NSD SHELL DAY02

1 案例1：条件测试操作

1.1 问题

本案例要求参考PPT上的示例，分别练习以下条件测试操作：

- 字符串匹配
- 比较整数值的大小
- 识别文件/目录的状态
- 多个条件/操作的逻辑组合

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：条件测试的基本用法

1）语法格式

使用“test 表达式”或者[表达式]都可以，表达式两边至少要留一个空格。

条件测试操作本身不显示出任何信息。测试的条件是否成立主要体现在命令执行后的返回状态(即 $?），所以可以在测试后查看变量$?的值来做出判断，或者结合&&、||等逻辑操作显示出结果(或作其他操作）。

步骤二：字符串测试

1）== 比较两个字符串是否相同

检查当前用户是否为root。

当root用户执行时：

```
[root@svr5 ~]# [ $USER == "root" ]        //测试
[root@svr5 ~]# echo $?                    //查看结果0为对，非0为错
```

当普通用户执行时：

```
[zengye@svr5 ~]$ [ $USER == "root" ]
[zengye@svr5 ~]$ echo $?                  //查看结果0为对，非0为错
```

Top

2）!= 比较两个字符串是否不相同

当普通用户执行时：

```
01.  [zengye@svr5 ~]$ [ $USER != "root" ]
```

当root用户执行时：

```
01.  [root@svr5 ~]# [ $USER != "root" ]
```

3）一行执行多条命令的情况

```
01.  # A && B                  //仅当A命令执行成功，才执行B命令
02.  # A || B                  //仅当A命令执行失败，才执行B命令
03.  # A ; B                   //执行A命令后执行B命令，两者没有逻辑关系
04.  # A && B || C             //思考？
```

4) -z 检查变量的值是否未设置（空值）

```
01.  [root@svr5 ~]# var1="nb" ; var2=""
02.  [root@svr5 ~]# [ -z "$var1" ] && echo "空值" || echo "非空值"
03.  非空值
04.  [root@svr5 ~]# [ -z $var2 ] && echo "空值" || echo "非空值"
05.  空值                              //变量var2已设置，但无任何值，视为空
06.  [root@svr5 ~]# [ ! -z $var1 ]          //测试var1是否为非空
```

还有一个-n可以测试变量是否不为空（相当于! -z）。

步骤三：整数值比较

参与比较的必须是整数（可以调用变量），比较非整数值时会出错：

```
01.  [root@svr5 ~]# A=20.4
02.  [root@svr5 ~]# [ $A -gt 10 ]           //不支持小数比较
03.  -bash: [: 20.4: integer expression expected
```

1）-eq 比较两个数是否相等。

Top

```
01.  [root@svr5 ~]# X=20                    //定义一个测试变量
```

```
[root@svr5 ~]# [ $X -eq 20 ] && echo "相等" || echo "不相等"
相等
[root@svr5 ~]# [ $X -eq 30 ] && echo "相等" || echo "不相等"
不相等
```

2）-ne 比较两个数是否不相等。

```
[root@svr5 ~]# X=20                    //定义一个测试变量
[root@svr5 ~]# [ $X -ne 20 ] && echo "不等于" || echo "等于"
等于
[root@svr5 ~]# [ $X -ne 30 ] && echo "不等于" || echo "等于"
不等于
```

3）-gt 比较前面的整数是否大于后面的整数。

```
[root@svr5 ~]# X=20                    //定义一个测试变量
[root@svr5 ~]# [ $X -gt 10 ] && echo "大于" || echo "否"
大于
[root@svr5 ~]# [ $X -gt 20 ] && echo "大于" || echo "否"
否
[root@svr5 ~]# [ $X -gt 30 ] && echo "大于" || echo "否"
否
```

4）-ge 比较前面的整数是否大于或等于后面的整数。

```
[root@svr5 ~]# X=20                    //定义一个测试变量
[root@svr5 ~]# [ $X -ge 10 ] && echo "大于或等于" || echo "否"
大于或等于
[root@svr5 ~]# [ $X -ge 20 ] && echo "大于或等于" || echo "否"
大于或等于
[root@svr5 ~]# [ $X -ge 30 ] && echo "大于或等于" || echo "否"
否
```

5）-lt 比较前面的整数是否小于后面的整数。

Top

```
[root@svr5 ~]# X=20                    //定义一个测试变量
```

```
02.  [root@svr5 ~]# [ $X -lt 10 ] && echo "小于" || echo "否"
03.  否
04.  [root@svr5 ~]# [ $X -lt 20 ] && echo "小于" || echo "否"
05.  否
06.  [root@svr5 ~]# [ $X -lt 30 ] && echo "小于" || echo "否"
07.  小于
```

6）-le 比较前面的整数是否小于或等于后面的整数。

```
01.  [root@svr5 ~]# X=20                          //定义一个测试变量
02.  [root@svr5 ~]# [ $X -le 10 ] && echo "小于或等于" || echo "否"
03.  否
04.  [root@svr5 ~]# [ $X -le 20 ] && echo "小于或等于" || echo "否"
05.  小于或等于
06.  [root@svr5 ~]# [ $X -le 30 ] && echo "小于或等于" || echo "否"
07.  小于或等于
```

7）提取当前登录的用户数，比较是否大于等于3。

```
01.  [root@svr5 ~]# who | wc -l                          //确认已登录的用户数
02.  2
03.  [root@svr5 ~]# N=$(who | wc -l)                      //赋值给变量N
04.  [root@svr5 ~]# [ $N -ge 3 ] && echo "超过了" || echo "没超过"
05.  没超过
```

上述赋值给变量N及与3比较的操作，可以简化为如下形式：

```
01.  [root@svr5 ~]# [ $(who | wc -l) -ge 3 ] && echo "超过了" || echo "没超过"
02.  没超过
```

步骤四：识别文件/目录的状态

1）-e 判断对象是否存在（不管是目录还是文件）

```
01.  [root@svr5 ~]# [ -e "/usr/" ] && echo "存在" || echo "不存在"
02.  存在
03.  [root@svr5 ~]# [ -e "/etc/fstab" ] && echo "存在" || echo "不存在"
```

Top

```
存在
[root@svr5 ~]# [ -e "/home/nooby" ] && echo "存在" || echo "不存在"
不存在
```

2）-d 判断对象是否为目录（存在且是目录）

```
[root@svr5 ~]# [ -d "/usr/" ] && echo "是目录" || echo "不是目录"
是目录
[root@svr5 ~]# [ -d "/etc/fstab" ] && echo "是目录" || echo "不是目录"
不是目录
[root@svr5 ~]# [ -d "/home/nooby" ] && echo "是目录" || echo "不是目录"
不是目录
```

3）-f 判断对象是否为文件（存在且是文件）

```
[root@svr5 ~]# [ -f "/usr/" ] && echo "是文件" || echo "不是文件"
不是文件
[root@svr5 ~]# [ -f "/etc/fstab" ] && echo "是文件" || echo "不是文件"
是文件
[root@svr5 ~]# [ -f "/home/nooby" ] && echo "是文件" || echo "不是文件"
不是文件
```

4）-r 判断对象是否可读

此测试对root用户无效，无论文件是否设置r权限，root都可读：

```
[root@svr5 ~]# cp /etc/hosts /tmp/test.txt          //复制一个文件做测试
[root@svr5 ~]# chmod -r /tmp/test.txt               //去掉所有的r权限
[root@svr5 ~]# [ -r "/tmp/test.txt" ] && echo "可读" || echo "不可读"
可读                                                //root测试结果仍然可读
```

切换为普通用户，再执行相同的测试，结果变为“不可读”：

```
[zengye@svr5 ~]$ [ -r "/tmp/test.txt" ] && echo "可读" || echo "不可读"
不可读
```

Top

5）-w 判断对象是否可写

此测试同样对root用户无效，无论文件是否设置w权限，root都可写：

```
[root@svr5 ~]# chmod -w /tmp/test.txt          //去掉所有的w权限
[root@svr5 ~]# ls -l /tmp/test.txt             //确认设置结果
---------- 1 root root 33139 12-11 10:43 /tmp/test.txt
[root@svr5 ~]# [ -w "/tmp/test.txt" ] && echo "可写" || echo "不可写"
可写
```

切换为普通用户，可以正常使用-w测试：

```
[zengye@svr5 ~]$ ls -l /tmp/test.txt
---------- 1 root root 33139 12-11 10:52 /tmp/test.txt
[zengye@svr5 ~]$ [ -w "/tmp/test.txt" ] && echo "可写" || echo "不可写"
不可写
```

6）-x 判断对象是否具有可执行权限

这个取决于文件本身、文件系统级的控制，root或普通用户都适用：

```
[root@svr5 ~]# chmod 644 /tmp/test.txt         //重设权限，无x
[root@svr5 ~]# ls -l /tmp/test.txt             //确认设置结果
-rw-r--r-- 1 root root 33139 12-11 10:52 /tmp/test.txt
[root@svr5 ~]# [ -x "/tmp/test.txt" ] && echo "可执行" || echo "不可执行"
不可执行
[root@svr5 ~]# chmod +x /tmp/test.txt          //添加x权限
[root@svr5 ~]# [ -x "/tmp/test.txt" ] && echo "可执行" || echo "不可执行"
可执行
```

步骤五：多个条件/操作的逻辑组合

1）&&，逻辑与

给定条件必须都成立，整个测试结果才为真。

检查变量X的值是否大于10，且小于30：

```
[root@svr5 ~]# X=20                    //设置X变量的值为20
[root@svr5 ~]# [ $X -gt 10 ] && [ $X -lt 30 ] && echo "YES"
YES
```

Top

2）||，逻辑或

只要其中一个条件成立，则整个测试结果为真。

只要/tmp/、/var/spool/目录中有一个可写，则条件成立：

```
[root@svr5 ~]# [ -w "/tmp/" ] || [ -w "/var/spool/" ] && echo "OK"
OK
```

2 案例2：使用if选择结构

2.1 问题

本案例要求编写3个Shell脚本，分别实现以下目标：

- 检测/media/cdrom目录，若不存在则创建
- 检测并判断指定的主机是否可ping通
- 从键盘读取一个论坛积分，判断论坛用户等级，等级分类如下:

大于等于90 神功绝世

大于等于80，小于90 登峰造极

大于等于70，小于80 炉火纯青

大于等于60，小于70 略有小成

小于60 初学乍练

2.2 方案

if单分支的语法组成：

```
if 条件测试
then
命令序列
fi
```

if双分支的语法组成：

```
if 条件测试
then
命令序列1
else
命令序列2
fi
```

Top

if多分支的语法组成：

```
if    条件测试1 ;then
命令序列1
elif  条件测试2 ;then
命令序列2
else
命令序列n
fi
```

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：检测/media/cdrom目录，若不存在则创建

1）编写脚本如下：

```
[root@svr5 ~]# vim mountdir.sh
#!/bin/bash
dir="/media/cdrom/"
if [ ! -d $dir ]
then
    mkdir -p $dir
fi

[root@svr5 ~]# chmod +x mountdir.sh              //添加可执行权限
```

2）测试、验证脚本功能

```
[root@svr5 ~]# ls -ld /media/cdrom               //本来没有/media/cdrom目录
ls: /media/cdrom: 没有那个文件或目录
[root@svr5 ~]# ./mountdir.sh                     //执行脚本
[root@svr5 ~]# ls -ld /media/cdrom               //再检查已经有了
drwxr-xr-x 2 root root 4096 12-11 15:16 /media/cdrom
```

有了/media/cdrom文件夹以后，再次执行上述脚本，实际上不做任何有效操作：

```
[root@svr5 ~]# ./mountdir.sh
```

Top

步骤二：检测并判断指定的主机是否可ping通

1）分析任务需求

使用ping命令检测目标主机时，人工可直接判断反馈结果，而脚本却不方便。但是当ping测试成功时，执行状态$?的值为0；而ping测试失败时，$?的值不为0。因此在Shell脚本中可以利用这一点来判断ping目标主机的成败。

为了节省ping测试时间，可以只发送3个测试包（-c 3）、缩短发送测试包的间隔秒数（-i 0.2）、等待反馈的超时秒数（-W 1）。比如，检查可ping通的主机：

```
[root@svr5 ~]# ping -c 3 -i 0.2 -W 1 192.168.4.5
PING 192.168.4.5 (192.168.4.5) 56(84) bytes of data.
64 bytes from 192.168.4.5: icmp_seq=1 ttl=64 time=0.131 ms
64 bytes from 192.168.4.5: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 192.168.4.5: icmp_seq=3 ttl=64 time=0.073 ms

--- 192.168.4.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 402ms
rtt min/avg/max/mdev = 0.073/0.093/0.131/0.027 ms

[root@svr5 ~]# echo $?                          //执行状态表示成功
0
```

2）脚本编写参考如下：

```
[root@svr5 ~]# vim pinghost.sh
#!/bin/bash
ping -c 3 -i 0.2 -W 1 $1 &> /dev/null
if [ $? -eq 0 ] ; then
    echo "Host $1 is up."
else
    echo "Host $1 is down."
fi

[root@svr5 ~]# chmod +x pinghost.sh
```

3）测试、验证脚本功能

```
[root@svr5 ~]# ./pinghost.sh 192.168.4.5
Host 192.168.4.5 is up.
[root@svr5 ~]# ./pinghost.sh 192.168.4.50
```

Top

```
04.  Host 192.168.4.50 is down.
```

步骤三：从键盘读取一个论坛积分，判断论坛用户等级

1）脚本编写参考如下：

大于等于90 神功绝世

大于等于80，小于90 登峰造极

大于等于70，小于80 炉火纯青

大于等于60，小于70 略有小成

小于60 初学乍练

```
01.  [root@svr5 ~]# vim grade.sh
02.  #!/bin/bash
03.  read -p "请输入积分（0-100）：" JF
04.  if [ $JF -ge 90 ] ; then
05.      echo "$JF 分，神功绝世"
06.  elif [ $JF -ge 80 ] ; then
07.      echo "$JF 分，登峰造极"
08.  elif [ $JF -ge 70 ] ; then
09.      echo "$JF 分，炉火纯青"
10.  elif [ $JF -ge 60 ] ; then
11.      echo "$JF 分，略有小成"
12.  else
13.      echo "$JF 分，初学乍练"
14.  fi
15.
16.  [root@svr5 ~]# chmod +x grade.sh
```

3）测试、验证脚本

```
01.  [root@svr5 ~]# ./grade.sh
02.  请输入积分（0-100）：74
03.  74 分，炉火纯青
04.  [root@svr5 ~]# ./grade.sh
05.  请输入分数（0-100）：68
06.  68 分，略有小成
07.  [root@svr5 ~]# ./grade.sh
08.  请输入分数（0-100）：87
09.  87 分，登峰造极
```

Top

3 案例3：使用for循环结构

3.1 问题

本案例要求编写一个Shell脚本chkhosts.sh，利用for循环来检测多个主机的存活状态，相关要求及说明如下：

- 对192.168.4.0/24网段执行ping检测
- 脚本能遍历ping各主机，并反馈存活状态

执行检测脚本以后，反馈结果如图-1所示。

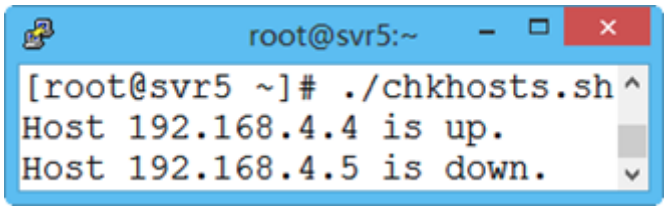

图-1

3.2 方案

在Shell脚本应用中，常见的for循环采用遍历式、列表式的执行流程，通过指定变量从值列表中循环赋值，每次复制后执行固定的一组操作。

for循环的语法结构如下所示：

```
for 变量名 in 值列表
do
   命令序列
done

for 变量名 in {1..5}
do
   命令序列
done
for 变量名 in `seq 5`
do
   命令序列
done

for 变量名 in `ls /etc/*.conf`
do
   命令序列
done
```

Top

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：练习for循环基本用法

脚本1，通过循环批量显示5个hello world：

```
[root@svr5 ~]# vim for01.sh
#!/bin/bash
for i in 1 2 3 4 5
do
        echo "hello world"
done
[root@svr5 ~]# chmod +x for01.sh
[root@svr5 ~]# ./for01.sh
```

脚本2，通过循环批量显示10个hello world：

```
[root@svr5 ~]# vim for02.sh
#!/bin/bash
for i in {1..10}
do
        echo "hello world"
done
[root@svr5 ~]# chmod +x for02.sh
[root@svr5 ~]# ./for02.sh
```

脚本3，通过循环批量显示10个数字：

```
[root@svr5 ~]# vim for03.sh
#!/bin/bash
for i in {1..10}
do
        echo "$i"
done
[root@svr5 ~]# chmod +x for03.sh
[root@svr5 ~]# ./for03.sh
```

Top

步骤二：批量检测多个主机的存活状态

1）编写脚本如下：

命令备注：ping命令可以测试某台主机的连通性，
使用-c选项可以设置ping的次数，
使用-i选项可以设置多次ping之间的间隔时间（单位秒），
使用-W选项可以设置ping不通时的超时时间（单位秒）。

```
[root@svr5 ~]# vim chkhosts.sh
#!/bin/bash
for i in {1..254}
do
    ping -c 3 -i 0.2 -W 1 192.168.4.$i &> /dev/null
    if [ $? -eq 0 ] ; then
        echo "Host 192.168.4.$i is up."
    else
        echo "Host 192.168.4.$i is down."
    fi
done
[root@svr5 ~]# chmod +x chkhosts.sh
```

4）测试、验证脚本

```
…… ……
[root@svr5 ~]# ./chkhosts.sh
Host 192.168.4.5 is up.
Host 192.168.4.6 is down
…… ……
```

步骤三：创建账户的案例

创建users.txt，写入无规律的账户名称，最后使用for循环读取该文件，批量创建账户并设置密码。

```
[root@svr5 ~]# vim addfor.sh
#!/bin/bash
for i in `cat /root/user.txt`
do
    useradd $i
    echo 123456 | passwd --stdin $i
done
```

Top

附加扩展知识（C语言风格的for循环语法格式）

```
[root@svr5 ~]# vim cfor.sh
#!/bin/bash
for ((i=1;i<=5;i++))
do
   echo $i
done
```

4 案例4：使用while循环结构

4.1 问题

本案例要求编写2个使用while循环的脚本程序，分别实现以下目标：

- 提示用户猜测一个随机数，直到才对为止
- 检测192.168.4.0/24网段，列出不在线的主机地址

4.2 方案

while循环属于条件式的执行流程，会反复判断指定的测试条件，只要条件成立即执行固定的一组操作，直到条件变化为不成立为止。所以while循环的条件一般通过变量来进行控制，在循环体内对变量值做相应改变，以便在适当的时候退出，避免陷入死循环。

while循环的语法结构如下所示：

```
while  条件测试
do
    命令序列
done


while :
do
    命令序列
done
```

4.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：练习while循环基本用法

脚本1，无心的死循环脚本：

Top

```
[root@svr5 ~]# vim while01.sh
#!/bin/bash
i=1
while [ $i -le 5 ]
do
        echo "$i"
done
[root@svr5 ~]# chmod +x while01.sh
[root@svr5 ~]# ./while01.sh              //死循环，需要使用Ctrl+C终止脚本
```

脚本2，有效循环脚本：

```
[root@svr5 ~]# vim while02.sh
#!/bin/bash
i=1
while [ $i -le 5 ]
do
        echo "$i"
        let i++
done
[root@svr5 ~]# chmod +x while02.sh
[root@svr5 ~]# ./while02.sh
```

脚本3，死循环的一般格式：

```
[root@svr5 ~]# vim while03.sh
#!/bin/bash
while :
do
        echo "hello world"
done
[root@svr5 ~]# chmod +x while03.sh
[root@svr5 ~]# ./while03.sh              //死循环，需要使用Ctrl+C终止脚本
```

步骤二：提示用户猜测一个随机数，直到才对为止

Top

使用系统自带变量RANDOM提取随机数（1-100），使用while :制作死循环。

脚本编写参考如下：

```
[root@svr5 ~]# vim guess.sh
#!/bin/bash
num=$[RANDOM%100+1]
i=0
while :
do
    read -p "随机数1-100,你猜:" guess
    let i++                          //猜一次，计数器加1，统计猜的次数
    if [ $guess -eq $num ];then
        echo "恭喜，猜对了"
        echo "你猜了$i次"
        exit
    elif [ $guess -gt $num ];then
        echo "猜大了"
    else
        echo "猜小了"
    fi
[root@svr5 ~]# chmod +x guess.sh
```

执行脚本并验证结果：

```
[root@svr5 ~]# ./guess.sh
```

步骤三：检测192.168.4.0/24网段，列出不在线的主机地址

1）任务需求及思路分析

要求的是“检测192.168.4.0/24网段，列出不在线的主机地址”。

检测目标是一个网段，其网络部分“192.168.4.”可以作为固定的前缀；而主机部分包括从1~254连续的地址，所以可结合while循环和自增变量进行控制。

2）根据实现思路编写脚本

```
[root@svr5 ~]# vim chknet.sh
#!/bin/bash
i=1
while [ $i -le 254 ]
do
    IP="192.168.4.$i"
    ping -c 3 -i 0.2 -W 1 $IP &> /dev/null
    if [ $? -eq 0 ] ; then
```

Top

```
09.             echo "Host $IP is up."
10.         else
11.             echo "Host $IP is down."
12.         fi
13.         let i++
14.     done
15.     [root@svr5 ~]# chmod +x chknet.sh
```

3）测试、验证脚本

```
01.     [root@svr5 ~]# ./chknet.sh
02.     Host 192.168.4.1 is down.
03.     Host 192.168.4.2 is down.
04.     Host 192.168.4.3 is down.
05.     Host 192.168.4.4 is down.
06.     Host 192.168.4.5 is up.
07.     .. ..
08.     Host 192.168.4.250 is down.
09.     Host 192.168.4.251 is down.
10.     Host 192.168.4.252 is down.
11.     Host 192.168.4.253 is down.
12.     Host 192.168.4.254 is down.
```

Top