



首页
最新文章
经典回顾
开发
设计
IT技术
职场
业界
极客
创业
访谈
在国外

- 导航条 -

[伯乐在线](#) > [首页](#) > [所有文章](#) > [IT技术](#) > UNIX 高手的 10 个习惯

UNIX 高手的 10 个习惯

2013/06/30 · [IT技术](#) · [3 评论](#) · [Linux](#), [Unix](#)

分享到: 73

[高性能产品的必由之路—性能测试工具
如何实现“新手引导”效果](#)[JAVA遇见HTML——Servlet篇
Echarts3.0入门基础与实战](#)原文出处: [IBM DeveloperWorks](#)

采用 10 个能够提高您的 UNIX 命令行效率的好习惯——并在此过程中摆脱不良的使用模式。本文循序渐进地指导您学习几项用于命令行操作的技术，这些技术非常好，但是通常被忽略。了解常见错误和克服它们的方法，以便您能够确切了解为何值得采用这些 UNIX 习惯。

引言

当您经常使用某个系统时，往往会陷入某种固定的使用模式。有时，您没有养成以尽可能最好的方式做事的习惯。有时，您的不良习惯甚至会导致出现混乱。纠正此类缺点的最佳方法之一，就是有意识地采用抵制这些坏习惯的好习惯。本文提出了 10 个值得采用的 UNIX 命令行习惯——帮助您克服许多常见使用怪癖，并在该过程中提高命令行工作效率的好习惯。下面列出了这 10 个好习惯，之后对进行了更详细的描述。

采用 10 个好习惯

要采用的十个好习惯为：

1. 在单个命令中创建目录树。
2. 更改路径；不要移动存档。
3. 将命令与控制操作符组合使用。
4. 谨慎引用变量。
5. 使用转义序列来管理较长的输入。
6. 在列表中对命令分组。
7. 在 find 之外使用 xargs。
8. 了解何时 grep 应该执行计数——何时应该绕过。
9. 匹配输出中的某些字段，而不只是对行进行匹配。
10. 停止对 cat 使用管道。

在单个命令中创建目录树

清单 1 演示了最常见的 UNIX 坏习惯之一：一次定义一个目录树。

清单 1. 坏习惯 1 的示例：单独定义每个目录树

```
1 ~ $ mkdir tmp
2 ~ $ cd tmp
3 ~/tmp $ mkdir a
4 ~/tmp $ cd a
5 ~/tmp/a $ mkdir b
6 ~/tmp/a $ cd b
7 ~/tmp/a/b/ $ mkdir c
8 ~/tmp/a/b/ $ cd c
9 ~/tmp/a/b/c $
```

使用 `mkdir` 的 `-p` 选项并在单个命令中创建所有父目录及其子目录要容易得多。但是即使对于知道此选项的管理员，他们在命令行上创建子目录时也仍然束缚于逐步创建每级子目录。花时间有意识地养成这个好习惯是值得的：

清单 2. 好习惯 1 的示例：使用一个命令来定义目录树

```
1 ~ $ mkdir -p tmp/a/b/c
```

您可以使用此选项来创建整个复杂的目录树（在脚本中使用是非常理想的），而不只是创建简单的层次结构。例如：

清单 3. 好习惯 1 的另一个示例：使用一个命令来定义复杂的目录树

```
1 ~ $ mkdir -p project/{lib/ext,bin,src,doc/{html,info,pdf},demo/stat/a}
```

过去，单独定义目录的唯一借口是您的 `mkdir` 实现不支持此选项，但是在大多数系统上不再是这样了。IBM、AIX、`mkdir`、GNU `mkdir` 和其他遵守单一 UNIX 规范 (Single UNIX Specification) 的系统现在都具有此选项。

对于仍然缺乏该功能的少数系统，您可以使用 `mkdirhier` 脚本，此脚本是执行相同功能的 `mkdir` 的包装：

```
1 ~ $ mkdirhier project/{lib/ext,bin,src,doc/{html,info,pdf},demo/stat/a}
```

更改路径；不要移动存档

另一个不良的使用模式是将 `.tar` 存档文件移动到某个目录，因为该目录恰好是您希望在其中提取 `.tar` 文件的目录。其实您根本不需要这样做。您可以随心所欲地将任何 `.tar` 存档文件解压缩到任何目录——这就是 `-c` 选项的用途。在解压缩某个存档文件时，使用 `-c` 选项来指定要在其中解压缩该文件的目录：

清单 4. 好习惯 2 的示例：使用选项 `-c` 来解压缩 `.tar` 存档文件

```
1 ~ $ tar xvf -C tmp/a/b/c newarc.tar.gz
```

相对于将存档文件移动到您希望在其中解压缩它的位置，切换到该目录，然后才解压缩它，养成使用 `-c` 的习惯则更加可取——当存档文件位于其他某个位置时尤其如此。

将命令与控制操作符组合使用

您可能已经知道，在大多数 Shell 中，您可以在单个命令行上通过在命令之间放置一个分号 (;) 来组合命令。该分号是 Shell 控制操作符，虽然它对于在单个命令行上将离散的命令串联起来很有用，但它并不适用于所有情况。例如，假设您使用分号来组合两个命令，其中第二个命令的正确执行完全依赖于第一个命令的成功完成。如果第一个命令未按您预期的那样退出，第二个命令仍然会运行——结果会导致失败。相反，应该使用更适当的控制操作符（本文将描述其中的部分操作符）。只要您的 Shell 支持它们，就值得养成使用它们的习惯。

仅当另一个命令返回零退出状态时才运行某个命令

使用 `&&` 控制操作符来组合两个命令，以便仅当第一个命令返回零退出状态时才运行第二个命令。换句话说，如果第一个命令运行成功，则第二个命令将运行。如果第一个命令失败，则第二个命令根本就不运行。例如：

清单 5. 好习惯 3 的示例：将命令与控制操作符组合使用

```
1 ~ $ cd tmp/a/b/c && tar xvf ~/archive.tar
```

在此例中，存档的内容将提取到 `~/tmp/a/b/c` 目录中，除非该目录不存在。如果该目录不存在，则 `tar` 命令不会运行，因此不会提取

任何内容。

仅当另一个命令返回非零退出状态时才运行某个命令

类似地，`||` 控制操作符分隔两个命令，并且仅当第一个命令返回非零退出状态时才运行第二个命令。换句话说，如果第一个命令成功，则第二个命令不会运行。如果第一个命令失败，则第二个命令才会运行。在测试某个给定目录是否存在时，通常使用此操作符，如果该目录不存在，则创建它：

清单 6. 好习惯 3 的另一个示例：将命令与控制操作符组合使用

```
1 ~ $ cd tmp/a/b/c || mkdir -p tmp/a/b/c
```

您还可以组合使用本部分中描述的控制操作符。每个操作符都影响最后的命令运行：

清单 7. 好习惯 3 的组合示例：将命令与控制操作符组合使用

```
1 ~ $ cd tmp/a/b/c || mkdir -p tmp/a/b/c && tar xvf -C tmp/a/b/c ~/archive.tar
```

谨慎引用变量

始终要谨慎使用 Shell 扩展和变量名称。一般最好将变量调用包括在双引号中，除非您有不这样做的足够理由。类似地，如果您直接在字母数字文本后面使用变量名称，则还要确保将该变量名称包括在方括号 (`[]`) 中，以使其与周围的文本区分开来。否则，Shell 将把尾随文本解释为变量名称的一部分——并且很可能返回一个空值。清单 8 提供了变量的各种引用和非引用及其影响的示例。

清单 8. 好习惯 4 的示例：引用（和非引用）变量

```
1 ~ $ ls tmp/
2 a b
3 ~ $ VAR="tmp/*"
4 ~ $ echo $VAR
5 tmp/a tmp/b
6 ~ $ echo "$VAR"
7 tmp/*
8 ~ $ echo $VARa
9
10 ~ $ echo "$VARa"
11
12 ~ $ echo "${VAR}a"
13 tmp/*a
14 ~ $ echo ${VAR}a
15 tmp/a
16 ~ $
```

使用转义序列来管理较长的输入

您或许看到过使用反斜杠 (`$` 来将较长的行延续到下一行的代码示例，并且您知道大多数 Shell 都将您通过反斜杠联接的后续行上键入的内容视为单个长行。然而，您可能没有在命令行中像通常那样利用此功能。如果您的终端无法正确处理多行回绕，或者您的命令行比通常小（例如在提示符下有长路经的时候），反斜杠就特别有用。反斜杠对于了解键入的长输入行的含义也非常有用，如以下示例所示：

清单 9. 好习惯 5 的示例：将反斜杠用于长输入

```
1 ~ $ cd tmp/a/b/c || \
2 > mkdir -p tmp/a/b/c && \
3 > tar xvf -C tmp/a/b/c ~/archive.tar
```

或者，也可以使用以下配置：

清单 10. 好习惯 5 的替代示例：将反斜杠用于长输入

```
1 ~ $ cd tmp/a/b/c \
2 > || \
3 > mkdir -p tmp/a/b/c \
4 > && \
5 > tar xvf -C tmp/a/b/c ~/archive.tar
```

然而，当您输入行划分到多行上时，Shell 始终将其视为单个连续的行，因为它总是删除所有反斜杠和额外的空格。

注意：在大多数 Shell 中，当您按向上箭头键时，整个多行输入将重绘到单个长输入行上。

在列表中对命令分组

大多数 Shell 都具有在列表中对命令分组的方法，以便您能将它们的合计输出向下传递到某个管道，或者将其任何部分或全部流重定向到相同的地方。您一般可以通过在某个 Subshell 中运行一个命令列表或通过在当前 Shell 中运行一个命令列表来实现此目的。

在 Subshell 中运行命令列表

使用括号将命令列表包括在单个组中。这样做将在一个新的 Subshell 中运行命令，并允许您重定向或收集整组命令的输出，如以下示例所示：

清单 11. 好习惯 6 的示例：在 Subshell 中运行命令列表

```
1 ~ $ ( cd tmp/a/b/c/ || mkdir -p tmp/a/b/c && \  
2 > VAR=$PWD; cd ~; tar xvf -C $VAR archive.tar ) \  
3 > | mailx admin -S "Archive contents"
```

在此示例中，该存档的内容将提取到 tmp/a/b/c/ 目录中，同时将分组命令的输出（包括所提取文件的列表）通过邮件发送到地址 admin。

当您在命令列表中重新定义环境变量，并且您不希望将那些定义应用于当前 Shell 时，使用 Subshell 更可取。

在当前 Shell 中运行命令列表

将命令列表用大括号 ({}) 括起来，以在当前 Shell 中运行。确保在括号与实际命令之间包括空格，否则 Shell 可能无法正确解释括号。此外，还要确保列表中的最后一个命令以分号结尾，如以下示例所示：

清单 12. 好习惯 6 的另一个示例：在当前 Shell 中运行命令列表

```
1 ~ $ { cp ${VAR}a . && chown -R guest.guest a && \  
2 > tar cvf newarchive.tar a; } | mailx admin -S "New archive"
```

在 find 之外使用 xargs

使用 xargs 工具作为筛选器，以充分利用从 find 命令挑选的输出。find 运行通常提供与某些条件匹配的文件列表。此列表被传递到 xargs 上，后者然后使用该文件列表作为参数来运行其他某些有用的命令，如以下示例所示：

清单 13. xargs 工具的经典用法示例

```
1 ~ $ find some-file-criteria some-file-path | \  
2 > xargs some-great-command-that-needs-filename-arguments
```

然而，不要将 xargs 仅看作是 find 的辅助工具；它是一个未得到充分利用的工具之一，当您养成使用它的习惯时，将会希望进行所有试验，包括以下用法。

传递空格分隔的列表

在最简单的调用形式中，xargs 就像一个筛选器，它接受一个列表（每个成员分别在单独的行上）作为输入。该工具将那些成员放置在单个空格分隔的行上：

清单 14. xargs 工具产生的输出示例

```
1 ~ $ xargsabcControl-D  
2 a b c  
3 ~ $
```

您可以发送通过 xargs 来输出文件名的任何工具的输出，以便为其他某些接受文件名作为参数的工具获得参数列表，如以下示例所示：

清单 15. xargs 工具的使用示例

```
1 ~/tmp $ ls -l | xargs
2 December_Report.pdf README a archive.tar mkdirhier.sh
3 ~/tmp $ ls -l | xargs file
4 December_Report.pdf: PDF document, version 1.3
5 README: ASCII text
6 a: directory
7 archive.tar: POSIX tar archive
8 mkdirhier.sh: Bourne shell script text executable
9 ~/tmp $
```

xargs 命令不只用于传递文件名。您还可以在需要将文本筛选到单个行中的任何时候使用它：

清单 16. 好习惯 7 的示例：使用 xargs 工具来将文本筛选到单个行中

```
1 ~/tmp $ ls -l | xargs
2 -rw-r--r-- 7 joe joe 12043 Jan 27 20:36 December_Report.pdf -rw-r--r-- 1 \
3 root root 238 Dec 03 08:19 README drwxr-xr-x 38 joe joe 354082 Nov 02 \
4 16:07 a -rw-r--r-- 3 joe joe 5096 Dec 14 14:26 archive.tar -rwxr-xr-x 1 \
5 joe joe 3239 Sep 30 12:40 mkdirhier.sh
6 ~/tmp $
```

谨慎使用 xargs

从技术上讲，使用 xargs 很少遇到麻烦。缺省情况下，文件结束字符串是下划线 (_)；如果将该字符作为单个输入参数来发送，则它之后的所有内容将被忽略。为了防止这种情况发生，可以使用 -e 标志，它在不带参数的情况下完全禁用结束字符串。

了解何时 grep 应该执行计数——何时应该绕过

避免通过管道将 grep 发送到 wc -l 来对输出行数计数。grep 的 -c 选项提供了对与特定模式匹配的行的计数，并且一般要比通过管道发送到wc 更快，如以下示例所示：

清单 17. 好习惯 8 的示例：使用和不使用 grep 的行计数

```
1 ~ $ time grep and tmp/a/longfile.txt | wc -l
2 2811
3
4 real    0m0.097s
5 user    0m0.006s
6 sys     0m0.032s
7 ~ $ time grep -c and tmp/a/longfile.txt
8 2811
9
10 real    0m0.013s
11 user    0m0.006s
12 sys     0m0.005s
13 ~ $
```

除了速度因素外，-c 选项还是执行计数的好方法。对于多个文件，带 -c 选项的 grep 返回每个文件的单独计数，每行一个计数，而针对 wc 的管道则提供所有文件的组合总计数。

然而，不管是否考虑速度，此示例都表明了另一个要避免地常见错误。这些计数方法仅提供包含匹配模式的行数——如果那就是您要查找的结果，这没什么问题。但是在行中具有某个特定模式的多个实例的情况下，这些方法无法为您提供实际匹配实例数量的真实计数。归根结底，若要对实例计数，您还是要使用 wc 来计数。首先，使用 -o 选项（如果您的版本支持它的话）来运行 grep 命令。此选项仅输出匹配的模式，每行一个模式，而不输出行本身。但是您不能将它与 -c 选项结合使用，因此要使用 wc -l 来对行计数，如以下示例所示：

清单 18. 好习惯 8 的示例：使用 grep 对模式实例计数

```
1 ~ $ grep -o and tmp/a/longfile.txt | wc -l
2 3402
3 ~ $
```

在此例中，调用 wc 要比第二次调用 grep 并插入一个虚拟模式（例如 grep -c）来对行进行匹配和计数稍快一点。

匹配输出中的某些字段，而不只是对行进行匹配

当您只希望匹配输出行中特定字段中的模式时，诸如 awk 等工具要优于 grep。

下面经过简化的示例演示了如何仅列出 12 月修改过的文件。

清单 19. 坏习惯 9 的示例：使用 `grep` 来查找特定字段中的模式

```
1 ~/tmp $ ls -l /tmp/a/b/c | grep Dec
2 -rw-r--r-- 7 joe joe 12043 Jan 27 20:36 December_Report.pdf
3 -rw-r--r-- 1 root root 238 Dec 03 08:19 README
4 -rw-r--r-- 3 joe joe 5096 Dec 14 14:26 archive.tar
5 ~/tmp $
```

在此示例中，`grep` 对行进行筛选，并输出其修改日期和名称中带 `Dec` 的所有文件。因此，诸如 `December_Report.pdf` 等文件是匹配的，即使它自从一月份以来还未修改过。这可能不是您希望的结果。为了匹配特定字段中的模式，最好使用 `awk`，其中的一个关系运算符对确切的字段进行匹配，如下所示：

清单 20. 好习惯 9 的示例：使用 `awk` 来查找特定字段中的模式

```
1 ~/tmp $ ls -l | awk '$6 == "Dec"'
2 -rw-r--r-- 3 joe joe 5096 Dec 14 14:26 archive.tar
3 -rw-r--r-- 1 root root 238 Dec 03 08:19 README
4 ~/tmp $
```

停止对 `cat` 使用管道

`grep` 的一个常见的基本用法错误是通过管道将 `cat` 的输出发送到 `grep` 以搜索单个文件的内容。这绝对是不必要的，纯粹是浪费时间，因为诸如 `grep` 这样的工具接受文件名作为参数。您根本不需要在这种情况下使用 `cat`，如下所示：

清单 21. 好习惯和坏习惯 10 的示例：使用带和不带 `cat` 的 `grep`

```
1 ~ $ time cat tmp/a/longfile.txt | grep and
2 2811
3
4 real    0m0.015s
5 user    0m0.003s
6 sys     0m0.013s
7 ~ $ time grep and tmp/a/longfile.txt
8 2811
9
10 real    0m0.010s
11 user    0m0.006s
12 sys     0m0.004s
13 ~ $
```

此错误存在于许多工具中。由于大多数工具都接受使用连字符 (-) 的标准输入作为一个参数，因此即使使用 `cat` 来分散 `stdin` 中的多个文件，参数也通常是无效的。仅当您使用带多个筛选选项之一的 `cat` 时，才真正有必要在管道前首先执行连接。

结束语：养成好习惯

最好检查一下您的命令行习惯中的任何不良的使用模式。不良的使用模式会降低您的速度，并且通常会导致意外错误。本文介绍了 10 个新习惯，它们可以帮助您摆脱许多最常见的使用错误。养成这些好习惯是加强您的 UNIX 命令行技能的积极步骤。

加入伯乐在线专栏作者。扩大知名度，还能得赞赏！详见《[招募专栏作者](#)》

👍 赞

🔖 8 收藏





相关文章

- [Unix 高手的另外 10 个习惯](#)
- [UNIX 系统中的文本操作简介](#)
- [Linux Shell 创建序列数组](#)
- [5个有趣且能提高效率的超酷Unix操作](#)
- [一些实用但不为人知的Unix命令](#)
- [Ubuntu桌面生存指南\(1\)：选择 Linux](#)
- [在 Unix 系统上查找数据的最佳工具和技巧](#)

- [王垠：Unix的缺陷](#)
- [28个Unix/Linux的命令行神器](#)
- [详谈Unix环境下的进程异常退出](#)

可能感兴趣的话题

- [2017京东C/C++工程师笔试题](#)
- [今日头条2017前端工程师笔试题\(Javascript\)](#)
- [在校生一枚，想学点东西，望指教。 · \[👁 8\]\(#\)](#)
- [如何利用Java多线程编写一个web网站 · \[👁 4\]\(#\)](#)
- [零基础自学Python感觉很难，不像大家说的那么简单 · \[👁 61\]\(#\)](#)
- [当程序员面对小学数学题 · \[👁 18\]\(#\)](#)
- [前端大神有没有啊？遇到点问题想请教下！ · \[👁 1\]\(#\)](#)
- [你有哪些怪异的编程习惯？ · \[👁 4\]\(#\)](#)
- [程序猿的弱弱的亚健康 · \[👁 1\]\(#\)](#)
- [一个人去教堂是什么体验 · \[👁 29\]\(#\)](#)


[« 编译器是如何工作的？](#)
[十大前端开发框架（下） »](#)

登录后评论

新用户注册

直接登录


最新评论



- 

[Joseph Pan](#)

2014/01/14

谢谢，总结的非常好！看到了以前犯过的一些错误！






赞 [回复](#) 
- 

[Dante](#)

2014/01/14

mark



赞 [回复](#) 
- 

[陈建平](#)


2014/01/15


tar xvf -C tmp/a/b/c newarc.tar.gz

这条有问题

这样是对的，我用的是 ubuntu

tar xvf newarc.tar.gz -C tmp/a/b/c



赞 [回复](#) 

文章

输入搜索关键字

搜索



- [本周热门文章](#)
- [本月热门文章](#)
- [热门标签](#)

0 [LeetCode 刷题指南（1）：为什么...](#)

1 [换一个灯泡，要多少个程序员？](#)

- 2 [被 Facebook 开除是什么样的一种...](#)
- 3 [很认真地聊一聊程序员的自我修养](#)
- 4 [一周时间，初试 ML](#)
- 5 [细数 20 世纪最伟大的十大算法](#)
- 6 [浅谈程序员的英语学习](#)
- 7 [LVS：三种负载均衡方式比较+另三...](#)
- 8 [那些被岁月遗忘的 UNIX 经典著作](#)
- 9 [这里有一份面经，请查收（3）](#)



业界热点资讯

更多 »



[网联框架方案通过 第三方支付直连银行将终结](#)
10 小时前 · 7



[乌云多名高管“被捕”](#)
3 天前 · 50 · 21



[C是2016年最流行语言](#)
3 天前 · 16 · 2



[Uber从Postgres切换到MySQL](#)
3 天前 · 7 · 1



[出版商统计出最受欢迎的编程语言：Python 居首](#)
3 天前 · 6 · 1

精选工具资源

更多资源 »



[Jinja2：一个纯Python实现的模板引擎](#)
[Python](#), [模板引擎](#)



[Smack：一个开源的XMPP用于即时通讯的客户端类库](#)
[Java](#), [消息传递](#)



CSS的命名方式：BEM（区块、元素、修饰符）
CSS, 命名习惯和方式 · [Q_1](#)



如何精简TrelloCSS架构
CSS, 大型网站的CSS开发



Chardet：通用编码检测器
Python, 文本处理

最新评论

-  Re: [关于持续集成打包平台的Jenkins...](#)
服务器是在linux上吗
-  Re: [七种WebSocket框架的性能比较](#)
谁把数据整理一下, 弄个图表出来
-  Re: [你真的会用 Java 中的三目运...](#)
Map<String,Boolean> map = new HashM...
-  Re: [一个朋友过来面试引发我要说的一...](#)
正常情况是不用递归来实现链表的，递归会有栈溢出的潜在bug，况且还有没有考虑线程安全问题
-  Re: [如何在 Linux 上录制你的终端...](#)
没关系，asccinema完全可以在本地跑的。
-  Re: [加入伯乐在线专栏作者，扩大知名...](#)
你已经加入了专栏作者吗？只有加入了专栏才可以发文章
-  Re: [加入伯乐在线专栏作者，扩大知名...](#)
怎么发表文章？
-  Re: [Getty：Java NIO 框架设...](#)
如果有能力并且感兴趣的话，可以尝试写一个自己和自己聊天程序

关于伯乐在线博客

在这个信息爆炸的时代，人们已然被大量、快速并且简短的信息所包围。然而，我们相信：过多“快餐”式的阅读只会令人“虚胖”，缺乏实质的内涵。伯乐在线内容团队正试图以我们微薄的力量，把优秀的原创文章和译文分享给读者，为“快餐”添加一些“营养”元素。

快速链接
[问题反馈与求助](#) »
[加入伯乐翻译小组](#) »
[加入专栏作者](#) »

关注我们

新浪微博：[@伯乐在线官方微博](#)
RSS：[订阅地址](#)
推荐微信号



程序员的那些事

UI设计达人

极客范

合作联系

Email : bd@jobbole.com

QQ : 2302462408 (加好友请注明来意)

更多频道

- [小组](#) - 好的话题、有启发的回复、值得信赖的圈子
- [头条](#) - 分享和发现有价值的内容与观点
- [相亲](#) - 为IT单身男女服务的征婚传播平台
- [资源](#) - 优秀的工具资源导航
- [翻译](#) - 翻译传播优秀的外文文章
- [文章](#) - 国内外的精选文章
- [设计](#) - UI,网页,交互和用户体验
- [iOS](#) - 专注iOS技术分享
- [安卓](#) - 专注Android技术分享
- [前端](#) - JavaScript, HTML5, CSS
- [Java](#) - 专注Java技术分享
- [Python](#) - 专注Python技术分享

