAB CTRL-V CTRL-M]*$//1' -e 'f,/1/d' file
```

—JP

25.11 crush: 一个略过所有空白行的 cat

我有时有一组文件或者仅有一个文件有许多空白行。有些系统有一个 *cat* 的 *-s* 选项，它使 *cat* 将相邻的空白行压缩成一行。如果没有该选项，还可以使用 *crush*。 *crush* 脚本略过值为空的所有行或者仅有空白和/或者 TAB 的所有行。

下面为 *crush*：



```
#!/bin/sed -f  
/[ ]*/d
```

538

第二十五章

译者序

UNIX 操作系统自 20 世纪 70 年代由贝尔实验室推出以来, 20 世纪 80 年代经过一些大学、研究所、工业实验室的应用和发展, 现已成为全世界计算机网络通信、工作站系统的主流工具, 并已经进入商业市场和个人电脑领域。尤其是 20 世纪 90 年代 Internet 的迅速发展, 更为 UNIX 的发展应用推波助澜。目前, 我们的生活处处与 UNIX 操作系统相关。

大多数操作系统不支持自由软件的加入, 而 UNIX 则不然, 它允许成千上万个独立的程序员的工作融合进来。因此, 在 UNIX 的商业化过程中 UNIX 成为了一个兼收并蓄的系统。UNIX 的支持工具众多, 因此, 一本关于 UNIX 实用工具的书不仅要集中于实用程序, 而且更要以如何充分发挥这些实用程序的特性为重点, 因为这些特性已日渐成为 UNIX 的一部分了。

本书就是为针对这一迫切需求而推出的, 书中介绍了各种 UNIX 实用工具。这些实用工具成功地扩展了 UNIX 操作系统的功能, 使得 UNIX 成为了凡人的乐园。在翻译本书的过程中, 我们深深感到 UNIX 的博大精深, 也对 UNIX 开放的设计思想有了更深刻的认识。原文书由多位作者合作完成, 涉及知识面很宽, 对于 UNIX 的一些技巧以及一些常见的疑难问题, 作者提供了各种解决方案。因此, 本书特别适合于有一定基础的 UNIX 开发者和管理人员。同时, 作者对一些基本的概念进行了解释, 对于 UNIX 初学者也有一定的裨益。

本书的一个特点是内容结构比较松散, 并不像大部分书那样需要读者从头至尾通读每一章, 本书通过许多交叉参考和跳转提示, 帮助读者对感兴趣的部分进行跳跃式阅读。本书主要针对 UNIX 单机系统的实用工具, 没有覆盖任何 UNIX 通信和网络的功能——如 E-mail、WWW、News、telnet、ftp 等。关于这部分内容, 读者可参考相关书籍。

本书包括九个部分 53 章, 内容覆盖 UNIX 基础知识、Shell 命令行、文件系统、文件搜索、文本编辑、管理进程、终端与打印机、Shell 编程以及其他工具, 如办公自动化等。本书内容翔实, 示例得当, 深入浅出地讲解了大量工具的使用, 是关于 UNIX 实用工具的一本不可多得的好书。本书适合于 UNIX 中高级用户和编程人员用作手头的工具书, 也可供大专院校、研究机构参考。

本书由张志龙、李波、赵宏刚、李海燕翻译，参加主要部分翻译工作的还有刘建华、郑伟、江帆、李卫红、邓济哲、陈兵华、李桂华等，全书由张晓明、孙彩霞统校，灯芯工作室的其他工作人员承担了全书的录入、校对和审稿等工作。感谢电力出版社的刘江和夏平两位编辑，他们对工作的认真精神，对本书的负责态度，值得我们学习。由于原著由多人协作完成，书中涉及的知识和技术范围较广，尽管我们为此付出了许多努力，但限于水平以及忠于原著的考虑，书中难免还存在一些不当之处，恳请广大读者批评指正。

译者

2002年5月

作者简介

Jerry Peek 从 20 世纪 80 年代初期开始使用 UNIX 系统。他是 UNIX 和 VMS 方面的顾问，开发并教授 UNIX 课程，是 O'Reilly & Associates 的资深作者，也是一名程序员和系统管理员。他现在是计算机专业的一名研究生。Jerry 只要一有时间就徒步或者骑自行车到加利福尼亚的海岸，或者到拉丁美洲一个无名的小地方去旅行。

Mike Loukides 是 O'Reilly & Associates 的一名高级编辑。他目前主要关注 Java 编程和网络等课题。他也是 O'Reilly 出版的其他两本书，《System Performance Tunning》和《UNIX for FORTRAN Programmers》的作者。Mike 以前曾经在 Multiflow Computer 公司工作，那时他创建了编程语言上的所有 Multiflow 文档。

除了 Java 和网络以外，Mike 的技术兴趣还包括编程语言、系统管理和计算机体系结构。他还是一位合格的钢琴师——实际上，他是少数的业余钢琴家之一，甚至曾经弹奏过 Olivier Messiaen 的作品。他和他的妻子 Judy、女儿 Alexandra 一起住在美国的康涅狄格州。

Tim O'Reilly 是 O'Reilly & Associates 的创始人和总裁，该公司出版的具有权威性的书籍都拥有针对专业软件开发的合法免费软件。其开拓性的网站 GNN (Global Network Navigator) 是第一个真正意义上的商业网站。

目录

前言	1
第一章 简介	13
1.01 UNIX 特殊在什么地方?	13
1.02 谁接收用户的输入?	15
1.03 将程序设计成协同工作方式	16
1.04 用管道创建新工具	17
1.05 任何人都可以对 shell 编程	19
1.06 用于编辑的超级工具	20
1.07 让用户更加高效	21
1.08 shell 的种类	22
1.09 正在运行哪个 shell?	24
1.10 内部命令和外部命令	25
1.11 shell 如何执行其他命令?	26
1.12 shell 脚本的组成	26
1.13 为什么基本原理很重要?	27
1.14 内核和后台进程 (daemon)	28
1.15 文件名	30

1.16	通配符	31
1.17	扩展文件名	33
1.18	谁处理通配符?	35
1.19	文件系统的树型结构	36
1.20	用户的主目录	38
1.21	创建路径	39
1.22	UNIX 如何跟踪文件: 索引节点	41
1.23	文件访问权限	42
1.24	超级用户 (root)	46
1.25	访问目录	46
1.26	多用户系统可以做什么?	47
1.27	后台处理如何工作?	48
1.28	后台处理的一些注意事项	49
1.29	何时文件变成非文件?	50
1.30	重定向输入和输出	51
1.31	X Window 系统	51
1.32	一个大的漏洞	52
1.33	UNIX 联网和通信	53
1.34	UNIX 有什么问题?	55

第一部分 请放轻松

第二章	登录	59
2.01	定制 shell	59
2.02	shell 设置文件 —— 哪个、何处和为什么	59
2.03	shell 设置文件的内容	62
2.04	改变账户设置的提示: 使 shell 随时可用	63
2.05	提高登录速度的技巧	63
2.06	在 shell 设置文件中使用绝对路径名	66

2.07	C shell 设置文件没有被读入时该怎么办?	67
2.08	识别登录 shell	68
2.09	用设置提示符测试加速 C shell	69
2.10	设置提示符测试中的注意事项	70
2.11	使用 \$- 测试的快速 ksh 和 bash 启动	71
2.12	不同终端的自动设置	71
2.13	设置每台主机所需的 .cshrc.\$HOST 文件	74
2.14	motd.diff: 在登录信息中显示新内容	75
2.15	整理登录: 仅显示登录信息一次	76
2.16	已获得认可的 shell: 使用未获认可的登录 shell	77

第三章 退出 80

3.01	退出时执行命令	80
3.02	在 Bourne/Korn shell 退出时执行命令	81
3.03	电子幸运饼	82
3.04	自动清除文件	82
3.05	阻止意外的 C shell 退出	83
3.06	阻止意外的 Bourne shell 退出	84
3.07	用 screen 分离会话	85
3.08	我使用的 tty 是什么?	86

第四章 组织主目录 87

4.01	什么? 我们自己组织?	87
4.02	存放程序和脚本的 bin 目录	88
4.03	组织非执行脚本	88
4.04	Emacs 小程序目录	89
4.05	私有 (个人) 目录	89
4.06	命名文件	90
4.07	生成更多的目录!	91
4.08	生成目录的方法	92

4.09	用 .exrc 文件建立 vi	93
4.10	用 whereiz 找出所有的命令版本	94

第五章 设置用户终端类型 96

5.01	关于终端有许多内容需要了解	96
5.02	终端数据库的思想	96
5.03	登录时设置终端类型	98
5.04	用 tset 设置 TERMCAP 变量	101
5.05	查询用户终端类型: qterm	101
5.06	检查清单: 登录时终端挂起	104
5.07	termcap 和 terminfo 能完成什么和不能控制什么?	105
5.08	终端转义序列	106
5.09	设置擦除、终止和中断字符	107
5.10	找出可以使用的终端名	109
5.11	用 tset 初始化终端	111
5.12	用 tput 初始化终端	112

第六章 shell 及环境变量 114

6.01	环境变量有什么好处?	114
6.02	父 - 子关系	116
6.03	预定义的环境变量	117
6.04	PATH 环境变量	119
6.05	PATH 和 path	121
6.06	TZ 环境变量	121
6.07	日本现在是什么时间?	122
6.08	shell 变量	123
6.09	特殊的 C shell 变量	125
6.10	用不同的临时环境执行命令	127

第七章 设置用户 shell 提示符 128

- 7.01 为什么要改变用户提示符? 128
- 7.02 设置提示符基础 129
- 7.03 C shell 提示符在 vi、rsh 等中造成的麻烦 129
- 7.04 用内置命令进行快速提示符设置 130
- 7.05 多行 shell 提示符 132
- 7.06 用户终端状态行中的对话信息 133
- 7.07 初学者的“菜单提示符” 135
- 7.08 在 shell 提示符中突出显示 135
- 7.09 用 \$SHLVL 显示 subshell 层 137
- 7.10 空白的 shell 提示符有什么好处? 139
- 7.11 用户提示符中的 dirs: 比 \$pwd 好 140
- 7.12 通过外部命令发出信号来设置变量 141
- 7.13 bash 中的预提示命令 143

第二部分 让计算机承担繁琐的工作

第八章 shell 如何解释用户的输入 147

- 8.01 shell 能做什么? 147
- 8.02 bash 简介 148
- 8.03 tcsh 简介 149
- 8.04 命令评价和意外地覆盖文件 150
- 8.05 命令行的评价 (evaluation) 151
- 8.06 输出命令行参数 153
- 8.07 设置用户搜索路径 156
- 8.08 用户不应该运行的命令目录 158
- 8.09 别名中的通配符 159
- 8.10 eval: 当用户需要另外一个机会时 160
- 8.11 bash 将用哪一个? 162

8.12	C shell 将使用哪一个?	163
8.13	是“2>&1>file”还是“>file 2>&1”?	166
8.14	Bourne shell 引用	166
8.15	Bourne shell 引用和 C shell 引用之间的不同	171
8.16	引用处理文件名中的特殊字符	173
8.17	verbose 和 echo 变量显示引用	174
8.18	here 文档	175
8.19	“特殊”字符和操作符	176
8.20	有多少个反斜杠?	180

第九章 快捷地使用命令行 182

9.01	UNIX 命令行的特殊之处	182
9.02	用行删除或字擦除快速修复一行	183
9.03	用 CTRL-r 重新显示命令行	184
9.04	能否使用通配符创建文件?	185
9.05	用 {} 建立串	185
9.06	串编辑 (冒号) 操作符	187
9.07	ksh 和 bash 中的串编辑	189
9.08	文件名完成: 加速文件名输入	190
9.09	在文件名完成中不要匹配无用文件	192
9.10	文件名完成并不总是万灵丹	192
9.11	用 foreach 循环重复一个命令	193
9.12	Bourne shell 的 for 循环	195
9.13	多行命令, 次提示符	196
9.14	将 here 文档用于格式化信件等	197
9.15	针对复杂命令的一次性脚本	198
9.16	命令替换	199
9.17	用临时文件处理一系列文本	201
9.18	进程替换	202
9.19	超前输入	205