8 for循环语句

`for` 循环语句

在日常的Linux运维工作中，经常需要**反复执行相同的代码**。

在Shell脚本中就可以通过循环语句实现重复执行特定代码块的功能。

Shell脚本支持的循环语句包括 `for` 、 `while` 、 `until` 和 `select` 。

一、 `for` 循环语法结构

`for` 循环是Shell脚本中最常见的循环结构，主要用于执行**固定次数**的循环。

`for` 循环是一种**运行前测试**语句，也就是在运行任何循环体之前**先要判断循环条件是否成立，只有在成立的情况下才会运行循环体，否则将退出循环。**每完成一次循环后，**在进行下一次循环之前都会再次进行测试。**

`for` 循环语句根据书写习惯又分为**带列表**的 `for` 循环和**C语言风格**的 `for` 循环。

1.1 带列表的 `for` 循环

带列表的 `for` 循环用于执行**固定次数**的循环（**循环次数等于列表元素个数**），其语法结构如下：

Shell | 复制代码

```
for  临时变量 in 取值列表
do
   循环体
done
```

在 `for` 循环语句中， `for` 关键字后面会有一个 `临时变量` ， `临时变量` 依次获取 `in` 关键字后面的 `取值列表` 内容(**默认分隔符为空格、Tab或换行**)，**每次**仅取**一个值**，然后进入循环(`do` 和 `done` 之间的部分)执行**循环体**内的**命令序列**，当执行到 `done` 时结束本次循环。之后**临时变量**再继续获取 `取值列表` 里的**下一个值**，继续执行循环体内的命令序列，当执行到 `done` 时结束并返回。以此类推，直到获取变量列表里的最后一个值，并进入循环执行到 `done` 结束为止。

案例1：简易计数for循环

```shell
[root@Shell ~]# vi for1.sh
#!/bin/bash
for i in 1 2 3 4 5
do
    echo "The value is ${i}"
done
[root@Shell ~]# . for1.sh
The value is 1
The value is 2
The value is 3
The value is 4
The value is 5
```

上面脚本的写法并不是最好的，因为一旦列表元素改变了，你就不得不去改相应的 `for` 循环语句块。好的习惯是**将列表定义为一个变量**，然后在 `for` 语句中使用该变量。

```shell
[root@Shell ~]# vi for2.sh
#!/bin/bash
lists="1 2 3 4 5"
for i in ${lists}
do
    echo "The value is ${i}"
done
[root@Shell ~]# . for2.sh
The value is 1
The value is 2
The value is 3
The value is 4
The value is 5
```

前面的案例中需要手动生成列表，比较繁琐，可以利用 `{n..m}` **生成数字序列**。

```shell
[root@Shell ~]# echo {1..5}
1 2 3 4 5
[root@Shell ~]# vi for3.sh
#!/bin/bash
for i in {1..5}
do
    echo "The value is ${i}"
done
[root@Shell ~]# . for3.sh
The value is 1
The value is 2
The value is 3
The value is 4
The value is 5
```

`for` 循环结构中的 `取值列表` **可以使用** `命令替换` **生成。**

案例2：使用命令替换生成数值列表

`seq` 命令可生成一个**数值序列**。

```shell
seq 终止值
seq 起始值 终止值
seq 起始值 步长 终止值
```

```shell
[root@Shell ~]# seq 1 5
1
2
3
4
5
[root@Shell ~]# seq 1 2 5
1
3
5
[root@Shell ~]# vi for4.sh
#!/bin/bash
for i in $(seq 1 5)
do
    echo "The value is ${i}"
done
[root@Shell ~]# . for4.sh
The value is 1
The value is 2
The value is 3
The value is 4
The value is 5
```

案例3：使用命令替换生成文件名列表

获取 `/etc/yum.repos.d` 目录下的文件的名称，并将其作为变量列表打印输出。

```shell
[root@Shell ~]# ls /etc/yum.repos.d/
opt_soft_.repo
[root@Shell ~]# vi for5.sh
#!/bin/bash
# for file in $(ls /etc/yum.repos.d/)
for file in `ls /etc/yum.repos.d/`
do
    echo "The filename is ${file}"
done
[root@Shell ~]# . for5.sh
The filename is opt_soft_.repo
```

> 💬 *`for` 循环语句可以写成一行语句 `for 变量名 in 取值列表;do循环体;done` 。*

1.2 C语言风格的for循环

Shell支持类C语言风格的`for`循环。了解C语言或类C语言的读者一定会对`(i=1; i<=10; i++)`这样的结构十分熟悉，Shell中类似语法结构如下。

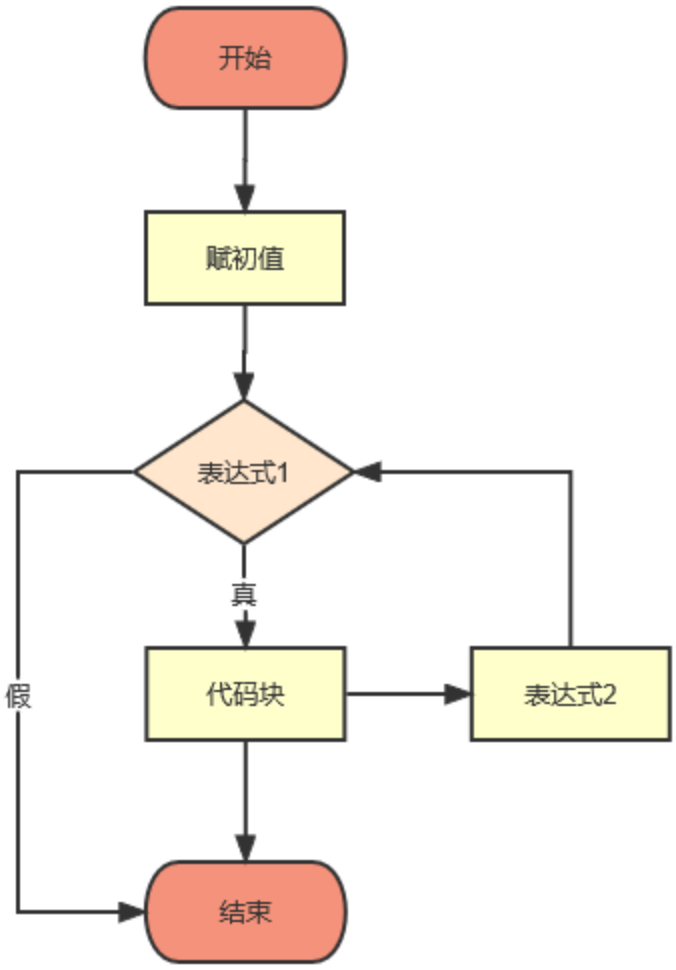
Shell 复制代码

```shell
for  ((赋初值,表达式1,表达式2))
do
   循环体
done
```

`for`关键字后的**双括号**内是三个表达式。

- 赋初值表达式作用为**变量初始化赋值**（例如：`i=0`）
- 表达式1作用为**循环判断条件**（例如：`i<10`）
- 表达式2作用为**变量自增或自减**（例如：`i++`）

当**表达式1条件成立**时，就执行**循环体**和**表达式2**；条件**不成立**时就**退出循环**。



案例3：将案例1改写为C语言风格

```shell
[root@Shell ~]# vi for6.sh
#!/bin/bash
for ((i=1;i<=5;i++))
do
    echo "The value is ${i}"
done
[root@Shell ~]# . for6.sh
The value is 1
The value is 2
The value is 3
The value is 4
The value is 5
```

二、`for`循环语句实战案例

2.1 批量主机`ping`探测

在生产环境中，查看主机是否为存活状态很重要，当主机数量较多时，手动查看主机状态，不仅工作量大，而且工作效率很低。这时就需要编写一个实现批量主机探测的脚本。

案例4：输出存活主机IP

构造主机IP列表，对每个IP都`ping`一次，如果返回值为0，则屏幕输出IP地址。

Shell | 复制代码

```
[root@Shell ~]# vi for_ping1.sh
#!/bin/bash
for i in {1..5}
do
    ip=192.168.149.${i}
    # ping主机，结果不在屏幕输出
    ping -c1 -W1 $ip &>/dev/null
    # 如果ping通输出ip
    if [ $? -eq 0 ];then
        echo "$ip"
    fi
done
echo "finished..."
[root@Shell ~]# . for_ping1.sh
192.168.149.1
192.168.149.2
192.168.149.3
finished...
```

进一步优化，将存活主机IP地址保存到 `ip.txt` 文件中。

```shell
[root@Shell ~]# vi for_ping2.sh
#!/bin/bash
#先清空 ip.txt 文件内容
>ip.txt
#tee命令用于多重重定向（标准输出、文件）
for i in {1..5}
do
    ip=192.168.149.${i}
    ping -c1 -W1 $ip &>/dev/null
    if [ $? -eq 0 ]; then
        echo "$ip" | tee -a ip.txt
    fi
done
echo "finished..."
[root@Shell ~]# . for_ping2.sh
192.168.149.1
192.168.149.2
192.168.149.3
finished...
[root@Shell ~]# cat ip.txt
192.168.149.1
192.168.149.2
192.168.149.3
```

2.2 批量用户创建

批量创建用户在运维工作需求中也是很常见的。

案例5：批量创建用户

根据输入模式（前缀 密码 编号）批量创建用户。

```shell
[root@Shell ~]# vi create_users.sh
#!/bin/bash
read -p "Please enter prefix & password &num[yan 123 2]:" prefix pass num
# seq -w等长
for i in $(seq -w $num); do
    # 构造用户名
    user=$prefix$i
    id $user &>/dev/null
    # 检测用户是否存在
    if [ $? -eq 0 ]; then
        echo "user $user already exists"
    else
        useradd $user
        echo "$pass" | passwd --stdin "$user" &>/dev/null
        # 检测密码是否设置成功
        if [ $? -eq 0 ]; then
            echo "$user" is created.
        fi
    fi
done
[root@Shell ~]# . create_users.sh
Please enter prefix & password &num[yan 123 2]:yan 123 2
yan1 is created.
yan2 is created.
# 切换用户验证
[root@Shell ~]# su yan1
[yan1@Shell root]$ su yan2
Password:
# 删除用户
[yan2@Shell root]$ exit
exit
[yan1@Shell root]$ exit
exit
[root@Shell ~]# userdel -r yan1
[root@Shell ~]# userdel -r yan2
```

2.3 通过文件批量创建用户和删除用户（扩展）

案例6：根据文本文件批量创建和删除用户

根据文件批量用户操作的原理是先把批量用户和密码放在某一个文本文件中，然后写 `for` 循环语句调用这个文件。文本文件中的用户和密码如下。

Shell 复制代码

```
[root@Shell ~]# vi user1.txt
abcd 123
qwer 123
```

Shell 复制代码

```
# 批量创建用户
[root@Shell ~]# vi create_users_by_txt.sh
#!/bin/bash
#考虑一种特殊情况，如果变量是空行，其解决方法是重新定义分隔符
#希望 for 处理文件按回车分隔，而不是空格或 tab 空格
#重新定义分隔符，IFS为内部字段分隔符
IFS=$'\n'
#判断参数是否是文件，如果是文件继续进行处理
if [ ! -f $1 ]; then
  echo "error file"
else
    #根据每行的信息迭代创建建用户和密码
    for line in $(cat $1); do
        # echo ${line}
        #如果是空行退出循环
        if [ ${#line} -eq 0 ];then
            echo "Nothing to do"
            continue
        fi
        #提取用户和密码信息
        user=$(echo "$line" | awk '{print $1}')
        pass=$(echo "$line" | awk '{print $2}')
        # echo ${user} ${pass}
        #创建用户
        if id $user &>/dev/null;then
            echo "user $user already exists"
        else
            useradd $user
            echo "$pass" | passwd --stdin "$user" &>/dev/null
            if [ $? -eq 0 ]; then
                echo "$user is created."
            fi
        fi
    done
fi
[root@Shell ~]# . create_users_by_txt.sh user1.txt
abcd is created.
qwer is created.
```

```shell
# 根据文件删除用户
[root@Shell ~]# vi del_users.sh
#!/bin/bash
IFS=$'\n'
for line in $(cat $1)
do
    user=$(echo "$line" |awk '{print $1}')
    if id -u "${user}" &>/dev/null ;then
        userdel -r "${user}"
        echo "user $user deleted"
    fi
done
[root@Shell ~]# . del_users.sh user1.txt
user abcd deleted
user qwer deleted
```

小结

- `for` 循环格式
- `for` 循环应用

课程目标

- 知识目标：熟练掌握for语句的基本语法。
- 技能目标：能够根据实际需求利用for语句实现流程控制。

课外拓展

- 进一步了解for语句的应用场景。

参考资料

- `help for`
- 《Linux Shell核心编程指南》，丁明一，电子工业出版社