简介 模式扩展

Bash 的基本语法

网道（WangDoc.com），互联网文档计划

本章介绍 Bash 的最基本语法。

目录 [隐藏]

1. echo 命令

由于后面的例子会大量用到 `echo` 命令，这里先介绍这个命令。

`echo` 命令的作用是在屏幕输出一行文本，可以将该命令的参数原样输出。

```
$ echo hello world
```

```
hello world
```

上面例子中， `echo` 的参数是 `hello world` ，可以原样输出。

如果想要输出的是多行文本，即包括换行符。这时就需要把多行文本放在引号里面。

```
$ echo "<HTML>
    <HEAD>
          <TITLE>Page Title</TITLE>
    </HEAD>
    <BODY>
          Page body.
    </BODY>
</HTML>"
```

上面例子中， `echo` 可以原样输出多行文本。

1.1 -n 参数

默认情况下， `echo` 输出的文本末尾会有一个回车符。 `-n` 参数可以取消末尾的回车符，使得下一个提示符紧跟在输出内容的后面。

```
$ echo -n hello world
hello world$
```

上面例子中， `world` 后面直接就是下一行的提示符 `$` 。

```
$ echo a;echo b
a
b

$ echo -n a;echo b
ab
```

上面例子中， `-n` 参数可以让两个 `echo` 命令的输出连在一起，出现在同一行。

链接

本文源码
代码仓库
反馈

1.2 `-e` 参数

`-e` 参数会解释引号（双引号和单引号）里面的特殊字符（比如换行符 `\n` ）。如果不使用 `-e` 参数，即默认情况下，引号会让特殊字符变成普通字符， `echo` 不解释它们，原样输出。

```
$ echo "Hello\nWorld"
Hello\nWorld

# 双引号的情况
$ echo -e "Hello\nWorld"
Hello
World

# 单引号的情况
$ echo -e 'Hello\nWorld'
Hello
World
```

上面代码中， `-e` 参数使得 `\n` 解释为换行符，导致输出内容里面出现换行。

2. 命令格式

命令行环境中，主要通过使用 Shell 命令，进行各种操作。Shell 命令基本都是下面的格式。

```
$ command [ arg1 ... [ argN ]]
```

上面代码中， `command` 是具体的命令或者一个可执行文件， `arg1 ... argN` 是传递给命令的参数，它们是可选的。

```
$ ls -l
```

上面这个命令中， `ls` 是命令， `-l` 是参数。

有些参数是命令的配置项，这些配置项一般都以一个连词线开头，比如上面的 `-l` 。同一个配置项往往有长和短两种形

式，比如 `-l` 是短形式， `--list` 是长形式，它们的作用完全相同。短形式便于手动输入，长形式一般用在脚本之中，可读性更好，利于解释自身的含义。

```
# 短形式
$ ls -r

# 长形式
$ ls --reverse
```

上面命令中， `-r` 是短形式， `--reverse` 是长形式，作用完全一样。前者便于输入，后者便于理解。

Bash 单个命令一般都是一行，用户按下回车键，就开始执行。有些命令比较长，写成多行会有利于阅读和编辑，这时可以在每一行的结尾加上反斜杠，Bash 就会将下一行跟当前行放在一起解释。

```
$ echo foo bar

# 等同于
$ echo foo \
bar
```

3. 空格

Bash 使用空格（或 Tab 键）区分不同的参数。

```
$ command foo bar
```

上面命令中， `foo` 和 `bar` 之间有一个空格，所以 Bash 认为它们是两个参数。

如果参数之间有多个空格，Bash 会自动忽略多余的空格。

```
$ echo this is a     test
this is a test
```

上面命令中， `a` 和 `test` 之间有多个空格，Bash 会忽略多余的空格。

4. 分号

分号（ `;` ）是命令的结束符，使得一行可以放置多个命令，上一个命令执行结束后，再执行第二个命令。

```
$ clear; ls
```

上面例子中，Bash 先执行 `clear` 命令，执行完成后，再执行 `ls` 命令。

注意，使用分号时，第二个命令总是接着第一个命令执行，不管第一个命令执行成功或失败。

5. 命令的组合符 && 和 ||

除了分号，Bash 还提供两个命令组合符 `&&` 和 `||` ，允许更好地控制多个命令之间的继发关系。

```
Command1 && Command2
```

上面命令的意思是，如果 `Command1` 命令运行成功，则继续运行 `Command2` 命令。

```
Command1 || Command2
```

上面命令的意思是，如果 `Command1` 命令运行失败，则继续运行 `Command2` 命令。

下面是一些例子。

```
$ cat filelist.txt ; ls -l filelist.txt
```

上面例子中，只要 `cat` 命令执行结束，不管成功或失败，都会继续执行 `ls` 命令。

```
$ cat filelist.txt && ls -l filelist.txt
```

上面例子中，只有 `cat` 命令执行成功，才会继续执行 `ls` 命令。如果 `cat` 执行失败（比如不存在文件 `flielist.txt` ），那么 `ls` 命令就不会执行。

```
$ mkdir foo || mkdir bar
```

上面例子中，只有 `mkdir foo` 命令执行失败（比如 `foo` 目录已经存在），才会继续执行 `mkdir bar` 命令。如果 `mkdir foo` 命令执行成功，就不会创建 `bar` 目录了。

6. type 命令

Bash 本身内置了很多命令，同时也可以执行外部程序。怎么知道一个命令是内置命令，还是外部程序呢？

`type` 命令用来判断命令的来源。

```
$ type echo
echo is a shell builtin
$ type ls
ls is hashed (/bin/ls)
```

上面代码中， `type` 命令告诉我们， `echo` 是内部命令， `ls` 是外部程序（ `/bin/ls` ）。

`type` 命令本身也是内置命令。

```
$ type type
type is a shell builtin
```

如果要查看一个命令的所有定义，可以使用 `type` 命令的 `-a` 参数。

```
$ type -a echo
echo is shell builtin
echo is /usr/bin/echo
echo is /bin/echo
```

上面代码表示， `echo` 命令既是内置命令，也有对应的外部程序。

`type` 命令的 `-t` 参数，可以返回一个命令的类型：别名（alias），关键词（keyword），函数（function），内置命令（builtin）和文件（file）。

```
$ type -t bash
file
$ type -t if
keyword
```

上面例子中， `bash` 是文件， `if` 是关键词。

7. 快捷键

Bash 提供很多快捷键，可以大大方便操作。下面是一些最常用的快捷键，完整的介绍参见《行操作》一章。

- `Ctrl + L` ：清除屏幕并将当前行移到页面顶部。
- `Ctrl + C` ：中止当前正在执行的命令。
- `Shift + PageUp` ：向上滚动。
- `Shift + PageDown` ：向下滚动。
- `Ctrl + U` ：从光标位置删除到行首。
- `Ctrl + K` ：从光标位置删除到行尾。
- `Ctrl + W` ：删除光标位置前一个单词。
- `Ctrl + D` ：关闭 Shell 会话。
- `↑` ， `↓` ：浏览已执行命令的历史记录。

除了上面的快捷键，Bash 还具有自动补全功能。命令输入到一半的时候，可以按下 Tab 键，Bash 会自动完成剩下的部分。比如，输入 `tou` ，然后按一下 Tab 键，Bash 会自动补上 `ch` 。

除了命令的自动补全，Bash 还支持路径的自动补全。有时，需要输入很长的路径，这时只需要输入前面的部分，然后按下 Tab 键，就会自动补全后面的部分。如果有多个可能的选择，按两次 Tab 键，Bash 会显示所有选项，让你选择。

简介 模式扩展

分享本文 

联系：contact@wangdoc.com