第5节 选择语句

1. if语句基本语法

单分支结构

第一种语法:

```
if <条件表达式>
  then
    指令
fi
```

第二种语法

```
if <条件表达式>;then
    指令
fi
```

上文的“ < 条件表达式 > ”部分可以是test、[]、[[]]、(())等条件表达式，甚至可以直接使用命令作为条件表达式。每个if条件语句都以if开头，并带有then，最后以fi结尾。

在所有编程语言里，if条件语句几乎是最简单的语句格式，且用途最广。当if后面的 < 条件表达式 > 成立时（真），就会执行then后面的指令或语句；否则，就会忽略then后面的指令或语句，转而执行fi下面的程序。

条件语句还可以嵌套（即if条件语句里面还有if条件语句），注意每个if条件语句中都要有一个与之对应的fi（if反过来写），每个if和它下面最近的fi成对搭配，语法示例如下：

```
if <条件表达式>
  then
     if <条件表达式>
        then
           指令
     fi
fi
```

通常在书写Shell条件语句编程时，要让成对的条件语句关键字的缩进相对应，以便于阅读浏览。

前文曾讲解过的文件条件表达式[-f "$file1"] && echo 1就等价于下面的if条件语句。

```
if [ -f "$file1" ];then
    echo 1
fi
```

双分支结构

if条件语句的双分支结构语法为：

```
if <条件表达式>
   then
     指令
else
   指令
fi
```

前文的文件测试条件表达式[-f "$file1"] && echo 1 || echo 0就相当于下面的双分支的if条件语句。

```
if [ -f "$file1" ]
   then
     echo 1
else
     echo 0
fi
```

多分支结构

if条件语句多分支语法为：

```
if <条件表达式>
   then
     指令
elif <条件表达式2>
   then
   指令
elif <条件表达式2>
   then
   指令
...
else
   指令
fi
```

- 注意多分支elif的写法，每个elif都要带有then。
- 最后结尾的else后面没有then。

2. read命令

Shell变量除了可以直接赋值或脚本传参外，还可以使用read命令从标准输入中获得，read为bash内置命令，可通过help read查看帮助。

语法格式：

```
read [参数] [变量名]
```

常用参数如下。

- -p prompt：设置提示信息。
- -t timeout：设置输入等待的时间，单位默认为秒。

实现read的基本读入功能。

```
read -t 10 -p "input a number:" num # 变量前需要有空格
echo $num
read -t 10 -p "input two number:" num1 num2 # 变量前需要有空格
echo $num1 $num2
```

案例实操，使用read读入数字比较两个整数的大小，创建脚本文件`if.sh`

```
read -p "input two number": a b
if [ $a -lt $b ]; then
    echo "$a < $b"
elif [ $a -eq $b ]; then
    echo "$a = $b"
else
    echo "$a > $b"
fi
```

3. case语句

case条件语句相当于多分支的if/elif/else条件语句，但是它比这些条件语句看起来更规范更工整

在case语句中，程序会将case获取的变量的值与表达式部分的值1、值2、值3等逐个进行比较，如果获取的变量值和某个值（例如值1）相匹配，就会执行值（例如值1）后面对应的指令（例如指令1，其可能是一组指令），直到执行到双分号（;;）才停止，然后再跳出case语句主体，执行case语句（即esac字符）后面的其他命令。

如果没有找到匹配变量的任何值，则执行“*）”后面的指令（通常是给使用者的使用提示），直到遇到双分号（;;）（此处的双分号可以省略）或esac结束，这部分相当于if多分支语句中最后的else语句部分。另外，case语句中表达式对应值的部分，还可以使用管道等更多功能来匹配。

case条件语句的语法格式为：

```
case "变量" in
    值1)
        指令..
        ;;
    值2)
        指令..
        ;;
    *)
        指令..
esac
```

当变量的值等于值1时，执行指令1；等于值2时执行指令2，以此类推；如果都不符合，则执行“*）”后面的指令，即指令3。

根据用户的输入判断用户输入的是哪个数字,如果是其他数字及字符，则返回输入不正确的提示并退出程序，编写case.sh脚本，内容如下：

```
read -p "input a number:" num
case "$num" in
    1)
        echo "the num is 1"
```

```
        ;;
    2)
        echo "the num is 2"
        ;;
    [3-9]) # 支持正则表达式
        echo "the num is $num"
        ;;
    *)
        echo "error"
esac
```