外壳脚本2

约瑟夫·哈利特

2023 年 1 月 12 日

上次

我们引入了 shell 脚本作为自动化工具

- ▶给出了语法的基本概述▶提到了 env 和 shellcheck

这次

- ▶更多语法和控制流程▶变量和技术

和以前一样,我将尝试保持 POSIX shell 并标记出 Bashism 的位置.....

- ▶但了解一些 Bash 主义是有用的

变量

所有程序都有变量……Shell 语言也不例外：

创建变量：

GREETING=“世界,你好！”

（= 周围没有空格）

使用变量

回声“${问候}”

如果您希望变量存在于作为环境变量启动的程序中： export GREETING

摆脱变量

取消设置问候语

出色地…

shell 语言中的变量往往更像宏变量。 ▶使用未定义的选项不会受到任何惩罚。

```
NAME=  Joe
unset NAME
echo  你好，  ${NAME}
```

你好，

如果这让您烦恼：

```
set -o nounset echo
  ${NAME:?变量 1 传递给程序}
```

（除了 :? 之外，还有很多shell 参数扩展技巧，它们可以执行搜索和替换、字符串长度和各种魔法……）

标准变量

${0}脚本名称${1}、${2}、${3}...传递给脚本的参数

${#}传递给脚本的参数数量${@} 和 ${*}所有参数

控制流

可以使用带有通配符的if语句和for循环： # 或 [-x myscript.sh]; # 或者 -x

myscript.sh ;如果使用 Bash if test
-x myscript.sh;然后./myscript.sh

是

对于 *.py 中的文件； do python
${file} 完成

其他循环

好吧……好吧,你只有 for 真的……但你可以用它做其他事情:

对于 1 2 3 4 5 中的 n;做 echo
-n ${n}
完毕

1 2 3 4 5

序列5

1 2 3 4 5

对于 $(seq 5) 中的 n;执行 echo -n
${n} 完成

1 2 3 4 5

序列-s,

1,2,3,4,5

```
# IFS = 字段分隔符
IFS= ,
for n in $(seq -s, 5);执行 echo -n ${n}
完成
```

1 2 3 4 5

还有案例陈述！

```
3 # 删除 ${SHELL} 中最后一个 / 之前的所有内容 case    ${SHELL       */}    in
    bash) echo    I    m using
          bash!    ;; zsh) echo    哦,太喜欢 zsh 用户
          了！     ;;鱼)回声“有鱼腥味！”;; *) echo“哦还有别的
          事！”;;
```

埃萨克

基本名称和目录名称

在前面的示例中,我使用了“${VAR */}”技巧来删除最后一个 /... 之前的所有内容。
这给你整齐地提供了文件的名称.....但我每次使用
它时都必须查找它。
相反,我们可以使用 $(basename ${shell}) 来获取相同的信息。 echo $

{SHELL} echo
${SHELL */}
echo $(basename ${SHELL})
echo $(dirname ${SHELL})

您甚至可以使用它来删除文件扩展名： for f in *.jpg;转换 ${f} $

(basename $
{f} .jpg).png 完成

管道

作为 shell 脚本编写的一部分,从其他命令链中构建命令通常很有用。
例如,我可以使用 ps 列出我的计算机上的所有进程,并使用 grep 进行搜索。 ▶ Firefox使用了多少个进程?

ps -A | grep -i 火狐浏览器

43172 ?? SpU 0:10.69 /usr/local/bin/firefox 59551 ?? Sp 0:00.06 /usr/local/lib/
firefox/firefox -contentproc -appDir 7023 ?? SpU 0:06.10 /usr/local/lib/firefox/firefox -contentproc {a032331 59478 ?? SpU 0:00.21 /
usr/local/lib/firefox/firefox -contentproc {3cd651d 47320 ?? SpU 0:00.60 /usr/local/lib/firefox/firefox -contentproc {50d5261
26734 ?? SpU 0:00.18 /usr/local/lib/firefox/firefox -contentproc {68aa722 308 ?? SpU 0:00.16 /usr/local/lib/firefox/firefox -contentproc
{bd6ff5f 42479 ?? SpU 0:00.14 /usr/local/lib/firefox/firefox -contentproc {d874750 45572 ?? Rp/2 0:00.00 grep -i

火狐浏览器

信息太多了！

让我们使用 awk 命令将其剪切为仅第一列和第五列！ ps -A | grep -i 火狐| awk

{打印 $1, $5}

43172 /usr/local/bin/firefox 59551 /
usr/local/lib/firefox/firefox 7023 /usr/local/
lib/firefox/firefox 59478 /usr/local/lib/
firefox/firefox 47320 /usr/local/lib/火狐/火
狐 26734 /usr/local/lib/火狐/火狐 308 /usr/
local/lib/火狐/火狐 42479 /usr/local/lib/火
狐/火狐 5634 grep

为什么 grep 在那里？

哦，是的……当我们搜索firefox时，我们在命令行中使用firefox创建一个新进程。
让我们删除最后一行 ps -A

| grep -i 火狐| awk {print $1, $5} |头-n -1

43172 /usr/local/bin/firefox 59551 /
usr/local/lib/firefox/firefox 7023 /usr/local/
lib/firefox/firefox 59478 /usr/local/lib/
firefox/firefox 47320 /usr/local/lib/火狐/火
狐 26734 /usr/local/lib/火狐/火狐 308 /usr/
local/lib/火狐/火狐 42479 /usr/local/lib/火狐/
火狐

实际上我只想计算进程数量

```
ps -A | grep -i 火狐| awk    {print $1, $5}    |头 -n -1 |厕所-l
8
```

其他管道技术

▶ |管道将标准输出复制到标准输入... ▶ > 管道将标准输出复制到命名文件...（例如 ps -A >processes.txt,另请参阅 tee 命令） ▶ >> 管道将标准输出附加到命名文件… ▶ < 管道将文件读入标准输入… （例如 grep firefox <processes.txt） ▶ <<< 管道接受一个字符串并将其放在标准输入中输入▶如果您知道流的文件描述符,您甚至可以复制和合并流（例如,将 2>&1 附加到命令将运行该命令,并将标准错误合并到标准输出中）

结束语继续

执行 shell 脚本！

我们涵盖的内容

▶变量扩展▶通用控制流语句

▶不同的管道技巧