序

真的！我已经 30 年未写 Shell 脚本了？！？现在仔细想想，我想应该有吧，虽然一开始只是作些简单的工作（早期的 UNIX Shell，在 Bourne Shell 之前，是极为原始的，因此要写个实用的脚本是很难的事，幸好那段日子并不长）。

近几年来，Shell 一直被忽略，是一个不受重视的脚本语言。Shell 虽然是 UNIX 的第一个脚本语言，但它仍是相当优秀的。它结合了延展性与效率，持续保有独具的特色，并不断地被改良，使它们多年来一直能与那些花招很多的脚本语言保持抗衡。GUI 是比命令行 Shell 更流行的用户界面，但脚本语言时常都是这些花哨的屏幕图形界面最强有力的支柱，并一直称职地扮演这个角色。

Shell 需依赖其他程序才能完成大部分的工作，这或许是它的缺陷，但它不容置疑的长处是：简洁的脚本语言标记方式，而且比 C（还有其他语言）所编写的程序执行更快、更有效率。它使用通用的、一般用途的数据表示方式，文本行，在一个大的（且可扩展的）工具集中，让脚本语言能够搭配工具程序，产生无穷的组合。用户可以得到比那些独占性软件包更灵活、功能更强大的工具。Shell 的早期成功即以此法强化 UNIX 的开发哲学，构建一套专门性、单一目的工具，并将它们整合在一起做更多的事。该原则接着鼓励了 Shell 的改良，允许用这种方式完成更多的工作。

Shell 脚本还有一个超越 C 程序的优势，同样也优于其他脚本语言的地方，可用一般方式轻松地读取与修改。即便不是C的程序设计人员，也能像现今许多系统管理人员一样，很快就能接受 Shell 脚本。如此种种，让 Shell 脚本成为延展用户环境与定制化软件包的重要一环。

的确，它其实有一种“周而复始”的特性，在我看过这么多软件项目之后。项目将简单的 Shell 脚本置于关键位置，让用户容易地从他们的角度来定制软件。然而，也因为这

些项目的 Shell 脚本与周围的 C 程序码相比较，要更容易解决问题，所以不断产生更复杂的脚本。最后，它们终于复杂到让用户很难轻易地处理（我们在 C News 项目里的部分脚本就拥有著名的 Shell 压力测试，完全未考虑用户的立场），且必须提供新的脚本集合，供用户进行定制……

长久以来，一直都没有编写 Shell 脚本相关的好书出现。UNIX 程序设计环境方面的书籍偶有触及这方面议题，但通常只是简短带过，作为它众多主题的一部分，有些写得不错的书也很久没有更新了。好的参考文件应该是针对各种不同的 Shell 讨论，但必须是贴近新手的实战手册，涵盖工具程序与 Shell，以循序渐近的方式介绍，告诉我们如何得到更好的结果与输出，还要注意到实例面，像是可读性议题。最好它还讨论各式 Shell 的异同，而不是好像世上只有一个 Shell 存在一样。

这本书就是这样的，甚至做到比上面说的还多。至少，它是第一本且最好的一本、内容最新的、以最轻松的方式介绍 UNIX 脚本语言的书。以实用的范例进行解说，让工具充分发挥自己的效能。它包括了标准 UNIX 工具，让用户有个好的开始（对于觉得看手册页有点难的用户来说，这会是个相当不错的参考教材）。我最高兴的是看到将 awk 列入取材范围，这是相当有用且不容忽视的工具程序，适于整合其他工具及简洁地完成小型程序设计的工作。

我建议所有正在编写 Shell 脚本或管理 UNIX 系统的人都要读这本书。我在这本书上学到很多，我想你也会。

—— Henry Spencer
SP Systems

前言

刚开始使用UNIX（注1）的用户与程序员突然面对各式各样的程序时，都会有很多疑问，例如“它们的功能是什么”，还有“我怎么使用它们”。

本书可以回答你这些问题。告诉你如何结合UNIX工具，将其与标准的Shell相结合完成工作。Shell脚本的编写是门艺术，需要的不只是Shell语言的相关知识，还要你对各个独立的UNIX程序有基本认识：为什么会有这些工具，要怎么单纯地使用它们，怎么将它们与其他程序结合应用。

为什么需要学习如何编写Shell命令？因为大部分情况下，中型到大型的问题都能拆成较小的部分，这些小部分也多半都能找到现成的UNIX工具处理。用心编写的好用Shell脚本常常能够比C或C++语言编写的程序更快地解决相同的问题。也可以让Shell脚本提供可移植性，也就是说，可以跨越UNIX与POSIX兼容的系统，有时仅需略作修改，甚至不必修改，即可使用。

谈到UNIX程序时，我们使用工具（tool）这个字。以UNIX工具箱（toolbox）的做法解决问题，长久以来以“软件工具（Software Tools）”哲学（注2）为人所熟知。

瑞士军刀是很多人口袋里的好帮手。它有刀刃、螺丝起子、开罐器、牙签等工具。功能更齐备的，还有其他像拔塞钻、放大镜等工具。瑞士军刀能派上用场的时候很多，虽然用它来修削和进行简单雕刻很不错，但你绝不会拿它来盖狗屋或制作鸟类喂食器。相反，

注1：整本书里，我们都使用UNIX这个字，指的不单单是商用的UNIX系统原始版本，像Solaris、Mac OS X，与HP-UX，还包括可自由取得的类似系统，例如GNU/Linux与各种BSD系统：BSD/OS、NetBSD、FreeBSD，与OpenBSD。

注2：此法因《Software Tools》（Addison-Wesley）这本书而广受欢迎。

做这类工作时你会寻求更专门的工具，例如铁槌、锯子、夹钳或刨刀等。同理，当你在解决程序化问题时，使用专门的软件工具会比较好。

这是给谁看的书

这本书是写给那些在UNIX环境下发现必须写些Shell脚本，以利于工作进行的计算机用户与软件开发人员。例如，你可能是正在念计算科学的学生，手上有学校给你的第一个UNIX系统账号，你想知道在UNIX下更多的东西，例如你的Windows个人计算机无法处理的那些工作（这种情况下，你通常得写几个脚本来定制个人环境）。或者，你可能是个系统管理新手，需要为公司或学校写几个专用程序（可能是处理事件日志文件，账号、账单管理之类的事情）。你也可能是Mac OS的开发老手，但转到崭新的Mac OS X的世界，它的安装程序是以Shell脚本写成。不管你来自哪里，如果你想学Shell脚本，这本书就是写给你的。在这本书里你能学到：

软件工具设计概念与原则

一些好的软件工具设计与实例上的实践规则。我们会解释这些原则，还会在这本书里贯彻执行。

UNIX工具是什么

UNIX的核心工具组会在我们编写Shell脚本时不断地重复使用。我们会介绍Shell与正则表达式的基本概念，并在解决特定问题时展现各种核心工具的用法。除了介绍工具能做什么之外，我们还会告诉你，为什么要使这个工具，为什么它有这些特殊选项。

《Learning UNIX》这本书是在介绍UNIX系统，让你从对UNIX毫无经验成长为会基本操作的用户。《UNIX in a Nutshell》这本书则是广泛地介绍UNIX工具包，对于使用时机与特定工具用法的介绍很少。我们的目的就在弥补这两本书之间的鸿沟：如何灵活运用这些UNIX提供的工具包，让工作更顺畅，更有效率，也更从容（我们的期望）。

如何结合所有工具，完成工作

编写Shell脚本时，其实会是“整体的功能比各部分加起来的总和还强大”。Shell的使用就像整合个别工具的黏着剂，让你只要花点心思，就能得到惊人的效果。

标准工具几个常见的扩展

如果你已经是GNU/Linux或BSD系统的用户，很可能你的工具还有其他额外的、好用的功能或选项。这部分我们也会介绍。

不可或缺的非标准工具

有些程序，在大部分传统的UNIX系统里并非“标准的”，但我们又不能没有它。我们会在适当的地方介绍它们，也会提供使用时机的相关信息。

对长期使用UNIX的开发人员与管理者来说，软件工具的设计原则一直没有什么改变。因此，推广的书籍虽然还算堪用，但已经20年未更新了，甚至更久！UNIX系统在这些书写成之后，有了许多变动。因此，我们觉得是更新这些想法的时候了，我们利用这些工具的现行版本、在现行系统下展示范例。下面是我们将要强调的部分：

- 所有的呈现是以POSIX为基础。POSIX为一系列描述可移植操作系统环境的标准正式名称的缩写。POSIX标准是开发人员的挑战，他们必须兼顾程序与Shell脚本在不同厂商所提供的各种平台上的可移植性。我们将在最新的POSIX标准下展现Shell语言、各个工具程序及其选项。

  该标准的官方名称为IEEE Std. 1003.1-2001（注3），它包括数个可选用的部分，最重要的一个就是X/Open System Interface（XSI）规格。这些功能文件详细描述了UNIX系统长久以来的行为模式。我们会介绍现行标准与早期1992标准间的差异，也会提供与XSI相关的特点。要了解UNIX相关标准，*http://www.UNIX.org/*（注4）是一个很好的起点。

  Single UNIX Specification的官方网站为*http://www.UNIX.org/version3/*。该标准可在线访问，不过得先到*http://www.UNIX.org/version3/online.html*注册。

  有时，该标准会将特殊行为模式保留为未定义（unspecified）。这是为了能让厂商以扩展的方式支持该行为，例如：额外的功能与标准本身未做成文件的部分。

- 除了告诉你如何执行特定程序外，我们还会强调这些程序存在的原因，及它们能解决什么问题。了解为什么会有这样的程序，有助于你进一步了解它的使用时机与方式。

- 大部分的程序都提供了相当多的选项组合。但通常只有一小部分是日常工作用得上的。这类程序，我们会让你知道它的哪些选项较方便好用。事实上，我们无法遍及每个程序的所有选项，未提及的部分，你可以通过程序的使用手册或其他参考书籍，例如《UNIX in a Nutshell》（O'Reilly）与《Linux in a NutShell》（O'Reilly），来了解它。

在你看完这本书之后，你不仅能了解UNIX工具集，还能吸收到UNIX的中心思想与软件工具设计的原则。

注3：该标准的2004版在本书内文底定后才发表。对学习Shell脚本而言，2001与2004间的差异并不重要。

注4：有关IEEE Std. 1003.1-2001的常见技术性问答（FAQ）文件，你或许可以在*http://www.opengroup.org/austin/papers/posix_faq.html*找到。与标准有关的后台知识，则在*http://www.opengroup.org/austin/papers/backgrounder.html*中。

你应该先有什么基础

我们认为读者应该了解以下背景知识：

- 如何登录 UNIX 系统
- 如何在命令行上执行程序
- 如何做一个简单的命令管道，与使用简单的输出 / 入重定向，例如 < 与 >
- 如何以 & 将程序放到后台执行
- 如何建立与编辑文件
- 如何使用 chmod，将脚本设为可执行权限

再者，如果你想试着操作本书范例，在你的终端机（或终端机模拟器）下达命令时，我们建议你使用 POSIX 兼容的 Shell，例如 ksh93 的最新版本，或 bash 的近期版本。请特别注意：商用 UNIX 系统上的 /bin/sh 可能并非完全兼容于 POSIX。

ksh93、bash 与 zsh 的下载网址请见第 14 章。

各章介绍

我们建议你依序阅读本书，因为每个章节都与前面章节息息相关。我们在此逐章介绍如下。

第 1 章，背景知识

此处提供简短的 UNIX 历史沿革。特别是贝尔实验室的运算环境，也就是 UNIX 开发的地方，激发了许多软件工具设计上的哲学。该章还会介绍这些原则，并在本书中贯彻执行。

第 2 章，入门

该章从编译语言与脚本语言间的取舍开始讨论。之后再介绍两个相当简单但很实用的 Shell 脚本程序。涵盖范围包括了命令、选项、参数、Shell 变量、echo 与 printf 的输出、基本输入 / 输出重定向、命令查找、从脚本里访问参数以及执行跟踪。最后则以国际化与本地化结束；这是在今日“地球村”环境下渐受重视的议题。

第 3 章，查找与替换

这里会介绍以正则表达式进行文字查找（或比对）。我们还会说明修改与提取文字的操作。这些都是最基本的 Shell 脚本编写的操作。

第4章，文本处理工具

该章介绍的是一些文字处理的软件工具，这些在Shell脚本编写时，都会一再地使用。其中最重要的两个就是sort与uniq，在重组与降低数据量上，它们扮演很重要的角色。本章还会带你看看如何重新编排段落、计算文字单位、显示文件以及取出文件的前几行、后几行数据。

第5章，管道的神奇魔力

该章以几个小型脚本为例，展示结合简单的UNIX工具程序能够产生更强大、更灵活的工具。本章的内容采取cookbook（问题描述与解决方案）的形式，它们共同的部分在于所有的解决方案都组合自线性的管道（pipelines）。

第6章，变量、判断、重复动作

这章介绍Shell语言里不可或缺的部分。包含了Shell变量与算法、退出状态的重要概念、如何判断，以及Shell循环的处理。最后以Shell的函数作结束。

第7章，输入/输出、文件与命令执行

该章为Shell描述的另一章，也是结尾，重点放在输入/输出、Shell所执行的各种替换、加引号、命令行执行顺序，以及Shell内置命令上。

第8章，产生脚本

我们在这里会示范如何结合UNIX的工具以处理更复杂的文本处理工作。本章的程序比第5章的还大，但仍是几分钟便能消化掉。甚至它们所完成的工作，如果使用传统的程序语言，例如C、C++或Java™来做，会很困难。

第9章，awk的惊人表现

该章介绍的是awk语言必备的组成部分。awk是一套功能强大且自给自足的语言。而awk程序更可用来与其他软件工具箱里的其他程序相结合，以执行简单的数据提取、处理与格式编排工作。

第10章，文件处理

该章介绍了处理文件的几个主要工具。包括列出文件、产生临时文件，以及利用指定标准寻找文件的find命令。另外还有两个与磁盘空间有关的重要命令，以及比较文件间异同的几个程序。

第11章，扩展实例：合并用户数据库

将所有东西串起来，解决既有趣又难易适中的挑战性工作。

第12章，拼写检查

该章利用拼写检查的问题，展现如何以数种方式解决它。这里展现了原始的UNIX Shell脚本管道以及两个小型的脚本：ispell与aspell命令，可自由下载，它们更适用于批处理的拼写检查工作。我们以awk写了一个大小适当的拼写检查程序，充分展现使用该语言的简单利落。

第 13 章，进程

该章将重点从文本处理的领域转到工作（job）与系统管理上。我们介绍了几个用于管理进程的必备工具，还有 `sleep` 命令，这在脚本需要等待某些事发生时很有用，另外则是其他一些用于延迟的标准工具，或修正日期时间命令的处理。最重要的是，该章也包括了 `trap` 命令，它可以让 Shell 脚本控制 UNIX 的信号。

第 14 章，Shell 可移植性议题与扩展

这里介绍的是一些更有用的扩展，可使用于 `ksh` 与 `bash` 之下，而非 POSIX。一般情况下，你都能安心地将这些扩展套用在你的脚本里。该章还会带你看几个“gotchas”，这是等待粗心大意的 Shell 脚本编写者跳入的陷阱。内容包括了在编写脚本时该注意的事项，还有在执行时可能出现的矛盾。除此之外，还包括有 `ksh` 与 `bash` 的下载与安装。该章最后会探讨各种不同的 Shell 实现间，Shell 初始化与终结的差异。

第 15 章，安全的 Shell 脚本：起点

该章会粗略介绍编写 Shell 脚本时的安全性议题。

附录 A，编写手册页

该附录讲的是如何编写手册页。这个必备的技术，在传统的 UNIX 的书籍里常被忽略。

附录 B，文件与文件系统

这个附录会介绍 UNIX 的字节数据流文件系统模型，并与较复杂的文件系统对照，然后解释其简洁的好处。

附录 C，重要的 UNIX 命令

该附录提供了许多 UNIX 命令的列表。建议你了解这些命令，它们可以增强你的能力。

参考书目

这是 UNIX 的 Shell 脚本编写的其他资料来源。

本书惯例

我们假设你已经知道，输入 Shell 命令时，最后会按下 Enter。Enter 在某些键盘上被表示为 Return。

提到 Ctrl-X 时，X 指的是任意字母，是在你按住 Ctrl（或 Ctl，或 Control）之后，接着按下的键。虽然我们这里用的是大写，不过你在按这个字母的时候无须按住 Shift 键。

其他特殊字符有换行符号（同于 Ctrl-J）、Backspace（同于 Ctrl-H）、Esc、Tab，与 Del（有时被标示为 Delete 或 Rubout）。

本书使用下列字体惯例：

斜体（Italic）

用在电子邮件地址、Internet URL、使用手册的引用。还用以表示参数，表示你在使用时，可以将它置换为你要的实际值，以及为范例提供说明性文字。

等宽字体（Constant Width）

提及UNIX文件名、扩展与内置命令、命令选项时，我们就会用到这个字体。除此之外，变量名称、Shell关键字、选项、函数、文件名结尾、范例中显示文件内容或命令的输出，以及当命令行或输入范例存在于正规文字中时，我们也会以等宽字体表示。简而言之，任何与计算机使用相关的，我们都用这个字体。

粗体等宽字体（**Constant Width Bold**）

这种字体用以区分比对文字中的正则表达式与Shell通配字符样式。在范例中，显示用户与Shell间的互动，我们也会使用这个字体，所有用户应键入的，我们都以粗体等宽字体显示，像这样：

```
$ pwd                              用户输入这个
/home/tolstoy/novels/w+p           系统显示这个
$
```

斜体等宽字体（*Constant width italic*）

这个字体是用在范例与内文中，需置换为正确值的命令行参数上。例如：

```
$ cd directory
```

注意：表示诀窍、建议或一般注意事项。

警告：表示警告与提醒。

参照UNIX User's Manual时，我们会使用标准形式：*name*(N)，*name*为命令名称，而N为section编号（通常是1），也就是寻找信息的地方。例如*grep*(1)即grep的Section 1的手册页。参考文件我们使用man page，或直接简写为manpage。

UNIX系统调用与C程序库，我们都这么写：open()、printf()。你可以使用man命令，查看这两者的manpage：

```
$ man open          查看open(2)的manpage
$ man printf        查看printf(3)的manpage
```

当我们要介绍一个程序时，就使用下面的方式，显示在正文附近，说明该工具程序与它的重要选项、语法与用途。

范例

语法

whizprog [*options* ...] [*arguments* ...]

说明如何执行这里指出的whizprog命令。

用途

说明此程序存在的目的。

主要选项

列出此程序每天要用到的重要选项。

行为模式

概括程序所做的事。

警告

所有要注意的事全在这。

程序码范例

本书并非只是解释命令与程序的功能，还提供了完整的Shell命令与程序的设计范例，便于用户或程序员直接应用于日常工作上。我们非常鼓励你修改与强化这些范例。

本书所提供的程序代码，都公开在GNU General Public License（GPL）条款下，允许程序复制、再利用与编修。请参阅范例包括的COPYING文件，了解许可权的实际条款。

代码可从原书网站获得：*http://www.oreilly.com/catalog/shellsrptg/index.html*。

我们会感谢你在使用程序码范例时注明出处，但这并非必要。这通常包括标题、作者、出版商与ISBN。例如：《Classic Shell Scripting》，作者Arnold Robbins与Nelson H.F. Beebe。版权所有2005 O'Reilly Media, Inc.，ISBN为0-596-00595-4。

Windows系统下的UNIX工具程序

很多的程序员初次接触UNIX系统后，再回到个人计算机的世界，常会希望也有一个像UNIX那样好用的环境（特别是在面对难以处理的MS-DOS命令行时），所以会有UNIX Shell式的界面出现在一些小型计算机的操作系统上，也不是什么奇怪的事。

近几年，我们不只看过Shell仿制品，还看过整个完整的UNIX环境的仿制品。其中两个就是使用bash与ksh93，其他则是提供自有的Shell重新实现（reimplementation）。本节将依次（以字母顺序）予以介绍，并附上联络方式与Internet下载信息。

Cygwin

Cygnus Consulting（现为 Red Hat）建立了 cygwin 环境。首先出现的是提供 UNIX 系统调用模拟器的共享程序库 cgywin.dll，该公司释出了许多 GNU 工具程序，供各种不同的 Microsoft Windows 版本使用。模拟器还包括了 Berkeley socket API 的 TCP/IP 网络。在 Windows/NT、Windows 2000 与 Windows XP 下，功能性最佳，不过在 Windows 95/98/ME 下也是可以运行的。

cygwin 环境使用自己的 Shell、自己的 C 编译器 GCC，以及搭配其 UNIX 工具集里的 GNU 工具程序。设计精良的 mount 命令提供了将 Windows C:\path 的标记方式对应到 UNIX 文件名。

http://www.cygwin.com/ 是你了解 cygwin 项目的起点。第一步就是下载它的安装程序，执行时，你可以选择要安装哪些额外包。整个安装过程都是在 Internet 上进行，没有官方的 cygwin CD，至少项目的维护人员并没有提供。

DJGPP

DJGPP 程序组提供了 MS-DOS 环境下所使用的 32 位 GNU 工具程序。以下内容摘自其网页：

> DJGPP 为执行 MS-DOS 的 Intel 80386（及更高级的）个人计算机提供了完整的 32 位 C/C++ 开发系统。其中包含许多 GNU 开发工具程序。这些开发的工具必须在 80386 或更高级的计算机上执行产生程序。大部分情况下，其所产生的程序可做商业用途而无须授权或版税。

其名称一开始是来自 D.J. Delorie，D.J. Delorie 将 GNU C++ 编译器 g++ 移植到 MS-DOS，而 g++ 一开始的名称为 GPP。之后逐渐茁壮成长，成为 MS-DOS 下完整的 UNIX 环境不可或缺的要素，并带有 GNU 工具，以 bash 作为其 Shell。不同于 cygwin 或 UWIN 的是：不需要使用任何的 Windows 版本，只要有完整的 32 位处理器与 MS-DOS 即可（当然，你也可以在 Windows 的 MS-DOS 窗口下使用 DJGPP）。官方网站为 *http://www.delorie.com/djgpp/*。

MKS Toolkit

个人计算机世界里的 UNIX 环境有许多都是以 Mortice Kern Systems 的 MKS Toolkit 建立的：

MKS Canada – Corporate Headquarters
410 Albert Street
Waterloo, ON
Canada N2L 3V3
1-519-884-2251
1-519-884-8861 (FAX)
1-800-265-2797 (Sales)
http://www.mks.com/

MKS Toolkit版本非常多，根据你的开发环境和开发人员的数量来决定要用哪一个版本。其中包含了与POSIX兼容的Shell，拥有1988 Korn Shell里的所有功能，超过300种的工具组，例如awk、perl、vi、make等。MKS程序库支持超过1500个UNIX API，使它更为完整，更易于应用在Windows环境下。

AT&T UWIN

UWIN包是David Korn与同事为了在Microsoft Windows下使用UNIX环境而产生的项目。其架构类似先前讨论的cygwin。共享程序库posix.dll提供了UNIX系统调用API的模拟器。其系统调用模拟器相当完整。其中一个有趣的创新就是将Windows的登录改为可在/reg文件系统下访问的方式。UWIN环境依赖原始的Microsoft Visual C/C++编译器，不过仍可自行下载GNU开发工具用于UWIN下。

*http://www.research.att.com/sw/tools/uwin/*是此项目的网页，说明可供下载的有哪些，并附上二进制文件的下载链接，还有与UWIN的使用许可权相关的信息。除此之外，另有UWIN的各类报告、额外的好用软件及其他类似包的链接。

UWIN包最大的优势为：它的Shell是一个真正的ksh93，因此在与UNIX的ksh93版本的兼容性上不会有问题。

联系我们

请将关于本书的意见和问题通过以下地址提供给出版商：

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

100035 北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室

奥莱利技术咨询（北京）有限公司

O'Reily公司是世界性的计算机信息出版公司。我们永远乐意听到读者对出版物的意见，包括如何让本书可以更好的建议，指正本书的错误或是读者建议本书往后改版时应该再加入的其他主题。以下是英文原书的联络数据：

http://oreilly.com/catalog/9780596005955/index.html

如果想要发表关于本书的评论和技术问题，请发邮件至：

bookquestions@oreilly.com

info@mail.oreilly.com.cn

关于图书、会议、资源中心和 O'Reilly 网络的更多信息，请查看我们的站点：

http://www.oreilly.com

http://www.oreilly.com.cn

致谢

我们俩要感谢对方的付出。虽然我们未曾见面，但协同作业的工作一直进行得很好。除了对彼此献上最热忱的感激，还要谢谢我们挚爱的妻子，谢谢她们在这段期间的贡献、耐心、爱，以及在此书编写期间的支持。

bash 的维护人员 Chet Ramey 回答了许许多多与 POSIX Shell 相关的细微问题。AT&T Research 的 Glenn Fowler 与 David Korn，与 GNU Project 的 Jim Meyering，也提供不少解答。我们以字母顺序排列下面要致谢的人：Keith Bostic、George Coulouris、Mary Ann Horton、Bill Joy、Rob Pike、Hugh Redelmeier、Dennis Ritchie，他们为 UNIX 的历史问题解惑。由于 O'Reilly Media 的 Nat Torkington、Allison Randall、Tatiana Diaz 的指导，让本书由概念逐渐成型。感激 O'Reilly 的 Robert Romano 将我们原始的 ASCII 草图与 pic 略图作成图片。也谢谢 Angela Howard 为本书制作全面性索引，造福读者。

以下仍以字母顺序：谢谢 Geoff Collyer、Robert Day、Leroy Eide、John Halleck、Mark Lucking 与 Henry Spencer 为本书初稿再作技术上的检阅，及 Sean Burke 审阅第 2 版草稿。谢谢他们提供无价且非常有帮助的回馈。

UNIX Guru 的 UNIX Guru（精神导师）Henry Spencer，谢谢你为本书作的序。

感谢University of Utah的Electrical and Computer Engineering、Mathematics，与Physics等系所，以及Center for High-Performance Computing，提供UNIX系统供访问；还有IBM与Hewlett-Packard提供访客访问，让我们取得编写本书所需的软件以供检测，谢谢，谢谢他们。

——Arnold Robbins

——Nelson H.F. Beebe