

Linux操作系统

7 Shell运算符

主讲：杨东平
中国矿大计算机学院

declare 命令(续)

➤例1: declare -i 之后可以直接对表达式求值

```
[root@localhost ~]# x=6/3
[root@localhost ~]# echo $x          # 显示: 6/3
[root@localhost ~]# declare -i x
[root@localhost ~]# echo $x          # 显示: 6/3
[root@localhost ~]# x=6/3
[root@localhost ~]# echo $x          # 显示: 2
# 可以把表达式直接赋给整型变量, bash 会对它求值
[root@localhost ~]# x=error
[root@localhost ~]# echo $x          # 显示: 0
# 把一个结果不是整数的表达式赋值给整型变量时, 就会变成 0
[root@localhost ~]# x=3.14
-bash: 3.14: syntax error: invalid arithmetic operator (error token is ".14")
# bash 不内置对浮点数的支持
[root@localhost ~]# declare +i x
# 此命令的结果是取消变量x的整型类型属性
[root@localhost ~]# x=6/3
[root@localhost ~]# echo $x          # 显示: 6/3
# 变量x不是整型, 不会自动对表达式求值。可以采用下面两种方式:
[root@localhost ~]# x=${6/3}
[root@localhost ~]# echo $x          # 显示: 2
[root@localhost ~]# x=$((6/3))
[root@localhost ~]# echo $x          # 显示: 2
```

网络安全与网络工程系杨东平 jsxhbc@163.com Linux操作系统 2018年9月14日1时25分 3

declare 命令

➤ declare 命令用于声明和显示已存在的 Shell 变量

➤ 语法: declare [选项] (参数)

➤ 选项:

- +/- 取消/设置变量的类型属性
- a 数组类型
- i 整型, 如果求值失败或者不是整数, 就设置为0
- x 环境变量, 可供 Shell 以外的程序使用
- r 只读变量
- p 显示指定变量的被声明的类型
- f 仅显示函数

➤ 参数

- ❖ 声明 Shell 变量, 初始化格式为 “变量名=值”

➤ 说明:

- ❖ 若不带任何参数选项, 则会显示所有 Shell 变量及其值
- ❖ declare 的功能与 typeset 命令的功能相同的

网络安全与网络工程系杨东平 jsxhbc@163.com Linux操作系统 2018年9月14日1时25分 2

declare 命令(续)

➤例1: declare -i 之后可以直接对表达式求值(续)

```
[root@localhost ~]# x=6/3
[root@localhost ~]# echo $x
6/3
[root@localhost ~]# declare -i x
[root@localhost ~]# echo $x
6/3
[root@localhost ~]# x=6/3
[root@localhost ~]# echo $x
2
[root@localhost ~]# x=error
[root@localhost ~]# echo $x
0
[root@localhost ~]# x=3.14
-bash: 3.14: syntax error: invalid arithmetic operator (error token is ".14")
[root@localhost ~]# declare +i x
[root@localhost ~]# echo $x
6/3
[root@localhost ~]# x=${6/3}
[root@localhost ~]# echo $x
2
[root@localhost ~]# x=$((6/3))
[root@localhost ~]# echo $x
2
```

网络安全与网络工程系杨东平 jsxhbc@163.com Linux操作系统 2018年9月14日1时25分 4

declare 命令(续)

➤例2: 只读变量

```
[root@localhost ~]# declare -r r      # 声明为只读变量
[root@localhost ~]# echo $r          # 没有初始化
# 显示:
[root@localhost ~]# r=xxx
# 显示: -bash: r: readonly variable
[root@localhost ~]# declare -r r=xxx
# 显示: -bash: declare: r: readonly variable
[root@localhost ~]# declare +r r
# 显示: -bash: declare: r: readonly variable
[root@localhost ~]# unset r
# 显示: -bash: unset: r: cannot unset: readonly variable
```

- 只读变量不能修改、不能删除、不能取消只读属性
- 变量只读性是临时的, 系统重启或重新登录后即失效

```
[root@localhost ~]# declare -r r
[root@localhost ~]# echo $r

[root@localhost ~]# r=xxx
-bash: r: readonly variable
[root@localhost ~]# declare -r r=xxx
-bash: declare: r: readonly variable
[root@localhost ~]# declare +r r
-bash: declare: r: readonly variable
[root@localhost ~]# unset r
-bash: unset: r: cannot unset: readonly variable
```

网络安全与网络工程系杨东平 jsxhbc@163.com Linux操作系统 2018年9月14日1时25分 5

declare 命令(续)

➤例3: 声组变量(实际上, 任何变量都可以当做数组来操作)

```
[root@localhost ~]# declare -a names
[root@localhost ~]# names=Jack
[root@localhost ~]# echo ${names[0]} # 显示: Jack
[root@localhost ~]# names[1]=Bone
[root@localhost ~]# echo ${names[1]} # 显示: Bone
# 直接引用names, 相当于引用names[0]
[root@localhost ~]# echo ${names}   # 显示: Jack
[root@localhost ~]# echo ${names[*]} # 显示: Jack Bone
[root@localhost ~]# echo ${names[@]} # 显示: 4
[root@localhost ~]# echo ${#names[*]} # 显示: 2
[root@localhost ~]# echo ${names[@]} # 显示: Jack Bone
declare -a names=([0]="Jack" [1]="Bone")
[root@localhost ~]# echo ${names[@]} # 显示: 2
[root@localhost ~]# declare -p names
# 显示: declare -a names=([0]="Jack" [1]="Bone")
```

网络安全与网络工程系杨东平 jsxhbc@163.com Linux操作系统 2018年9月14日1时25分 6

declare 命令(续)

➤例4：环境变量

- ❖ 语法：declare -x 变量名
- [root@localhost ~] # declare -x sum

➤例5：查询变量的属性

- ❖ 语法：declare -p
- ❖ 说明：查询所有变量的属性
- ❖ 语法：declare -p 变量名
- ❖ 说明：查询指定变量的属性

数值运算

➤数值运算方法1

- ❖ 语法：declare -i 变量=\$变量1+\$变量2
- ❖ 说明：
 - ☞ 变量与=之间不能有空格
 - ☞ 变量与+之间不能有空格

➤例： [root@localhost ~] # a=1
[root@localhost ~] # b=2
[root@localhost ~] # declare -i c=\$a+\$b
[root@localhost ~] # echo \$c

结果：3

```
[root@localhost ~]# a=1
[root@localhost ~]# b=2
[root@localhost ~]# declare -i c=$a+$b
[root@localhost ~]# echo $c
3
```

数值运算(续)

➤数值运算方法2：expr 数值运算工具

- ❖ 语法：变量=\$(expr \$变量 + \$变量)
- ❖ 说明：将需要运算的表达式写入在 expr 后面
- ❖ 注意：
 - ☞ =左右两边不能有空格
 - ☞ +左右两边必须有空格

```
[root@localhost ~]# a=1
[root@localhost ~]# b=2
[root@localhost ~]# c=$(expr $a + $b)
[root@localhost ~]# echo $c
3
[root@localhost ~]# x=1
[root@localhost ~]# y=2
[root@localhost ~]# z=$(expr $x+$y)
[root@localhost ~]# echo $z
1+2
```

数值运算(续)

➤数值运算方法3：\$((运算式)) 与 \${运算式}

- ❖ 语法：变量=\$((运算式))
变量=\${运算式}
- ❖ 说明：
 - ☞ =左右两边不能有空格
 - ☞ 运算式随便写，很自由

```
[root@localhost ~]# a=1
[root@localhost ~]# b=2
[root@localhost ~]# c=$((($a + $b))
[root@localhost ~]# echo $c
3
[root@localhost ~]# x=1
[root@localhost ~]# y=3
[root@localhost ~]# z=$(( $x + $y))
[root@localhost ~]# echo $z
4
```

Shell 基本运算符

➤Shell 支持多种运算符，包括：

- ❖ 算术运算符
- ❖ 关系运算符
- ❖ 布尔运算符
- ❖ 字符串运算符
- ❖ 文件测试运算符

算术运算符

运算符	说明
+	加
-	减
*	乘
/	除
%	取余
=	赋值
==	相等，用于比较两个数字，相同则返回 true
!=	不相等，用于比较两个数字，不相同则返回 true

➤注意：为防止歧义，乘号(*)前边应该加反斜杠(\)

算术运算符(续)

> 例: 视频(8 算术运算符)

```
#!/bin/bash
# exp1.sh

a=10
b=20

val=`expr $a + $b`
echo "a + b : $val"
```

```
val=`expr $a - $b`
echo "a - b : $val"
```

```
val=`expr $a \* $b`
echo "a * b : $val"
```

```
val=`expr $a \ / $b`
echo "a \ / b : $val"
```

```
if [ $a == $b ]
then
    echo "a is equals b"
fi

if [ $a != $b ]
then
    echo "a is not equals b"
fi
```

```
[root@localhost ~]# ./exp1.sh
a + b : 30
a - b : -10
a * b : 200
a \ / b : 0
a is not equals b
```

网络安全与网络工程系东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

13

算术运算符(续)

优先级	运算符	说明
13	~, +	单目负、单目正
12	!, ~	逻辑非、按位取反或补码
11	*, /, %	乘、除、取模
10	+, -	加、减
9	<<, >>	按位左移、按位右移
8	<=, >=, <, >	小于或等于、大于或等于、小于、大于
7	==, !=	等于、不等于
6	&	按位与
5		按位异或
4	&&	按位或
3	&&	逻辑与
2		逻辑或
1	=, +=, -=, *=, /=, %=, &=, =, <<=, >>=	赋值、运算且赋值

网络安全与网络工程系东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

14

算术运算符(续)

> 例

- [root@localhost ~]# aa=\$(((11+3)*3/2))
- #虽然乘和除的优先级高于加, 但是通过小括号可以调整运算优先级
- [root@localhost ~]# bb=\$((14%3))
- #14不能被3整除, 余数是2
- [root@localhost ~]# cc=\$((1 && 0))
- #逻辑与运算只有想与的两边都是1, 与的结果才是1, 否则与的结果是0

网络安全与网络工程系东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

15

关系运算符

> 关系运算符只支持数字, 不支持字符串, 除非字符串的值是数字

运算符	说明
-eq	检测两个数是否相等, 相等返回 true
-ne	检测两个数是否不相等, 不相等返回 true
-gt	检测左边的数是否大于右边的, 如果是则返回 true
-lt	检测左边的数是否小于右边的, 如果是则返回 true
-ge	检测左边的数是否大于等于右边的, 如果是则返回 true
-le	检测左边的数是否小于等于右边的, 如果是则返回 true

网络安全与网络工程系东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

16

关系运算符(续)

> 例2: 视频(9 关系运算符)

```
#!/bin/bash
# exp2.sh

a=10
b=20

if [ $a -eq $b ]
then
    echo "$a -eq $b : a==b"
else
    echo "$a -eq $b : a!=b"
fi

if [ $a -ne $b ]
then
    echo "$a -ne $b : a!=b"
else
    echo "$a -ne $b : a==b"
fi

if [ $a -gt $b ]
then
    echo "$a -gt $b : a>b"
else
    echo "$a -gt $b : a<b"
```

```
echo "$a -gt $b: a>b"
else
    echo "$a -gt $b: a<b"
fi

if [ $a -lt $b ]
then
    echo "$a -lt $b: a<b"
else
    echo "$a -lt $b: a>b"
fi

if [ $a -ge $b ]
then
    echo "$a -ge $b: a>= b"
else
    echo "$a -ge $b: a<b"
fi

if [ $a -le $b ]
then
    echo "$a -le $b: a<=b"
else
    echo "$a -le $b: a>b"
fi
```

网络安全与网络工程系东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

17

布尔运算符

运算符	说明
!	非运算, 表达式为 true 则返回 false, 否则返回 true
-o	或运算, 有一个表达式为 true 则返回 true
-a	与运算, 两个表达式都为 true 才返回 true

> 例(视频: 10 布尔运算符):

```
#!/bin/bash
# exp3.sh

a=10
b=20

if [ $a -lt 100 -a $b -gt 15 ]
then
    echo "$a<100 And $b>15 : return true"
else
    echo "$a<100 And $b>15 : return false"
fi

if [ $a -lt 100 -o $b -gt 100 ]
then
    echo "$a<100 Or $b>100 : return true"
else
    echo "$a<100 Or $b>100 : return false"
fi
```

```
echo "$a<100 Or $b<100 : return true"
else
    echo "$a<100 Or $b>100 : return false"
fi

if [ $a -lt 5 -o $b -gt 100 ]
then
    echo "$a<5 Or $b>100 : return true"
else
    echo "$a<5 Or $b>100 : return false"
fi

[root@localhost ~]# chmod 755 exp3.sh
[root@localhost ~]# ./exp3.sh
a<100 and b>15 : return true
a<100 or b<100 : return true
a<5 or b>100 : return false
```

网络安全与网络工程系东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

18

逻辑运算符

运算符	说明
&&	逻辑的 AND
	逻辑的 OR

➤ 例(视频: 11 逻辑运算符):

```
# /bin/bash
# exp4.sh

a=10
b=20

if [[ $a -lt 100 && $b -gt 100 ]]
then
    echo "return true"
else
    echo "return false"
fi
```

[root@localhost ~]# chmod 755 exp4.sh
[root@localhost ~]# ./exp4.sh
return false

网络安全与网络工程系杨东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

20

字符串运算符

运算符	说明
=	检测两个字符串是否相等, 相等返回 true
!=	检测两个字符串是否相等, 不相等返回 true
-z	检测字符串长度是否为 0, 为 0 返回 true
-n	检测字符串长度是否为 0, 不为 0 返回 true
str	检测字符串 str 是否为空, 不为空返回 true

网络安全与网络工程系杨东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

20

字符串运算符(续)

➤ 例(视频: 12 字符串运算符):

```
# /bin/bash
# exp5.sh

a="abc"
b="efg"

if [ $a = $b ]
then
    echo "$a = $b : a equals b"
else
    echo "$a = $b: a is not equals to b"
fi

if [ $a != $b ]
then
    echo "$a != $b : a is not equals to b"
else
    echo "$a != $b: a equals b"
fi

if [ -z $a ]
then
    echo "-z $a : string length is 0"
else
    echo "-z $a : string length is not 0"
fi

if [ -n $a ]
then
    echo "-n $a : string length is not 0"
else
    echo "-n $a : string length is 0"
fi

if [ $a ]
then
    echo "$a : string is not empty"
else
    echo "$a : string is empty"
fi
```

[root@localhost ~]# ./exp5.sh
abc = efg : a is not equals b
abc != efg : a is not equals b
-z abc : string length is not 0
-n abc : string length is not 0
abc : string is not empty

网络安全与网络工程系杨东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

21

文件测试运算符

➤ 文件测试运算符用于检测 Linux 文件的各种属性

运算符	说明
-b file	检测文件是否是块设备文件, 如果是, 则返回 true
-c file	检测文件是否是字符设备文件, 如果是, 则返回 true
-d file	检测文件是否是目录, 如果是, 则返回 true
-f file	检测文件是否是普通文件(既不是目录, 也不是设备文件), 如果是, 则返回 true
-g file	检测文件是否设置了 SGID 位, 如果是, 则返回 true
-k file	检测文件是否设置了粘着位(Sticky Bit), 如果是, 则返回 true
-p file	检测文件是否是命名管道, 如果是, 则返回 true
-u file	检测文件是否设置了 SUID 位, 如果是, 则返回 true
-r file	检测文件是否可读, 如果是, 则返回 true
-w file	检测文件是否可写, 如果是, 则返回 true
-x file	检测文件是否可执行, 如果是, 则返回 true
-s file	检测文件是否为空(文件大小是否大于 0), 不为空返回 true
-e file	检测文件(包括目录)是否存在, 如果是, 则返回 true

网络安全与网络工程系杨东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

22

➤ 例(视频: 13 文件测试运算符):

```
# /bin/bash
# exp6.sh

file="/exp5.sh"

if [ -r $file ]
then
    echo "file can read"
else
    echo "file cannot read"
fi

if [ -w $file ]
then
    echo "file can write"
else
    echo "file cannot write"
fi

if [ -x $file ]
then
    echo "file can execute"
else
    echo "file cannot execute"
fi

if [ -f $file ]
then
    echo "file is ordinary"
else
    echo "file is special"
fi

if [ -d $file ]
then
    echo "file is directory"
else
    echo "file is not directory"
fi

if [ -s $file ]
then
    echo "file is not empty"
else
    echo "file is empty"
fi

if [ -e $file ]
then
    echo "file is exist"
else
    echo "file is not exist"
fi
```

[root@localhost ~]# ./exp6.sh
file="/exp5.sh"
file can read
file can write
file can execute
file is ordinary
file is not directory
file is empty
file is not exist

网络安全与网络工程系杨东平 jsxhbc@163.com

Linux操作系统

2018年9月14日1时25分

21