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