第三十二章 流程控制：case分支

In this chapter, we will continue to look at flow control. In Chapter 28, we constructed some simple menus and built the logic used to act on a user’s selection. To do this, we used a series of if commands to identify which of the possible choices has been selected. This type of construct appears frequently in programs, so much so that many programming languages (including the shell) provide a flow control mechanism for multiple-choice decisions.

在这一章中，我们将继续看一下程序的流程控制。在第28章中，我们构建了一些简单的菜单并创建了用来 应对各种用户选择的程序逻辑。为此，我们使用了一系列的 if 命令来识别哪一个可能的选项已经被选中。 这种类型的构造经常出现在程序中，出现频率如此之多，以至于许多编程语言（包括 shell） 专门为多选决策提供了一种流程控制机制。

**case**

The bash multiple-choice compound command is called case. It has the following syntax:

Bash 的多选复合命令称为 case。它的语法规则如下所示：

case word in

[pattern [| pattern]...) commands ;;]...

esac

If we look at the read-menu program from Chapter 28, we see the logic used to act on a user’s selection:

如果我们看一下第28章中的读菜单程序，我们就知道了用来应对一个用户选项的逻辑流程：

#!/bin/bash

# read-menu: a menu driven system information program

clear

echo "

Please Select:

1. Display System Information

2. Display Disk Space

3. Display Home Space Utilization

0. Quit

"

read -p "Enter selection [0-3] > "

if [[ $REPLY =~ ^[0-3]$ ]]; then

if [[ $REPLY == 0 ]]; then

echo "Program terminated."

exit

fi

if [[ $REPLY == 1 ]]; then

echo "Hostname: $HOSTNAME"

uptime

exit

fi

if [[ $REPLY == 2 ]]; then

df -h

exit

fi

if [[ $REPLY == 3 ]]; then

if [[ $(id -u) -eq 0 ]]; then

echo "Home Space Utilization (All Users)"

du -sh /home/\*

else

echo "Home Space Utilization ($USER)"

du -sh $HOME

fi

exit

fi

else

echo "Invalid entry." >&2

exit 1

fi

Using case, we can replace this logic with something simpler:

使用 case 语句，我们可以用更简单的代码替换这种逻辑：

#!/bin/bash

# case-menu: a menu driven system information program

clear

echo "

Please Select:

1. Display System Information

2. Display Disk Space

3. Display Home Space Utilization

0. Quit

"

read -p "Enter selection [0-3] > "

case $REPLY in

0) echo "Program terminated."

exit

;;

1) echo "Hostname: $HOSTNAME"

uptime

;;

2) df -h

;;

3) if [[ $(id -u) -eq 0 ]]; then

echo "Home Space Utilization (All Users)"

du -sh /home/\*

else

echo "Home Space Utilization ($USER)"

du -sh $HOME

fi

;;

\*) echo "Invalid entry" >&2

exit 1

;;

esac

The case command looks at the value of word, in our example, the value of the REPLY variable, and then attempts to match it against one of the specified patterns. When a match is found, the commands associated with the specified pattern are executed. After a match is found, no further matches are attempted.

case 命令检查一个变量值，在我们这个例子中，就是 REPLY 变量的变量值，然后试图去匹配其中一个具体的模式。 当与之相匹配的模式找到之后，就会执行与该模式相关联的命令。若找到一个模式之后，就不会再继续寻找。

**模式**

The patterns used by case are the same as those used by pathname expansion. Patterns are terminated with a “)” character. Here are some valid patterns:

这里 case 语句使用的模式和路径展开中使用的那些是一样的。模式以一个 “)” 为终止符。这里是一些有效的模式。

|  |  |
| --- | --- |
| *Table 32-1: case Pattern Examples* | |
| Pattern | Description |
| a) | Matches if word equals "a". |
| [[:alpha:]]) | Matches if word is a single alphabetic character. |
| ???) | Matches if word is exactly three characters long. |
| \*.txt) | Matches if word ends with the characters “.txt”. |
| \*) | Matches any value of word. It is good practice to include this as the last pattern in a case command, to catch any values of word that did not match a previous pattern; that is, to catch any possible invalid values. |

|  |  |
| --- | --- |
| *表32-1: case 模式实例* | |
| 模式 | 描述 |
| a) | 若单词为 “a”，则匹配 |
| [[:alpha:]]) | 若单词是一个字母字符，则匹配 |
| ???) | 若单词只有3个字符，则匹配 |
| \*.txt) | 若单词以 “.txt” 字符结尾，则匹配 |
| \*) | 匹配任意单词。把这个模式做为 case 命令的最后一个模式，是一个很好的做法， 可以捕捉到任意一个与先前模式不匹配的数值；也就是说，捕捉到任何可能的无效值。 |

Here is an example of patterns at work:

这里是一个模式使用实例：

#!/bin/bash

read -p "enter word > "

case $REPLY in

[[:alpha:]]) echo "is a single alphabetic character." ;;

[ABC][0-9]) echo "is A, B, or C followed by a digit." ;;

???) echo "is three characters long." ;;

\*.txt) echo "is a word ending in '.txt'" ;;

\*) echo "is something else." ;;

esac

It is also possible to combine multiple patterns using the vertical bar character as a separator. This creates an “or” conditional pattern. This is useful for such things as handling both upper- and lowercase characters. For example:

还可以使用竖线字符作为分隔符，把多个模式结合起来。这就创建了一个 “或” 条件模式。这对于处理诸如大小写字符很有用处。例如：

#!/bin/bash

# case-menu: a menu driven system information program

clear

echo "

Please Select:

A. Display System Information

B. Display Disk Space

C. Display Home Space Utilization

Q. Quit

"

read -p "Enter selection [A, B, C or Q] > "

case $REPLY in

q|Q) echo "Program terminated."

exit

;;

a|A) echo "Hostname: $HOSTNAME"

uptime

;;

b|B) df -h

;;

c|C) if [[ $(id -u) -eq 0 ]]; then

echo "Home Space Utilization (All Users)"

du -sh /home/\*

else

echo "Home Space Utilization ($USER)"

du -sh $HOME

fi

;;

\*) echo "Invalid entry" >&2

exit 1

;;

esac

Here, we modify the case-menu program to use letters instead of digits for menu selection. Notice how the new patterns allow for entry of both upper- and lowercase letters.

这里，我们更改了 case-menu 程序的代码，用字母来代替数字做为菜单选项。注意新模式如何使得大小写字母都是有效的输入选项。

**执行多个动作**

In versions of bash prior to 4.0, case allowed only one action to be performed on a successful match. After a successful match, the command would terminate. Here we see a script that tests a character:

早于版本号4.0的 bash，case 语法只允许执行与一个成功匹配的模式相关联的动作。 匹配成功之后，命令将会终止。这里我们看一个测试一个字符的脚本：

#!/bin/bash

# case4-1: test a character

read -n 1 -p "Type a character > "

echo

case $REPLY in

[[:upper:]]) echo "'$REPLY' is upper case." ;;

[[:lower:]]) echo "'$REPLY' is lower case." ;;

[[:alpha:]]) echo "'$REPLY' is alphabetic." ;;

[[:digit:]]) echo "'$REPLY' is a digit." ;;

[[:graph:]]) echo "'$REPLY' is a visible character." ;;

[[:punct:]]) echo "'$REPLY' is a punctuation symbol." ;;

[[:space:]]) echo "'$REPLY' is a whitespace character." ;;

[[:xdigit:]]) echo "'$REPLY' is a hexadecimal digit." ;;

esac

Running this script produces this:

运行这个脚本，输出这些内容：

[me@linuxbox ~]$ case4-1

Type a character > a

'a' is lower case.

The script works for the most part, but fails if a character matches more than one of the POSIX characters classes. For example, the character “a” is both lower case and alphabetic, as well as a hexadecimal digit. In bash prior to version 4.0 there was no way for case to match more than one test. Modern versions of bash, add the “;;&” notation to terminate each action, so now we can do this:

大多数情况下这个脚本工作是正常的，但若输入的字符不止与一个 POSIX 字符集匹配的话，这时脚本就会出错。 例如，字符 “a” 既是小写字母，也是一个十六进制的数字。早于4.0的 bash，对于 case 语法绝不能匹配 多个测试条件。现在的 bash 版本，添加 “;;&” 表达式来终止每个行动，所以现在我们可以做到这一点：

#!/bin/bash

# case4-2: test a character

read -n 1 -p "Type a character > "

echo

case $REPLY in

[[:upper:]]) echo "'$REPLY' is upper case." ;;&

[[:lower:]]) echo "'$REPLY' is lower case." ;;&

[[:alpha:]]) echo "'$REPLY' is alphabetic." ;;&

[[:digit:]]) echo "'$REPLY' is a digit." ;;&

[[:graph:]]) echo "'$REPLY' is a visible character." ;;&

[[:punct:]]) echo "'$REPLY' is a punctuation symbol." ;;&

[[:space:]]) echo "'$REPLY' is a whitespace character." ;;&

[[:xdigit:]]) echo "'$REPLY' is a hexadecimal digit." ;;&

esac

When we run this script, we get this:

当我们运行这个脚本的时候，我们得到这些：

[me@linuxbox ~]$ case4-2

Type a character > a

'a' is lower case.

'a' is alphabetic.

'a' is a visible character.

'a' is a hexadecimal digit.

The addition of the “;;&” syntax allows case to continue on to the next test rather than simply terminating.

添加的 “;;&” 的语法允许 case 语句继续执行下一条测试，而不是简单地终止运行。

**总结**

The case command is a handy addition to our bag of programming tricks. As we will see in the next chapter, it’s the perfect tool for handling certain types of problems.

case 命令是我们编程技巧口袋中的一个便捷工具。在下一章中我们将看到， 对于处理某些类型的问题来说，case 命令是一个完美的工具。

**拓展阅读**

* The Bash Reference Manual section on Conditional Constructs describes the case command in detail:
* Bash 参考手册的条件构造一节详尽的介绍了 case 命令：

<http://tiswww.case.edu/php/chet/bash/bashref.html#SEC21>

* The Advanced Bash-Scripting Guide provides further examples of case applications:
* 高级 Bash 脚本指南提供了更深一层的 case 应用实例：

<http://tldp.org/LDP/abs/html/testbranch.html>