
对于任何希望相当精通系统管理的人来说，shell 脚本的工作知识是必不可少的，即使他们没有预料到必须实际编写脚本。 考虑当 Linux 机器启动时，它会执行 /etc/rc.d 中的 shell 脚本来恢复系统配置并设置服务。 详细了解这些启动脚本对于分析系统行为以及可能对其进行修改非常重要。

编写脚本的技巧并不难掌握，因为脚本可以构建成一小部分，并且只有一小部分特定于 shell 的运算符和选项 [1] 需要学习。 语法很简单——甚至是朴素的——类似于在命令行中调用实用程序并将其链接在一起，并且只有几个 “规则” 来管理它们的使用。 大多数短脚本第一次就可以正常工作，即使是较长的脚本也很容易调试。

> 在个人计算的早期，BASIC 语言启用
> 任何具有相当计算机能力的人都能在早期编写程序
> 一代微机。 几十年后，Bash 脚本
> 语言使任何具有 Linux 基本知识的人或
> UNIX 在现代机器上做同样的事情。
>
> 我们现在拥有令人惊叹的微型单板计算机
> 功能，例如 Raspberry Pi。
> Bash 脚本提供了一种探索这些功能的方法
> 迷人的设备。

---

shell 脚本是对复杂应用程序进行原型设计的一种快速而简单的方法。 让功能的有限子集在脚本中工作通常是项目开发中有用的第一阶段。 通过这种方式，可以测试和修改应用程序的结构，并在继续使用 C、C++、Java、Perl 或 Python 进行最终编码之前发现主要缺陷。

Shell 脚本回到了经典的 UNIX 哲学，将复杂的项目分解为更简单的子任务，将组件和实用程序链接在一起。 许多人认为这是一种更好的解决问题的方法，或者至少是更美观的解决问题的方法，而不是使用新一代的高性能一体化语言之一，比如 Perl，它试图成为所有人的一切，但在 强迫你改变你的思维过程以适应工具的成本。

根据 Herbert Mayer 的说法，“一种有用的语言需要数组、指针和用于构建数据结构的通用机制。” 根据这些标准，shell 脚本在某种程度上还不够 “有用”。 或者，也许不是。 . . .

```
When not to use shell scripts

Resource-intensive tasks, especially where speed is a factor (sorting, hashing,
recursion [2] ...)

Procedures involving heavy-duty math operations, especially floating point
arithmetic, arbitrary precision calculations, or complex numbers (use C++ or
FORTRAN instead)

Cross-platform portability required (use C or Java instead)
```

```
Complex applications, where structured programming is a necessity (type-checking
of variables, function prototypes, etc.)

Mission-critical applications upon which you are betting the future of the company

Situations where security is important, where you need to guarantee the integrity
of your system and protect against intrusion, cracking, and vandalism

Project consists of subcomponents with interlocking dependencies

Extensive file operations required (Bash is limited to serial file access, and
that only in a particularly clumsy and inefficient line-by-line fashion.)

Need native support for multi-dimensional arrays

Need data structures, such as linked lists or trees

Need to generate / manipulate graphics or GUIs

Need direct access to system hardware or external peripherals

Need port or socket I/O

Need to use libraries or interface with legacy code

Proprietary, closed-source applications (Shell scripts put the source code right
out in the open for all the world to see.)

If any of the above applies, consider a more powerful scripting language --
perhaps Perl, Tcl, Python, Ruby -- or possibly a compiled language such as C, C++,
or Java. Even then, prototyping the application as a shell script might still be a
useful development step.
```

我们将使用 Bash，它是"Bourne-Again shell"的首字母缩写词 [3]，也是 Stephen Bourne 现在经典的 Bourne shell 的双关语。 Bash 已成为大多数 UNIX 风格的 shell 脚本的事实上的标准。 本书涵盖的大多数原则同样适用于使用其他 shell 编写脚本，例如 Korn Shell，Bash 从中派生了一些特性，[4] 和 C Shell 及其变体。 （请注意，由于某些固有的问题，不建议使用 C Shell 编程，正如 Tom Christiansen 在 1993 年 10 月的 Usenet 帖子中指出的那样。）

下面是关于 shell 脚本的教程。 它严重依赖示例来说明 shell 的各种特性。 示例脚本有效——它们已经过尽可能多的测试——其中一些甚至在现实生活中很有用。 读者可以玩弄源存档中示例的实际工作代码（scriptname.sh 或 scriptname.bash），[5] 给它们执行权限（chmod u+rx scriptname），然后运行它们看看会发生什么。 如果源存档不可用，则从 HTML 或 pdf 呈现的版本中剪切和粘贴。 请注意，此处介绍的某些脚本在解释之前介绍了功能，这可能需要读者暂时跳过以获得启发。

除非另有说明，否则本书的作者编写了以下示例脚本。

Notes

[1] 这些被称为内置功能，是外壳内部的功能。

---

[2] 尽管在 shell 脚本中递归是可能的，但它往往很慢，而且它的实现通常是一个丑陋的组合。

---

[3] 首字母缩略词是通过将单词的首字母粘贴在一起形成一个令人吃惊的短语而形成的 ersatz 单词。 这种道德败坏和有害的做法应该受到适当的严厉惩罚。 公开鞭笞不言自明。

---

[4] ksh88 的许多功能，甚至更新后的 ksh93 中的一些功能都已合并到 Bash 中。

---

[5] 按照惯例，与 Bourne shell 兼容的用户编写的 shell 脚本通常采用带有 .sh 扩展名的名称。 系统脚本，例如 /etc/rc.d 中的那些，不一定符合这个命名法。