大讲台
dajiangtai.com

Shell 流程控制

主讲人：杨老师

Shell 流程控制---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。

Shell 流程控制---if else

● if else

if else 语法格式：

```
if condition
then
    command1
    command2
    ...
    commandN
else
    command
fi
```

Shell 流程控制---if else

● if else-if else

if else-if else 语法格式：

```
if condition1
then
    command1
elif condition2
then
    command2
else
    commandN
fi
```

Shell 流程控制---if else

● if else-if else 实例

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
```

Shell 流程控制---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。

Shell 流程控制---for 循环---实例

```
for loop in 1 2 3 4 5
do
    echo "The number is: $loop"
done
echo '----------------------------'
for str in 'This is a shell course'
do
    echo $str
done
echo '----------------------------'
for file in /home/hadoop/data/shell/*
do
    echo $file
done
```

Shell 流程控制---while 循环

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
```

Shell 流程控制---while 循环---实例

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

```
#!/bin/sh
n=1
while(( $n<=10 ))
do
        echo $n
        let "n++"
done
```

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

Shell 流程控制---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 (( ; ; ))
```

Shell 流程控制---until 循环

until循环执行一系列命令直至条件为真时停止。

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
```

条件可为任意测试条件，测试发生在循环末尾，因此循环至少执行一次—请注意这一点。

Shell 流程控制---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
    commandN
    ;;
模式2）
    command1
    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。

Shell 流程控制---case---实例

下面的脚本提示输入课程名称，与每一种模式进行匹配：

```
echo '输入你想学习的课程名称:'
echo '你输入的课程为:'
read course
case $course in
    hadoop）echo '你选择了hadoop课程'
    ;;
    storm）echo '你选择了storm课程'
    ;;
    spark）echo '你选择了spark课程'
    ;;
    shell）echo '你选择了shell课程'
    ;;
    *）echo '你输入的课程没有找到'
    ;;
esac
```

Shell 流程控制---跳出循环

在循环过程中，有时需要在未达到循环结束条件时强制跳出循环，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课程找不到要跳出循环，返回到shell提示符下，需要使用break命令。

```
#!/bin/bash
while :
do
    echo -n "输入你想学习的课程:"
    read course
    case $course in
        java|linux|hadoop|storm|spark) echo "你选择的课程为 $course!"
        ;;
        *) echo "你输入的课程没有找到!"
            break
        ;;
    esac
done
```

Shell 流程控制---跳出循环

● continue

continue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：

```
while :
do
    echo -n "输入你想学习的课程:"
    read course
    case $course in
        java|linux|hadoop|storm|spark) echo "你选择的课程为 $course!"
        ;;
        *) echo "你输入的课程没有找到!"
            continue
            echo "退出选课"
        ;;
    esac
done
```

运行代码发现，当输入课程不存在时，该例中的循环不会结束，语句 echo "退出选课" 永远不会被执行。

大讲台
dajiangtai.com

THANKS

黄丽老师：3354223855

小夏老师：972628726