**Shell 流程控制**

和Java、PHP等语言不一样，sh的流程控制不可为空，如(以下为PHP流程控制写法)：

<?php

if (isset($\_GET["q"])) {

search(q);

}

else {

// 不做任何事情

}

在sh/bash里可不能这么写，如果else分支没有语句执行，就不要写这个else。

if else

if

if 语句语法格式：

if condition

then

command1

command2

...

commandN

fi

写成一行（适用于终端命令提示符）：

if [ $(ps -ef | grep -c "ssh") -gt 1 ]; then echo "true"; fi

末尾的fi就是if倒过来拼写，后面还会遇到类似的。

if else

if else 语法格式：

if condition

then

command1

command2

...

commandN

else

command

fi

if else-if else

if else-if else 语法格式：

if condition1

then

command1

elif condition2

then

command2

else

commandN

fi

以下实例判断两个变量是否相等：

a=10

b=20

if [ $a == $b ]

then

echo "a 等于 b"

elif [ $a -gt $b ]

then

echo "a 大于 b"

elif [ $a -lt $b ]

then

echo "a 小于 b"

else

echo "没有符合的条件"

fi

输出结果：

a 小于 b

if else语句经常与test命令结合使用，如下所示：

num1=$[2\*3]

num2=$[1+5]

if test $[num1] -eq $[num2]

then

echo '两个数字相等!'

else

echo '两个数字不相等!'

fi

输出结果：

两个数字相等!

for 循环

与其他编程语言类似，Shell支持for循环。

for循环一般格式为：

for var in item1 item2 ... itemN

do

command1

command2

...

commandN

done

写成一行：

for var in item1 item2 ... itemN; do command1; command2… done;

当变量值在列表里，for循环即执行一次所有命令，使用变量名获取列表中的当前取值。命令可为任何有效的shell命令和语句。in列表可以包含替换、字符串和文件名。

in列表是可选的，如果不用它，for循环使用命令行的位置参数。

例如，顺序输出当前列表中的数字：

for loop in 1 2 3 4 5

do

echo "The value is: $loop"

done

输出结果：

The value is: 1

The value is: 2

The value is: 3

The value is: 4

The value is: 5

顺序输出字符串中的字符：

for str in 'This is a string'

do

echo $str

done

输出结果：

This is a string

while 语句

while循环用于不断执行一系列命令，也用于从输入文件中读取数据；命令通常为测试条件。其格式为：

while condition

do

command

done

以下是一个基本的while循环，测试条件是：如果int小于等于5，那么条件返回真。int从0开始，每次循环处理时，int加1。运行上述脚本，返回数字1到5，然后终止。

#!/bin/bash

int=1

while(( $int<=5 ))

do

echo $int

let "int++"

done

运行脚本，输出：

1

2

3

4

5

使用中使用了 Bash let 命令，它用于执行一个或多个表达式，变量计算中不需要加上 $ 来表示变量，具体可查阅：Bash let 命令

。

while循环可用于读取键盘信息。下面的例子中，输入信息被设置为变量FILM，按<Ctrl-D>结束循环。

echo '按下 <CTRL-D> 退出'

echo -n '输入你最喜欢的网站名: '

while read FILM

do

echo "是的！$FILM 是一个好网站"

done

运行脚本，输出类似下面：

按下 <CTRL-D> 退出

输入你最喜欢的网站名:菜鸟教程

是的！菜鸟教程 是一个好网站

无限循环

无限循环语法格式：

while :

do

command

done

或者

while true

do

command

done

或者

for (( ; ; ))

until 循环

until 循环执行一系列命令直至条件为 true 时停止。

until 循环与 while 循环在处理方式上刚好相反。

一般 while 循环优于 until 循环，但在某些时候—也只是极少数情况下，until 循环更加有用。

until 语法格式:

until condition

do

command

done

condition 一般为条件表达式，如果返回值为 false，则继续执行循环体内的语句，否则跳出循环。

以下实例我们使用 until 命令来输出 0 ~ 9 的数字：

#!/bin/bash

a=0

until [ ! $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done

运行结果：

输出结果为：

0

1

2

3

4

5

6

7

8

9

case

Shell case语句为多选择语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

case 值 in

模式1)

command1

command2

...

commandN

;;

模式2）

command1

command2

...

commandN

;;

esac

case工作方式如上所示。取值后面必须为单词in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 \* 捕获该值，再执行后面的命令。

下面的脚本提示输入1到4，与每一种模式进行匹配：

echo '输入 1 到 4 之间的数字:'

echo '你输入的数字为:'

read aNum

case $aNum in

1) echo '你选择了 1'

;;

2) echo '你选择了 2'

;;

3) echo '你选择了 3'

;;

4) echo '你选择了 4'

;;

\*) echo '你没有输入 1 到 4 之间的数字'

;;

esac

输入不同的内容，会有不同的结果，例如：

输入 1 到 4 之间的数字:

你输入的数字为:

3

你选择了 3

跳出循环

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，Shell使用两个命令来实现该功能：break和continue。

break命令

break命令允许跳出所有循环（终止执行后面的所有循环）。

下面的例子中，脚本进入死循环直至用户输入数字大于5。要跳出这个循环，返回到shell提示符下，需要使用break命令。

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字:"

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的! 游戏结束"

break

;;

esac

done

执行以上代码，输出结果为：

输入 1 到 5 之间的数字:3

你输入的数字为 3!

输入 1 到 5 之间的数字:7

你输入的数字不是 1 到 5 之间的! 游戏结束

continue

continue命令与break命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。

对上面的例子进行修改：

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字: "

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的!"

continue

echo "游戏结束"

;;

esac

done

运行代码发现，当输入大于5的数字时，该例中的循环不会结束，语句 echo "游戏结束" 永远不会被执行。

esac

case的语法和C family语言差别很大，它需要一个esac（就是case反过来）作为结束标记，每个case分支用右圆括号，用两个分号表示break。