第六章使用命令

Up to this point, we have seen a series of mysterious commands, each with its own mysterious options and arguments. In this chapter, we will attempt to remove some of that mystery and even create some of our own commands. The commands introduced in this chapter are:

在这之前，我们已经知道了一系列神秘的命令，每个命令都有自己奇妙的 选项和参数。在这一章中，我们将试图去掉一些神秘性，甚至创建我们自己 的命令。这一章将介绍以下命令：

* type – Indicate how a command name is interpreted
* type – 说明怎样解释一个命令名
* which – Display which executable program will be executed
* which – 显示会执行哪个可执行程序
* man – Display a command’s manual page
* man – 显示命令手册页
* apropos – Display a list of appropriate commands
* apropos – 显示一系列适合的命令
* info – Display a command’s info entry
* info – 显示命令 info
* whatis – Display a very brief description of a command
* whatis – 显示一个命令的简洁描述
* alias – Create an alias for a command
* alias – 创建命令别名

**到底什么是命令？**

A command can be one of four different things:

命令可以是下面四种形式之一：

1. An executable program like all those files we saw in /usr/bin. Within this category, programs can be compiled binaries such as programs written in C and C++, or programs written in scripting languages such as the shell, perl, python, ruby, etc.
2. A command built into the shell itself. bash supports a number of commands internally called shell builtins. The cd command, for example, is a shell builtin.
3. A shell function. These are miniature shell scripts incorporated into the environment. We will cover configuring the environment and writing shell functions in later chapters, but for now, just be aware that they exist.
4. An alias. Commands that we can define ourselves, built from other commands.
5. 是一个可执行程序，就像我们所看到的位于目录/usr/bin 中的文件一样。 属于这一类的程序，可以编译成二进制文件，诸如用 C 和 C++语言写成的程序, 也可以是由脚本语言写成的程序，比如说 shell，perl，python，ruby，等等。
6. 是一个内建于 shell 自身的命令。bash 支持若干命令，内部叫做 shell 内部命令 (builtins)。例如，cd 命令，就是一个 shell 内部命令。
7. 是一个 shell 函数。这些是小规模的 shell 脚本，它们混合到环境变量中。 在后续的章节里，我们将讨论配置环境变量以及书写 shell 函数。但是现在， 仅仅意识到它们的存在就可以了。
8. 是一个命令别名。我们可以定义自己的命令，建立在其它命令之上。

**识别命令**

It is often useful to know exactly which of the four kinds of commands is being used and Linux provides a couple of ways to find out.

这经常很有用，能确切地知道正在使用四类命令中的哪一类。Linux 提供了一对方法来 弄明白命令类型。

**type － 显示命令的类型**

The type command is a shell builtin that displays the kind of command the shell will execute, given a particular command name. It works like this:

type 命令是 shell 内部命令，它会显示命令的类别，给出一个特定的命令名（做为参数）。 它像这样工作：

type command

Where “command” is the name of the command you want to examine. Here are some examples:

“command”是你要检测的命令名。这里有些例子：

[me@linuxbox ~]$ type type

type is a shell builtins

[me@linuxbox ~]$ type ls

ls is aliased to `ls --color=tty`

[me@linuxbox ~]$ type cp

cp is /bin/cp

Here we see the results for three different commands. Notice that the one for ls (taken from a Fedora system) and how the ls command is actually an alias for the ls command with the “--color=tty” option added. Now we know why the output from ls is displayed in color!

我们看到这三个不同命令的检测结果。注意，ls 命令（在 Fedora 系统中）的检查结果，ls 命令实际上 是 ls 命令加上选项”--color=tty”的别名。现在我们知道为什么 ls 的输出结果是有颜色的！

**which － 显示一个可执行程序的位置**

Sometimes there is more than one version of an executable program installed on a system. While this is not very common on desktop systems, it’s not unusual on large servers. To determine the exact location of a given executable, the which command is used:

有时候在一个操作系统中，不只安装了可执行程序的一个版本。然而在桌面系统中，这并不普遍， 但在大型服务器中，却很平常。为了确定所给定的执行程序的准确位置，使用 which 命令：

[me@linuxbox ~]$ which ls

/bin/ls

which only works for executable programs, not builtins nor aliases that are substitutes for actual executable programs. When we try to use which on a shell builtin, for example, cd, we either get no response or an error message:

这个命令只对可执行程序有效，不包括内部命令和命令别名，别名是真正的可执行程序的替代物。 当我们试着使用 shell 内部命令时，例如，cd 命令，我们或者得不到回应，或者是个错误信息：

[me@linuxbox ~]$ which cd

/usr/bin/which: no cd in

(/opt/jre1.6.0\_03/bin:/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/opt/jre1

.6.0\_03/bin:/usr/lib/ccache:/usr/local/bin:/usr/bin:/bin:/home/me/bin)

which is a fancy way of saying “command not found.”

说“命令没有找到”，真是很奇特。

**得到命令文档**

With this knowledge of what a command is, we can now search for the documentation available for each kind of command.

知道了什么是命令，现在我们来寻找每一类命令的可得到的文档。

**help － 得到 shell 内部命令的帮助文档**

bash has a built-in help facility available for each of the shell builtins. To use it, type “help” followed by the name of the shell builtin. For example:

bash 有一个内建的帮助工具，可供每一个 shell 内部命令使用。输入“help”，接着是 shell 内部命令名。例如：

[me@linuxbox ~]$ help cd

cd: cd [-L|-P] [dir]

Change ...

A note on notation: When square brackets appear in the description of a command’s syntax, they indicate optional items. A vertical bar character indicates mutually exclusive items. In the case of the cd command above:

注意表示法：出现在命令语法说明中的方括号，表示可选的项目。一个竖杠字符 表示互斥选项。在上面 cd 命令的例子中：

cd [-L|-P] [dir]

This notation says that the command cd may be followed optionally by either a “-L” or a “-P” and further, optionally followed by the argument “dir”.

这种表示法说明，cd 命令可能有一个“-L”选项或者“-P”选项，进一步，可能有参数“dir”。

While the output of help for the cd commands is concise and accurate, it is by no means tutorial and as we can see, it also seems to mention a lot of things we haven’t talked about yet! Don’t worry. We’ll get there.

虽然 cd 命令的帮助文档很简洁准确，但它决不是教材。正如我们所看到的，它似乎提到了许多 我们还没有谈论到的东西！不要担心，我们会学到的。

**--help - 显示用法信息**

Many executable programs support a “--help” option that displays a description of the command’s supported syntax and options. For example:

许多可执行程序支持一个 --help 选项，这个选项是显示命令所支持的语法和选项说明。例如：

[me@linuxbox ~]$ mkdir --help

Usage: mkdir [OPTION] DIRECTORY...

Create ...

Some programs don’t support the “--help” option, but try it anyway. Often it results in an error message that will reveal the same usage information.

一些程序不支持 --help 选项，但不管怎样试一下。这经常会导致输出错误信息，但同时能 揭示一样的命令用法信息。

**man － 显示程序手册页**

Most executable programs intended for command line use provide a formal piece of documentation called a manual or man page. A special paging program called man is used to view them. It is used like this:

许多希望被命令行使用的可执行程序，提供了一个正式的文档，叫做手册或手册页(man page)。一个特殊的叫做 man 的分页程序，可用来浏览他们。它是这样使用的：

man program

where “program” is the name of the command to view.

“program”是要浏览的命令名。

Man pages vary somewhat in format but generally contain a title, a synopsis of the command’s syntax, a description of the command’s purpose, and a listing and description of each of the command’s options. Man pages, however, do not usually include examples, and are intended as a reference, not a tutorial. As an example, let’s try viewing the man page for the ls command:

手册文档的格式有点不同，一般地包含一个标题，命令语法的纲要，命令用途的说明， 和命令选项列表，及每个选项的说明。然而，通常手册文档并不包含实例，它打算 作为一本参考手册，而不是教材。作为一个例子，浏览一下 ls 命令的手册文档：

[me@linuxbox ~]$ man ls

On most Linux systems, man uses less to display the manual page, so all of the familiar less commands work while displaying the page.

在大多数 Linux 系统中，man 使用 less 工具来显示参考手册，所以当浏览文档时，你所熟悉的 less 命令都能有效。

The “manual” that man displays is broken into sections and not only covers user commands but also system administration commands, programming interfaces, file formats and more. The table below describes the layout of the manual:

man 所显示的参考手册，被分成几个章节，它们不仅仅包括用户命令，也包括系统管理员 命令，程序接口，文件格式等等。下表描绘了手册的布局：

| *Table 6-1: Man Page Organization* | |
| --- | --- |
| Section | Contents |
| 1 | User commands |
| 2 | Programming interfaces kernel system calls |
| 3 | Programming interfaces to the C library |
| 4 | Special files such as device nodes and drivers |
| 5 | File formats |
| 6 | Games and amusements such as screen savers |
| 7 | Miscellaneous |
| 8 | System administration commands |

| *表6-1: 手册页的组织形式* | |
| --- | --- |
| 章节 | 内容 |
| 1 | 用户命令 |
| 2 | 程序接口内核系统调用 |
| 3 | C 库函数程序接口 |
| 4 | 特殊文件，比如说设备结点和驱动程序 |
| 5 | 文件格式 |
| 6 | 游戏娱乐，如屏幕保护程序 |
| 7 | 其他方面 |
| 8 | 系统管理员命令 |

Sometimes we need to look in a specific section of the manual to find what we are looking for. This is particularly true if we are looking for a file format that is also the name of a command. Without specifying a section number, we will always get the first instance of a match, probably in section 1. To specify a section number, we use man like this:

有时候，我们需要查看参考手册的特定章节，从而找到我们需要的信息。 如果我们要查找一种文件格式，而同时它也是一个命令名时,这种情况尤其正确。 没有指定章节号，我们总是得到第一个匹配项，可能在第一章节。我们这样使用 man 命令， 来指定章节号：

man section search\_term

For example:

例如：

[me@linuxbox ~]$ man 5 passwd

This will display the man page describing the file format of the /etc/passwd file.

命令运行结果会显示文件 /etc/passwd 的文件格式说明手册。

**apropos － 显示适当的命令**

It is also possible to search the list of man pages for possible matches based on a search term. It’s very crude but sometimes helpful. Here is an example of a search for man pages using the search term “floppy”:

也有可能搜索参考手册列表，基于某个关键字的匹配项。虽然很粗糙但有时很有用。 下面是一个以”floppy”为关键词来搜索参考手册的例子：

[me@linuxbox ~]$ apropos floppy

create\_floppy\_devices (8) - udev callout to create all possible

...

The first field in each line of output is the name of the man page, the second field shows the section. Note that the man command with the “-k” option performs the exact same function as apropos.

输出结果每行的第一个字段是手册页的名字，第二个字段展示章节。注意，man 命令加上”-k”选项， 和 apropos 完成一样的功能。

**whatis － 显示非常简洁的命令说明**

The whatis program displays the name and a one line description of a man page matching a specified keyword:

whatis 程序显示匹配特定关键字的手册页的名字和一行命令说明：

The Most Brutal Man Page Of Them All

最晦涩难懂的手册页

As we have seen, the manual pages supplied with Linux and other Unix-like systems are intended as reference documentation and not as tutorials. Many man pages are hard to read, but I think that the grand prize for difficulty has got to go to the man page for bash. As I was doing my research for this book, I gave it careful review to ensure that I was covering most of its topics. When printed, it’s over eighty pages long and extremely dense, and its structure makes absolutely no sense to a new user.

正如我们所看到的，Linux 和类 Unix 的系统提供的手册页，只是打算作为参考手册使用， 而不是教材。许多手册页都很难阅读，但是我认为由于阅读难度而能拿到特等奖的手册页应该是 bash 手册页。因为我正在为这本书做我的研究，所以我很仔细地浏览了整个 bash 手册，为的是确保我讲述了 大部分的 bash 主题。当把 bash 参考手册整个打印出来，其篇幅有八十多页且内容极其紧密， 但对于初学者来说，其结构安排毫无意义。

On the other hand, it is very accurate and concise, as well as being extremely complete. So check it out if you dare and look forward to the day when you can read it and it all makes sense.

另一方面，bash 参考手册的内容非常简明精确，同时也非常完善。所以，如果你有胆量就查看一下， 并且期望有一天你能读懂它。

**info － 显示程序 Info 条目**

The GNU Project provides an alternative to man pages for their programs, called “info.” Info pages are displayed with a reader program named, appropriately enough, info. Info pages are hyperlinked much like web pages. Here is a sample:

GNU 项目提供了一个命令程序手册页的替代物，称为”info”。info 内容可通过 info 阅读器 程序读取。info 页是超级链接形式的，和网页很相似。这有个例子：

File: coreutils.info, Node: ls invocation, Next: dir invocation,

Up: Directory listing

10.1 `ls': List directory contents

==================================

...

The info program reads info files, which are tree structured into individual nodes, each containing a single topic. Info files contain hyperlinks that can move you from node to node. A hyperlink can be identified by its leading asterisk, and is activated by placing the cursor upon it and pressing the enter key.

info 程序读取 info 文件，info 文件是树型结构，分化为各个结点，每一个包含一个题目。 info 文件包含超级链接，它可以让你从一个结点跳到另一个结点。一个超级链接可通过 它开头的星号来辨别出来，把光标放在它上面并按下 enter 键，就可以激活它。

To invoke info, type “info” followed optionally by the name of a program. Below is a table of commands used to control the reader while displaying an info page:

输入”info”，接着输入程序名称，启动 info。下表中的命令，当显示一个 info 页面时， 用来控制阅读器。

| *Tbale 6-2: info Commands* | |
| --- | --- |
| Command | Action |
| ? | Display command help |
| PgUp or Backspace | Display privious page |
| PgDn or Space | Display next page |
| n | Next - Display the next node |
| p | Previous - Display the previous node |
| u | Up - Display the parent node of the currently displayed node, usually a menu. |
| Enter | Follow the hyperlink at the cursor location |
| q | Quit |

| *表 6-2: info 命令* | |
| --- | --- |
| 命令 | 行为 |
| ? | 显示命令帮助 |
| PgUp or Backspace | 显示上一页 |
| PgDn or Space | 显示下一页 |
| n | 下一个 - 显示下一个结点 |
| p | 上一个 - 显示上一个结点 |
| u | Up - 显示当前所显示结点的父结点，通常是个菜单 |
| Enter | 激活光标位置下的超级链接 |
| q | 退出 |

Most of the command line programs we have discussed so far are part of the GNU Project’s “coreutils” package, so typing:

到目前为止，我们所讨论的大多数命令行程序，属于 GNU 项目”coreutils”包，所以输入：

[me@linuxbox ~]$ info coreutils

will display a menu page with hyperlinks to each program contained in the coreutils package.

将会显示一个包含超级链接的手册页，这些超级链接指向包含在 coreutils 包中的各个程序。

**README 和其它程序文档**

Many software packages installed on your system have documentation files residing in the /usr/share/doc directory. Most of these are stored in plain text format and can be viewed with less. Some of the files are in HTML format and can be viewed with a web browser. We may encounter some files ending with a “.gz” extension. This indicates that they have been compressed with the gzip compression program. The gzip package includes a special version of less called zless that will display the contents of gzip-compressed text files.

许多安装在你系统中的软件，都有自己的文档文件，这些文件位于/usr/share/doc 目录下。 这些文件大多数是以文本文件的形式存储的，可用 less 阅读器来浏览。一些文件是 HTML 格式， 可用网页浏览器来阅读。我们可能遇到许多以”.gz”结尾的文件。这表示 gzip 压缩程序 已经压缩了这些文件。gzip 软件包包括一个特殊的 less 版本，叫做 zless，zless 可以显示由 gzip 压缩的文本文件的内容。

**用别名（alias）创建你自己的命令**

Now for our very first experience with programming! We will create a command of our own using the alias command. But before we start, we need to reveal a small command line trick. It’s possible to put more than one command on a line by separating each command with a semicolon character. It works like this:

现在是时候，感受第一次编程经历了！我们将用 alias 命令创建我们自己的命令。但在 开始之前，我们需要展示一个命令行小技巧。可以把多个命令放在同一行上，命令之间 用”;”分开。它像这样工作：

command1; command2; command3...

Here’s the example we will use:

我们会用到下面的例子：

[me@linuxbox ~]$ cd /usr; ls; cd -

bin games kerberos lib64 local share tmp

...

[me@linuxbox ~]$

As we can see, we have combined three commands on one line. First we change directory to /usr then list the directory and finally return to the original directory (by using ‘cd -‘) so we end up where we started. Now let’s turn this sequence into a new command using alias. The first thing we have to do is dream up a name for our new command. Let’s try “test”. Before we do that, it would be a good idea to find out if the name “test” is already being used. To find out, we can use the type command again:

正如我们看到的，我们在一行上联合了三个命令。首先更改目录到/usr，然后列出目录 内容，最后回到原始目录（用命令”cd -“）,结束在开始的地方。现在，通过 alias 命令 把这一串命令转变为一个命令。我们要做的第一件事就是为我们的新命令构想一个名字。 比方说”test”。在使用”test”之前，查明是否”test”命令名已经存在系统中，是个很不错 的主意。为了查清此事，可以使用 type 命令：

[me@linuxbox ~]$ type test

test is a shell builtin

Oops! The name “test” is already taken. Let’s try “foo”:

哦！”test”名字已经被使用了。试一下”foo”:

[me@linuxbox ~]$ type foo

bash: type: foo: not found

Great! “foo” is not taken. So let’s create our alias:

太棒了！”foo”还没被占用。创建命令别名：

[me@linuxbox ~]$ alias foo='cd /usr; ls; cd -'

Notice the structure of this command:

注意命令结构：

alias name='string'

After the command “alias” we give alias a name followed immediately (no whitespace allowed) by an equals sign, followed immediately by a quoted string containing the meaning to be assigned to the name. After we define our alias, it can be used anywhere the shell would expect a command. Let’s try it:

在命令”alias”之后，输入“name”，紧接着（没有空格）是一个等号，等号之后是 一串用引号引起的字符串，字符串的内容要赋值给 name。我们定义了别名之后， 这个命令别名可以使用在任何地方。试一下：

[me@linuxbox ~]$ foo

bin games kerberos lib64 local share tmp

...

[me@linuxbox ~]$

We can also use the type command again to see our alias:

我们也可以使用 type 命令来查看我们的别名：

[me@linuxbox ~]$ type foo

foo is aliased to `cd /usr; ls ; cd -'

To remove an alias, the unalias command is used, like so:

删除别名，使用 unalias 命令，像这样：

[me@linuxbox ~]$ unalias foo

[me@linuxbox ~]$ type foo

bash: type: foo: not found

While we purposefully avoided naming our alias with an existing command name, it is not uncommon to do so. This is often done to apply a commonly desired option to each invocation of a common command. For instance, we saw earlier how the ls command is often aliased to add color support:

虽然我们有意避免使用已经存在的命令名来命名我们的别名，但这是常做的事情。通常， 会把一个普遍用到的选项加到一个经常使用的命令后面。例如，之前见到的 ls 命令，会 带有色彩支持：

[me@linuxbox ~]$ type ls

ls is aliased to 'ls --color=tty'

To see all the aliases defined in the environment, use the alias command without arguments. Here are some of the aliases defined by default on a Fedora system. Try and figure out what they all do:

要查看所有定义在系统环境中的别名，使用不带参数的 alias 命令。下面在 Fedora 系统中 默认定义的别名。试着弄明白，它们是做什么的：

[me@linuxbox ~]$ alias

alias l.='ls -d .\* --color=tty'

...

There is one tiny problem with defining aliases on the command line. They vanish when your shell session ends. In a later chapter, we will see how to add our own aliases to the files that establish the environment each time we log on, but for now, enjoy the fact that we have taken our first, albeit tiny, step into the world of shell programming!

在命令行中定义别名有点儿小问题。当你的 shell 会话结束时，它们会消失。随后的章节里， 我们会了解怎样把自己的别名添加到文件中去，每次我们登录系统，这些文件会建立系统环境。 现在，好好享受我们刚经历过的，步入 shell 编程世界的第一步吧，虽然微小。

**拜访老朋友**

Now that we have learned how to find the documentation for commands, go and look up the documentation for all the commands we have encountered so far. Study what additional options are available and try them out!

既然我们已经学习了怎样找到命令的帮助文档，那就试着查阅，到目前为止，我们学到的所有 命令的文档。学习命令其它可用的选项，练习一下！