# Shell 脚本：绝对基础知识

约瑟夫·哈利特

2023 年 1 月 12 日

## 这到底是怎么回事？

多年来，我编写了很多代码：作为工程师，为我自己的项目编写了汇编、C 和 Java Commonlisp Haskell构建编译器PostScript绘制真正高效的图表LATEX出版书籍……还有其他十几个东西

我用哪种语言编写的代码最多？

我使用哪种语言来解决大多数任务？

我最不喜欢哪种语言？

# 外壳脚本！

通常我们在命令行上为终端输入命令... ▶但我们可以自动化它们并将它们粘贴到脚本中

## 任何你必须做不止一次的事情.....

为它写一个脚本！ ▶节省

大量时间

▶通常比编写完整的程序更容易

# 例如···

```
#!  /bin/sh
GREP=grep
if [ $(uname) =    OpenBSD    ];然后 # 在
       OpenBSD 上使用 GNU Grep
       GREP=ggrep
是

${GREP} -Pi    ^${1}$    /usr/share/dict/words
```

有时我在 Wordle 上作弊：

▶我想知道一个与正则表达式完全匹配的单词

▶我可以在以下位置搜索系统词典文件：
/usr/share/dist/words ▶ grep

可以进行搜索，但我需要

在非默认系统上显式指定GNU Grep

结词    st[^aeo]pid    - 愚蠢

# 或者例如……

```
#!  /usr/bin/env bash if [ $1 =
  应该   -a $2 =   也   -a $3 =   运行   ];然后
   班次 3 口香
   糖确认“运行‘doas $*’？”        doas $* elif [ $1 =   应该
-a $2 =   也   -a $3 =   删除   ];然后口香糖确认“删除‘$4’？”        doas rm -fr   ${4}   else
   2>&1 printf                    你应该仔细阅读你   2>&1 printf   粘贴的
命令
   \n   fi
```

有时,当我升级计算机时,它会告诉我删除一些文

件或运行一些命令： 您还应该运行 rcctl restart pf 复制并粘贴精确的文本很痛苦... ▶我可以复制整行

并运行它吗？

（当然我可以……我应该吗？）

# 或者再举一个例子……

```
#!  /usr/bin/env bash # 修
复 kitty /usr/
local/opt/bin/fix-kitty

# 更新源 cd /usr/src
       cvs -q up -Pd -A cd /usr/ports
cvs -q up -Pd -A cd /usr/xenocara        cvs
-q up -Pd -A
```

升级计算机后,我需要运行几个标准命令。

- 我永远记不起他们
- 将它们分批进行!

# 那么这到底是怎么回事呢？

Shellscripting 是关于自动化所有那些乏味的小工作

▶拜占庭语法（基于 shell 命令） ▶调试起来很糟糕▶

需要神奇的知识▶可能是您

学到的最有用的东西

# 幸运的是我们有帮助

Shell 脚本有点神奇，并且有很多陷阱..... https://www.shellcheck.net发现 shell 脚本中不可移植/危险内容的绝佳工具▶

提供命令行工具

▶在您编写的所有内容上运行它▶ shellcheck 很棒shellcheck

`command -v knotwords` 在 /
home/joseph/.local/bin/knotwords 第 2 行：
GREP=grep ^--^ SC2209（警告）：使用 var=$ （命令）分配输出（或引用分
配字符串）。

在 /home/joseph/.local/bin/knotwords 第 3 行： if [ $(uname)
= OpenBSD ];然后
^------^ SC2046（警告）：引用此内容以防止分词。

有关更多信息： https://
www.shellcheck.net/wiki/SC2046 -- 引用此内容以防止分词... https://www.shellcheck.net/wiki/SC2209 -- 使用
var=$(command)分配输出...

# 那么怎么写呢？

文件以shebang #!开头然后是脚本解释器的路径加上任何参数:对于便携式 POSIX shellscripts #! /bin/sh/对于不太可移植的 BASH 脚本#! /usr/bin/

env bash 然后

- chmod +x my-script.sh
- ./my-script.sh

文件的其余部分将由您指定的解释器运行

- 或 sh my-script.sh 如果您不想/无法将其标记为可执行。

（嘿,这也是 Python 脚本启动 #!/usr/bin/env python3 的原因）

## 为什么是环境?

等等,你可能会说,我知道 bash 总是在 /bin/bash 中……我可以把它作为我的解释器路径吗?

## 对,但是…

一开始 /bin 只保留给系统程序使用

▶和 /usr/bin 用于管理员安装的程序▶和 /usr/local/bin 用于本地安装的程序▶和 /opt/bin 用于可选安装的程序▶和 /opt/local/bin 用于可选本地安装的程序▶和 ~/. local/bin 用于用户程序▶ …哦,有时它们甚至安装在不同的磁盘上!

这有点疯狂。

▶因此， Linux 系统必须说,我们将把所有内容都放在 /bin 中,并停止使用多个分区

▶但有人说不,应该是/usr/bin,有人说/Applications/,还有人把它们塞进去 /usr/bin 但将它们符号链接到 /bin

▶在某些系统上,用户厌倦了过时的系统 bash,并编译了自己的系统并将其安装在 ~/.local/bin 中……

▶ …曾经尝试过使用Python venv吗?

# 环境

环境(1) 通用命令手册 环境(1)

姓名

env – 设置和打印环境

概要 env [-

i] [名称=值 ...] [实用程序 [参数 ...

描述 env 在修改

命令行上指定的环境后执行实用程序。选项 name=value 指定一个环境变量 name，其值为 value。

env 所做的是查找 PATH 并尝试找到指定的程序并运行它。

# …小路？

有一个环境变量叫PATH，它告诉系统所有程序在哪里

是：

▶以冒号分隔的路径列表

如果你想改变它，你可以添加一行类似于你的 shell 的配置

导出 PATH= ${PATH}:/extra/directory/to/search

你的 shell 配置可能在 ~/.profile 中，但它经常变化……检查 ${SHELL} 的手册页

另外一些 shell 有不同的语法（例如鱼）……

```
$ tr : $ \n <<< $PATH /
home/joseph/.local/share/python/bin /bin /usr/
bin /
sbin /usr/
sbin /
usr/X11R6/
bin /usr/local /bin /
usr/local/sbin /
home/joseph/.local/
bin /usr/local/opt/bin /usr/
games /usr/local/games /
usr/local/
jdk-17/bin /home/
joseph/ .local/share/go/bin
```

# 基本语法

Shell 脚本是通过将命令链接在一起编写的

A; B运行 A 然后运行 B

一个 | B运行 A 并将其输出作为 B 的输入

A B运行 A,如果成功则运行 B

一个 || B运行 A,如果不成功则运行 B

## 它如何知道是否成功?

程序返回一个 1 字节的退出值（例如 C main 以 return 0 结束；）

▶每个命令运行后,这都会存储到变量 ${?} 中。 ▶ 0 表示成功（通常） ▶ >0 表示失败（通常）

然后可以将其与 test 等命令一起使用：

```
do_long_running_command
测试 $? -eq 0        printf    命令成功\n
```

或者稍微短一点：

```
do_long_running_command
[ $? -eq 0 ]        printf    命令成功\n
```

# 包起来

这就是 shell 脚本编写的基础知识， ▶包含#!

▶始终使用 env ▶ $?包
含退出代码

下次将介绍
shellscript 的控制流和更高级的 shell 脚本。

奖金谜题为什么会这样？
[ $? -eq 0 ] # 有效 [$? -eq 0]
# 不起作用

# 不同的贝壳

（只要使用 bash 除非你关心极端的可移植性，在这种情况下使用 POSIX sh）

## 典型的贝壳

sh POSIX shell

bash Bourne Again shell（Linux 上默认） zsh Z Shell（Mac 上默认），与 bash 类似，但具有更多功能ksh Korne shell（BSD 上默认）

## 其他贝壳

dash简化了更快的 bash，用于在 Linux 上启动

Busybox sh简化了嵌入式系统上的 bash

## 奇怪的贝壳

Fish更可用的 shell（但不同的不兼容语法） elvish更好的脚本语法（但与 POSIX 不兼容）

nushell更好的输出（但不兼容，而且很奇怪）