

# NSD SHELL DAY02

1. [案例1：Shell中的数值运算](#)
2. [案例2：条件测试操作](#)
3. [案例3：使用if选择结构](#)

## 1 案例1：Shell中的数值运算

### 1.1 问题

本案例要求熟悉Linux Shell环境的特点，主要练习以下操作：

- 使用expr、\$[ ]、let等整数运算工具：定义变量X=1234，然后计算X与78的四则运算及求模结果
- 使用bc实现小数运算操作：以交互方式计算12.34与56.78的四则运算结果，另外再以非交互方式重复上述计算，最多显示4位小数

### 1.2 步骤

实现此案例需要按照如下步骤进行。

#### 步骤一：整数运算工具

##### 1) 使用expr命令

乘法操作应采用\\*转义，避免被作为Shell通配符；参与运算的整数值与运算操作符之间需要以空格分开，引用变量时必须加\$符号。

首先定义变量X=1234，然后分别计算与78的加减乘除和求模运算结果：

```

01. [ root@svr5 ~] # X=1234                //定义变量X
02. [ root@svr5 ~] # expr $X + 78          //加法
03. 1312
04. [ root@svr5 ~] # expr $X - 78          //减法
05. 1156
06. [ root@svr5 ~] # expr $X \* 78         //乘法，操作符应添加\转义
07. 96252
08. [ root@svr5 ~] # expr $X / 78          //除法，仅保留整除结果
09. 15
10. [ root@svr5 ~] # expr $X % 78          //求模
11. 64

```

##### 2) 使用\$[]或\$(())表达式

乘法操作\*无需转义，运算符两侧可以无空格；引用变量可省略\$符号；计算结果替换表达式本身，可结合echo命令输出。

同样对于变量X=1234，分别计算与78的加减乘除和求模运算结果：

```

01. [ root@svr5 ~] # X=1234
02. [ root@svr5 ~] # echo $[ X+78]
03. 1312
04. [ root@svr5 ~] # echo $[ X- 78]
05. 1156
06. [ root@svr5 ~] # echo $[ X*78]
07. 96252
08. [ root@svr5 ~] # echo $[ X/78]
09. 15
10. [ root@svr5 ~] # echo $[ X%78]
11. 64

```

### 3 ) 使用let命令

expr或\$[], \$(())方式只进行运算，并不会改变变量的值；而let命令可以直接对变量值做运算再保存新的值。因此变量X=1234，在执行let运算后的值会变更；另外，let运算操作并不显示结果，但是可以结合echo命令来查看：

```

01. [ root@svr5 ~] # X=1234
02. [ root@svr5 ~] # let y=X+22
03. [ root@svr5 ~] # echo $y
04. 1256
05. [ root@svr5 ~] # let X++; echo $X      # X++( X=X+1)
06. [ root@svr5 ~] # let X-- ; echo $X     # X-- ( X=X- 1)
07. [ root@svr5 ~] # let X+=78 ; echo $X   # X+=78( X=X+78)
08. [ root@svr5 ~] # let X-=78 ; echo $X   # X-=78( X=X- 78)
09. [ root@svr5 ~] # let X*=78 ; echo $X   # X*=78( X=X*78)
10. [ root@svr5 ~] # let X/=78 ; echo $X   # X/=78( X=X/78)
11. [ root@svr5 ~] # let X%=78 ; echo $X   # X%=78( X=X%78)

```

## 步骤二：小数运算工具

### 1 ) bc交互式运算

先执行bc命令进入交互环境，然后再输入需要计算的表达式。以计算小数12.34与5.678的四则运算为例，相关操作如下：

```

01. [ root@svr5 ~] # bc
02. bc 1.06.95
03. Copyright 1991- 1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
04. This is free software with ABSOLUTELY NO WARRANTY.

```

[Top](#)

```

05.   For details type `warranty'.
06.   12.34+56.78                      //加法
07.   69.12
08.   12.34-56.78                      //减法
09.   -44.44
10.   12.34*56.78                      //乘法
11.   700.66
12.   12.34/56.78                      //除法
13.   0
14.   quit                             //退出交互计算器
15.   [root@svr5 ~]#

```

## 2) bc非交互式运算

将需要运算的表达式通过管道操作交给bc运算。注意，小数位的长度可采用scale=N限制，除此以外也受参与运算的数值的小数位影响。以计算小数12.34与5.678的四则运算为例，相关操作如下：

```

01.   [root@svr5 ~]# echo 'scale=4; 12.34+5.678' | bc
02.   18.018
03.   [root@svr5 ~]# echo 'scale=4; 12.34*5.678' | bc
04.   70.0665
05.   [root@svr5 ~]# echo 'scale=4; 12.34/5.678' | bc
06.   2.1733

```

## 2 案例2：条件测试操作

### 2.1 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

### 2.2 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。

[Top](#)

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：

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

当普通用户执行时：

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

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=""
```

[Top](#)

```

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。

```

01. [ root@svr5 ~]# X=20 //定义一个测试变量
02. [ root@svr5 ~]# [ $X -eq 20 ] && echo "相等" || echo "不相等"
03. 相等
04. [ root@svr5 ~]# [ $X -eq 30 ] && echo "相等" || echo "不相等"
05. 不相等

```

2) -ne 比较两个数是否不相等。

```

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

```

3) -gt 比较前面的整数是否大于后面的整数。

```

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

```

[Top](#)

- 05. 否
- 06. [ root@svr5 ~] # [ \$X - gt 30 ] && echo "大于" || echo "否"
- 07. 否

4 ) -ge 比较前面的整数是否大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 - ge 10 ] && echo "大于或等于" || echo "否"
- 03. 大于或等于
- 04. [ root@svr5 ~] # [ \$X - ge 20 ] && echo "大于或等于" || echo "否"
- 05. 大于或等于
- 06. [ root@svr5 ~] # [ \$X - ge 30 ] && echo "大于或等于" || echo "否"
- 07. 否

5 ) -lt 比较前面的整数是否小于后面的整数。

- 01. [ 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。

[Top](#)

```

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"
04. 存在
05. [root@svr5 ~]# [ -e "/home/nooby" ] && echo "存在" || echo "不存在"
06. 不存在

```

2) -d 判断对象是否为目录（存在且是目录）

```

01. [root@svr5 ~]# [ -d "/usr/" ] && echo "是目录" || echo "不是目录"
02. 是目录
03. [root@svr5 ~]# [ -d "/etc/fstab" ] && echo "是目录" || echo "不是目录"
04. 不是目录
05. [root@svr5 ~]# [ -d "/home/nooby" ] && echo "是目录" || echo "不是目录"
06. 不是目录

```

3) -f 判断对象是否为文件（存在且是文件）

```

01. [root@svr5 ~]# [ -f "/usr/" ] && echo "是文件" || echo "不是文件"
02. 不是文件
03. [root@svr5 ~]# [ -f "/etc/fstab" ] && echo "是文件" || echo "不是文件"

```

[Top](#)

- 04. 是文件
- 05. [ root@svr5 ~ ] # [ - f "/home/nooby" ] && echo "是文件" || echo "不是文件"
- 06. 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：

- 01. [ root@svr5 ~ ] # cp /etc/hosts /tmp/test.txt //复制一个文件做测试
- 02. [ root@svr5 ~ ] # chmod - r /tmp/test.txt //去掉所有的r权限
- 03. [ root@svr5 ~ ] # [ - r "/tmp/test.txt" ] && echo "可读" || echo "不可读"
- 04. 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”：

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

#### 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：

- 01. [ root@svr5 ~ ] # chmod - w /tmp/test.txt //去掉所有的w权限
- 02. [ root@svr5 ~ ] # ls - l /tmp/test.txt //确认设置结果
- 03. ----- 1 root root 33139 12- 11 10: 43 /tmp/test.txt
- 04. [ root@svr5 ~ ] # [ - w "/tmp/test.txt" ] && echo "可写" || echo "不可写"
- 05. 可写

切换为普通用户，可以正常使用-w测试：

- 01. [ zengye@svr5 ~ ] \$ ls - l /tmp/test.txt
- 02. ----- 1 root root 33139 12- 11 10: 52 /tmp/test.txt
- 03. [ zengye@svr5 ~ ] \$ [ - w "/tmp/test.txt" ] && echo "可写" || echo "不可写"
- 04. 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：

[Top](#)



```

01. [root@svr5 ~]# chmod 644 /tmp/test.txt //重设权限,无x
02. [root@svr5 ~]# ls -l /tmp/test.txt //确认设置结果
03. -rw-r--r-- 1 root root 33139 12- 11 10:52 /tmp/test.txt
04. [root@svr5 ~]# [ -x "/tmp/test.txt" ] && echo "可执行" || echo "不可执行"
05. 不可执行
06. [root@svr5 ~]# chmod +x /tmp/test.txt //添加x权限
07. [root@svr5 ~]# [ -x "/tmp/test.txt" ] && echo "可执行" || echo "不可执行"
08. 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：

```

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

```

### 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：

```

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

```

## 3 案例3：使用if选择结构

### 3.1 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

[Top](#)

小于60 初学乍练

## 3.2 方案

if单分支的语法组成：

01. if 条件测试
02. then
03. 命令序列
04. fi

if双分支的语法组成：

01. if 条件测试
02. then
03. 命令序列1
04. else
05. 命令序列2
06. fi

if多分支的语法组成：

01. if 条件测试1;then
02. 命令序列1
03. elif 条件测试2;then
04. 命令序列2
05. else
06. 命令序列n
07. fi

## 3.3 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：

01. [ root@svr5 ~] # vim mountdir.sh
02. #!/bin/bash

[Top](#)

```

03.  dir="/media/cdrom/"
04.  if [ ! -d $dir ]
05.  then
06.      mkdir -p $dir
07.  fi
08.
09.  [ root@svr5 ~] # chmod +x mountdir.sh           //添加可执行权限

```

## 2 ) 测试、验证脚本功能

```

01.  [ root@svr5 ~] # ls -ld /media/cdrom           //本来没有/media/cdrom目录
02.  ls: /media/cdrom: 没有那个文件或目录
03.  [ root@svr5 ~] # ./mountdir.sh                 //执行脚本
04.  [ root@svr5 ~] # ls -ld /media/cdrom           //再检查已经有了
05.  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：

```

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

```

## 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：

```

01.  [ root@svr5 ~] # ping -c 3 -i 0.2 -W 1 192.168.4.5
02.  PING 192.168.4.5 (192.168.4.5) 56(84) bytes of data.
03.  64 bytes from 192.168.4.5: icmp_seq=1 ttl=64 time=0.131 ms
04.  64 bytes from 192.168.4.5: icmp_seq=2 ttl=64 time=0.076 ms
05.  64 bytes from 192.168.4.5: icmp_seq=3 ttl=64 time=0.073 ms
06.
07.  --- 192.168.4.5 ping statistics ---
08.  3 packets transmitted, 3 received, 0% packet loss, time 402ms
09.  rtt min/avg/max/mdev = 0.073/0.093/0.131/0.027 ms
10.

```

[Top](#)

```
11. [root@svr5 ~] # echo $? //执行状态表示成功
12. 0
```

## 2) 脚本编写参考如下：

```
01. [root@svr5 ~] # vim pinghost.sh
02. #!/bin/bash
03. ping -c 3 -i 0.2 -W 1 $1 &> /dev/null
04. if [ $? -eq 0 ] ; then
05.     echo "Host $1 is up."
06. else
07.     echo "Host $1 is down."
08. fi
09.
10. [root@svr5 ~] # chmod +x pinghost.sh
```

## 3) 测试、验证脚本功能

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

[Top](#)

```
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](#)