**Shell02小数计算条件测试if选择for循环**

**一 小数计算**

**1.1交互式计算**

[student@room9pc01 ~]$ bc

bc 1.06.95 #进入提示符

1.1+1

2.1

**scale**=3 **#指定小数位数**

10/3

3.333

**1.2非交互式计算**

[student@room9pc01 ~]$ echo "1.1+1" | bc

2.1

[student@room9pc01 ~]$ echo "scale=3;10/3" | bc

3.333

**二 条件测试**

**2.1 test测试操作**

格式1: test 选项 参数

**格式2: [ 选项 参数 ]**  **#注意空格,两种格式效果相同**

help test 查询帮助

**2.2 字符串测试**

**格式: [ 操作符 字符串 ]** **#注意空格**

操作符 含义

== 两个字符串相同

!= 两个字符串不相同

-z 字符串的值为空,检查变量的值是否设置

-n 字符串的值不为空(相当于! -z)

**2.3 数字测试**

**格式: [ 整数值1 操作符 整数值2 ]** **#注意空格**

操作符 含义

-eq Equal

-ne Not Equal

-ge Greater or Equal

-le Lesser OR Equal

-gt Greater Than

-lt Lesser Than

**2.4文件测试**

**格式: [ 操作符 文件或目录 ]**  **#注意空格**

操作符 含义

**-e** 判断对象是否存在(Exist),若存在则结果为真

**-d** 判断对象是否为目录(Directory),是则为真

**-f** 判断对象是否为一般文件(File),是则为真

**-r** 判断对象是否可读(Read),是则为真

**-w** 判断对象是否可写(Write),是则为真

**-x** 判断对象是否可执行(eXcute),是则为真

**2.5逻辑测试:** 将字符串 数字 文件三个测试综合在一起测试

**三 一行执行多条命令**

# A && B #仅当A命令执行成功，才执行B命令

# A || B #仅当A命令执行失败，才执行B命令

# A ; B #执行A命令后执行B命令，两者没有逻辑关系

**&&，逻辑与 给定条件必须都成立，整个测试结果才为真**。

**||，逻辑或 只要其中一个条件成立，则整个测试结果为真。**

**案例：条件测试操作**

**字符串测试**

**== 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

**!= 比较两个字符串是否不相同**

当普通用户执行时：

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

当root用户执行时：

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

**-z 检查变量的值是否未设置（空值）**

[root@svr5 ~]# var1="nb" ; var2=""

[root@svr5 ~]# [ -z "$var1" ] && echo "空值" || echo "非空值"

非空值

[root@svr5 ~]# [ -z $var2 ] && echo "空值" || echo "非空值"

空值   #变量var2已设置，但无任何值，视为空

[root@svr5 ~]# [ ! -z $va1）r1 ]   #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）。

**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：

[root@svr5 ~]# A=20.4

[root@svr5 ~]# [ $A -gt 10 ] #不支持小数比较

-bash: [: 20.4: integer expression expected

**-eq 比较两个数是否相等**

[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

**-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

**-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

**-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

**-lt 比较前面的整数是否小于后面的整数。**

[root@svr5 ~]# X=20 #定义一个测试变量

[root@svr5 ~]# [ $X -lt 10 ] && echo "小于" || echo "否"

否

[root@svr5 ~]# [ $X -lt 20 ] && echo "小于" || echo "否"

否

[root@svr5 ~]# [ $X -lt 30 ] && echo "小于" || echo "否"

小于

**-le 比较前面的整数是否小于或等于后面的整数。**

[root@svr5 ~]# X=20  #定义一个测试变量

[root@svr5 ~]# [ $X -le 10 ] && echo "小于或等于" || echo "否"

否

[root@svr5 ~]# [ $X -le 20 ] && echo "小于或等于" || echo "否"

小于或等于

[root@svr5 ~]# [ $X -le 30 ] && echo "小于或等于" || echo "否"

小于或等于

**提取当前登录的用户数，比较是否大于等于3。**

[root@svr5 ~]# who | wc -l #确认已登录的用户数

**多逻辑符号,要将符号前的整体的执行结果,作为后面命令是否执行的参考**

**多逻辑符号中,&&可看做 且 关系; ||可看做 或 关系**

[root@svr5 ~]# N=$(who | wc -l) #赋值给变量N

[root@svr5 ~]# [ $N -ge 3 ] && echo "超过了" || echo "没超过"

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：

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

没超过

**识别文件/目录的状态**

**-e 判断对象是否存在（不管是目录还是文件）**

~]# [ -e "/usr/" ] && echo "存在" || echo "不存在"

存在

~]# [ -e "/etc/fstab" ] && echo "存在" || echo "不存在"

存在

~]# [ -e "/home/nooby" ] && echo "存在" || echo "不存在"

不存在

**-d 判断对象是否为目录（存在且是目录）**

~]# [ -d "/usr/" ] && echo "是目录" || echo "不是目录"

是目录

~]# [ -d "/etc/fstab" ] && echo "是目录" || echo "不是目录"

不是目录

~]# [ -d "/home/nooby" ] && echo "是目录" || echo "不是目录"

不是目录

**-f 判断对象是否为文件（存在且是文件）**

~]# [ -f "/usr/" ] && echo "是文件" || echo "不是文件"

不是文件

~]# [ -f "/etc/fstab" ] && echo "是文件" || echo "不是文件"

是文件

~]# [ -f "/home/nooby" ] && echo "是文件" || echo "不是文件"

不是文件

**-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**：

~]# cp /etc/hosts /tmp/test.txt         #复制一个文件做测试

~]# chmod -r /tmp/test.txt              #去掉所有的r权限

~]# [ -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”：

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

不可读

**-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**：

~]# chmod -w /tmp/test.txt             #去掉所有的w权限

~]# ls -l /tmp/test.txt             #确认设置结果

---------- 1 root root 33139 12-11 10:43 /tmp/test.txt

~]# [ -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：

~]$ ls -l /tmp/test.txt

---------- 1 root root 33139 12-11 10:52 /tmp/test.txt

~]$ [ -w "/tmp/test.txt" ] && echo "可写" || echo "不可写"

可写

**-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**：

~]# chmod 644 /tmp/test.txt         #重设权限，无x

~]# 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

~]# [ -x "/tmp/test.txt" ] && echo "可执行" || echo "不可执行"

不可执行

~]# chmod +x /tmp/test.txt         #添加x权限

~]# [ -x "/tmp/test.txt" ] && echo "可执行" || echo "不可执行"

可执行

**多个条件/操作的逻辑组合**

**&&，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

**||，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

案例:编写脚本,实现以下需求

每隔2分钟检查登录服务器的账户,如果超过3人,则发邮件给管理员报警

查看登录服务器的用户数

[root@desktop0 opt]# who | wc -l

2

发邮件

方法1: echo 123 | mail -s test root

方法2: mail -s test root < a.txt #a.txt中为提示信息

计划任务crontab

[root@desktop0 opt]# vim test1.sh

#!/bin/bash

n=$(who | wc -l)

[ $n -gt 3 ] && echo "有人入侵服务器啦!" | mail -s test root

[root@desktop0 opt]# chmod +x test1.sh

[root@desktop0 opt]# crontab -e

\*/2 \* \* \* \* /opt/test1.sh

新开终端,ssh到root@desktop

[root@desktop0 ~]# mail #打开邮件查看

**三 if 选择结构**

**3.1 if 单分支**

if 条件测试; then

命令序列

fi

if单分支案例: 编写脚本,添加用户,当未输入用户名时,提示”请输入用户名”.

[root@desktop0 opt]# vim useradd.sh

if [ -z $1 ];then

echo "请给用户名!"

exit

fi

useradd $1 &> /dev/null

[root@desktop0 opt]# . useradd.sh ccc

[root@desktop0 opt]# id ccc

uid=1005(ccc) gid=1005(ccc) 组=1005(ccc)

[root@desktop0 opt]# . useradd.sh

请给用户名!

**3.2 if 双分支**

if 条件测试; then

命令序列1

else

命令序列2

fi

if双分支案例: 编写脚本,测试一个IP地址,如果通输出ok,如果不通输出no.

[student@room9pc01 ~]$ ping -c 3 -i 0.2 -W 1 192.168.4.254

**-c** 指定ping的次数, **-i** 指定每次ping的时间间隔,

**-W** 指定ping的反馈时间

[student@room9pc01 ~]$ vim test2.sh

#!/bin/bash

ping -c 3 -i 0.2 -W 1 $! &> /dev/null

if [ $? -eq 0 ];then

echo "ok"

else

echo "no"

fi

[student@room9pc01 ~]$ . test2.sh 172.25.0.11

ok

[student@room9pc01 ~]$ . test2.sh 172.25.0.12

no

**3.3 if 多分支**

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

if 三分支案例:随机生成一个0-9的正整数,提示用户输入一个0-9的正整数,比较大小并给出提示信息.

$RANDOM 随机产生一个正整数

[root@desktop0 ~]# echo $RANDOM

24199

[root@desktop0 ~]# echo $[RANDOM%10] #生成0-9的随机数

[root@desktop0 ~]# vim test3.sh

#!/bin/bash

x=$[RANDOM%10]

read -p "请输入一个0-9的数字:" n

if [ $x -eq $n ];then

echo "恭喜,猜对了!"

elif [ $n -gt $x ];then

echo "猜大了!"

else

echo "猜小了!"

fi

**四 for循环**

**4.1 语法格式**

for 变量名 in 值列表

do

命令序列

done

**4.2 注意点**

循环次数

循环过程中调用的变量

**4.3 seq**

**格式: seq 正整数**

[root@desktop0 opt]# seq 5

1

...

5

在for循环中使用$(seq 5)或`seq 5`

**案例:** 换行连续输出1 2 3 4 5

[root@desktop0 opt]# vim test4.sh

#!/bin/bash

a=5

#for i in $(seq $a) #$(seq $a)与`seq $a`等效

for i in `seq $a`

do

echo $i

done

**案例:** 编写脚本,ping 172.25.0.1-172.25.0.15,并给出提示信息.

[root@desktop0 opt]# vim test5.sh

#!/bin/bash

for i in {1..15}

do

ping -c 3 -i 0.2 -W 1 172.25.0.$i &> /dev/null

if [ $? -eq 0 ];then

echo "172.25.0.$i能ping通!"

else

echo "172.25.0.$i不能ping通!"

fi

done

[root@desktop0 opt]# . test5.sh #测试脚本功能

172.25.0.1不能ping通!

......

172.25.0.10能ping通! #只有172.25.0.10和172.25.0.11能ping通

172.25.0.11能ping通!

......

172.25.0.15不能ping通!

**案例:** 编写脚本,ping 172.25.0.1-172.25.0.15,并给出提示信息,并给出通与不同的数量

#!/bin/bash

**a=0**

**b=0**

for i in {1..15}

do

ping -c 3 -i 0.2 -W 1 172.25.0.$i &> /dev/null

if [ $? -eq 0 ];then

echo "172.25.0.$i能ping通!"

**let a++**

else

echo "172.25.0.$i不能ping通!"

**let b++**

fi

done

**echo "$a台主机能ping通!"**

**echo "$b台主机不能ping通!"**

**五 while 循环**

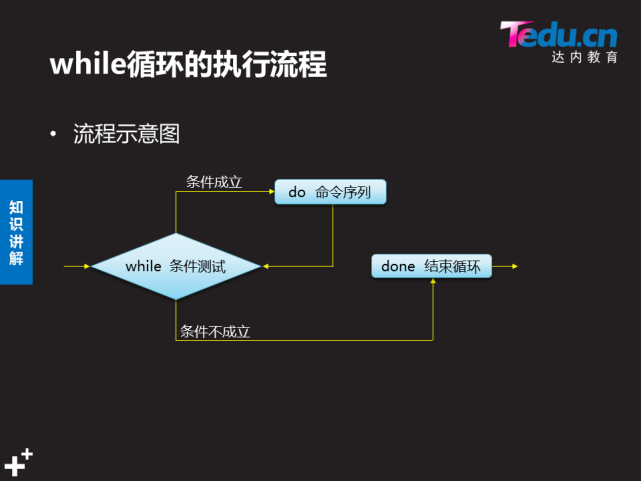
**5.1 语法格式**

while 条件测试

do

命令序列

done



[root@desktop0 opt]# vim test6.sh

#!/bin/bash

while [ 1 -eq 1 ]

do

echo 123

sleep 0.1 #部署一直执行任务时,此命令可解决CPU占用率高的问题

done

**#while : #永远正确,一直进行**

案例:编写脚本,让用户输入0-99的数字,将用户输入的数字与后台数字比较,并给出提示信息,直到用户猜对后退出.

[root@desktop0 opt]# vim test7.sh

#!/bin/bash

x=$[RANDOM%100]

while :

do

read -p "请输入数字(0-99):" n

if [ $x -eq $n ];then

echo "猜对了!"

**exit**

elif [ $n -lt $x ];then

echo "猜小了!"

else

echo "猜大了!"

fi

done

[root@desktop0 opt]# . test7.sh

请输入数字(0-99):50

猜小了!

请输入数字(0-99):60

猜大了!

请输入数字(0-99):55

猜对了!

案例: 编写脚本,让用户输入0-99的数字,将用户输入的数字与后台数字比较,并给出提示信息,直到用户猜对后退出,并给出用户猜的次数.

#!/bin/bash

x=$[RANDOM%100]

**y=0**

while :

do

**let y++**

read -p "请输入数字(0-99):" n

if [ $x -eq $n ];then

echo "猜对了!**你猜了$y次!**"

**exit**

elif [ $n -lt $x ];then

echo "猜小了!"

else

echo "猜大了!"

fi

done