Оглавление

31 Управление потоком выполнения: ветвление с помощью case

В этой главе мы продолжим знакомство с инструментами управления потоком выполнения. В главе 28 мы сконструировали простое меню и реализовали логику обработки выбора его пунктов пользователем. Для этого использовалась серия команд `if`, выясняющих, какой из возможных вариантов выбран. Такие конструкции часто можно увидеть в программах, причем так часто, что в некоторых языках программирования (включая командную оболочку) был реализован механизм управления потоком выполнения для случаев с множеством альтернативных вариантов.

Команда case

Командная оболочка `bash` поддерживает составную команду выбора из нескольких вариантов, которая называется `case`. Она имеет следующий синтаксис:

```
case слово in
      [шаблон [| шаблон]...) команды ;;]...
esac
```

Взгляните еще раз, как программа `read-menu` из главы 28 обрабатывает выбор пользователя:

```
#!/bin/bash

# read-menu: программа вывода системной информации,
#            управляемая с помощью меню
```

```
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
```

С помощью case можно сделать логику выбора немного проще:

```
#!/bin/bash

# case-menu: программа вывода системной информации,
```

```
#            управляемая с помощью меню

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
```

Команда `case` берет значение *слова* — в данном примере значение переменной `REPLY` — и затем сопоставляет его с указанными *шаблонами*. Найдя соответствие, она выполняет команды, связанные с найденным шаблоном. После нахождения соответствия сопоставление с нижележащими шаблонами уже не производится.

Шаблоны

Шаблоны обрабатываются командой `case` точно так же, как пути механизмом подстановки. Шаблоны завершаются символом `)`. В табл. 31.1 перечислены некоторые допустимые шаблоны.

Таблица 31.1. Примеры шаблонов в команде case

Шаблон	Описание
a)	Соответствует, если *слово* содержит a
[[:alpha:]])	Соответствует, если *слово* содержит единственный алфавитный символ
???)	Соответствует, если *слово* содержит ровно три символа
*.txt)	Соответствует, если *слово* заканчивается символами .txt
*)	Соответствует любому значению *слова*. Считается хорошей практикой включать этот шаблон в команду case последним, чтобы перехватывать любые значения *слова*, не соответствующие ни одному из предыдущих шаблонов, то есть чтобы перехватывать любые недопустимые значения

Следующий пример демонстрирует работу шаблонов:

```
#!/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
```

Мы можем объединить несколько шаблонов, перечислив их через символ вертикальной черты. В результате получается комбинированный условный шаблон, объединенный по «ИЛИ». Эта возможность может пригодиться, например, для обработки символов верхнего и нижнего регистров:

```
#!/bin/bash

# case-menu: программа вывода системной информации,
#            управляемая с помощью меню

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
```

Здесь мы изменили программу `case-menu`, предложив пользователю выбирать пункты меню вводом букв, а не цифр. Обратите внимание, что новые шаблоны позволяют вводить буквы обоих регистров — верхнего и нижнего.

Выполнение нескольких вариантов

В версиях `bash` до 4.0 команда `case` могла выполнить только один вариант, соответствующий совпавшему шаблону. После этого команда завершалась. Рассмотрим сценарий, проверяющий введенный символ:

```
#!/bin/bash

# case4-1: проверка символа

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
```

Вот как выглядит результат выполнения этого сценария:

```
[me@linuxbox ~]$ case4-1
Type a character > a
'a' is lower case.
```

В большинстве случаев сценарий прекрасно справляется со своей задачей, но терпит неудачу, если символ соответствует нескольким символьным классам POSIX. Например, символ `a` соответствует классам алфавитных символов и символов нижнего регистра, а также шестнадцатеричных цифр. В `bash` до версии 4.0 не было никакой возможности заставить `case` выполнить больше одной успешной проверки. Современные версии `bash` поддерживают дополнительную нотацию `;;&` в конце каждого варианта, которая используется, как показано ниже:

```
#!/bin/bash

# case4-2: проверка символа

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
```

Запустив этот сценарий, мы получим:

```
[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.
```

Дополнительный синтаксис `;;&` позволяет команде `case` продолжить проверку вместо простого завершения после первого найденного совпадения.

Заключение

Команда `case` является удобным дополнением к нашей коллекции приемов программирования. Как будет показано в следующей главе, она отлично подходит для решения некоторых видов задач.