**CentOS** 丛书目录 — 系统管理 — 网络服务 — 应用部署

# 深入 **Shell** 变量操作

内容提要

1. 掌握变量替换扩展的使用
2. 掌握变量的字符串操作
3. 掌握使用(())进行变量的整数计算
4. 学会使用 read 命令从标准输入读取变量的值

## 变量替换扩展

在 Shell 变量和 Shell 环境 中已经介绍了变量赋值和基本的变量替换。并已经知道 ${var} 与 $var 相同，下面介绍变量替换扩展。

| 表达式 | 说明 |
|---|---|
| ${var-DEFAULT} | 若var没被声明，则以 DEFAULT 作为其值 * |
| ${var:-DEFAULT} | 若var没被声明或者其值为空， 则以 DEFAULT 作为其值 * |
| ${var=DEFAULT} | 若var没被声明，则以 DEFAULT 作为其值 * |
| ${var:=DEFAULT} | 若var没被声明或者其值为空， 则以 DEFAULT 作为其值 * |
| ${var+OTHER} | 若var被声明了，那么其值就是 OTHER，否则就为空 |
| ${var:+OTHER} | 若var被声明了或者其值不为空，那么其值就是 OTHER，否则就为空 |
| ${!varprefix*} | 匹配之前所有以varprefix开头进行声明的变量 |
| ${!varprefix@} | 匹配之前所有以varprefix开头进行声明的变量 |

* 如果变量var已经被设置，那么其值就是$var。

下面举例说明变量替换扩展的使用方法。

```
# 将变量 var1 的值赋为空
$ var1=
$ var2=unix
# 因为 var1 已被声明，所以返回 var1 的值
$ echo ${var1-linux}

# 因为 var1 已被声明且值为空，所以返回 linux
$ echo ${var1:-linux}
linux
# 因为 vara 未被声明，所以返回 linux
$ echo ${vara-linux}
linux
# 打印变量的值
$ echo $var1,$var2,$vara
,unix,
# 当 var2 已赋值且值不为空，返回 var2 的值
# 此时， ${var2-linux} 与 ${var2:-linux} 相同
$ echo ${var2-linux}
unix
$ echo ${var2:-linux}
unix
# 因为 var1 已被声明（虽然值为空），所以返回 linux
$ echo ${var1+linux}
linux
```

```
# 因为 var2 已被声明，所以返回 linux
$ echo ${var2+linux}
linux
# 因为 var1 已被声明且值为空，所以返回空
$ echo ${var1:+linux}

# 当 var2 不为空时，与 ${var2+linux} 相同
$ echo ${var2:+linux}
linux
# 因为 vara 未被声明，所以返回空
$ echo ${vara+linux}

# 当 vara 未被声明时与 ${vara+linux} 相同
$ echo ${vara:+linux}

# 打印变量的值
$ echo $var1,$var2,$vara
,unix,
# 打印所有以 var 开头的变量
$ echo ${!var*}
var1 var2
# 与 ${!var*} 相同
$ echo ${!var@}
var1 var2
```

- 从以上的 echo $var1,$var2,$vara 命令可知，变量替换扩展并不改变变量的原值
- 通常变量替换扩展在作为赋值语句的右值使用，即将变量替换扩展再赋予另一个变量来使用
- 使用变量替换扩展可以 Shell 脚本中的 if 语句简化为一个使用变量替换扩展的赋值语句

## 变量的字符串操作

| 表达式 | 说明 |
|---|---|
| ${#string} | $string的长度 |
| ${string:position} | 在$string中，从位置$position开始提取子串 |
| ${string:position:length} | 在$string中，从位置$position开始提取长度为$length的子串 |
| ${string#substring} | 从变量$string的开头，删除最短匹配$substring的子串 |
| ${string##substring} | 从变量$string的开头，删除最长匹配$substring的子串 |
| ${string%substring} | 从变量$string的结尾，删除最短匹配$substring的子串 |
| ${string%%substring} | 从变量$string的结尾，删除最长匹配$substring的子串 |
| ${string/substring/replacement} | 使用$replacement，来代替第一个匹配的$substring |
| ${string/#substring/replacement} | 如果$string的前缀匹配$substring，那么就用$replacement来代替匹配到的$substring |
| ${string/%substring/replacement} | 如果$string的后缀匹配$substring，那么就用$replacement来代替匹配到的$substring |

下面举例说明变量的字符串操作使用方法。

```
$ str='I love linux. I love UNIX too.'
# 返回字符串 str 的长度
$ echo ${#str}
30
# 截取str从第13个字符到串尾
$ echo ${str:13}
I love UNIX too.
# 截取str从第7个字符开始的5个字符
$ echo ${str:7:5}
linux
# 删除开始的字符串 I love
$ echo ${str#I love}
linux. I love UNIX too.
# 删除开始的 I 到 . 的所有字符（最短匹配）
```

```
$ echo ${str#I*.}
I love UNIX too.
# 删除开始的 I 到 . 的所有字符（最长匹配）
$ echo ${str##I*}

# 替换开始的 I love 为 J'aime
$ echo ${str/I love/"J'aime"}
J'aime linux. I love UNIX too.
# 替换末尾的 too. 为 also.
$ echo ${str/%too./also.}
I love linux. I love UNIX also.
```

## 变量的数值计算

若一个变量的值是纯数字的，不包含字母或其他字符，bash 可以将其视为长整型值，并可做整型运算。

bash 不能进行浮点数值运算，可以使用 bc 命令进行浮点运算。但通常很少用到。

Chet Ramey 在Bash 2.04 版本之后引入了shell算术运算符 ((...)) ，在此运算符中可以使用C语言风格的表达式结构。这为在 Shell 中进行运算提供了极大地方便。在此之前，要进行数值计算要使用 let 命令或更早出现的 expr 命令。

下面给出几个使用 shell 算术运算的例子：

```
# 使用shell算术运算做赋值
# ** 是幂运算；% 是模运算
$ ((a=2+3**2-1001%5))
$ echo $a
10
# 也可以将计算结果赋给变量，但要使用置换形式
$ a=$((2+3**2-1001%5))
# 可以直接输出计算结果，但要使用置换形式
$ echo $((2+3**2-1001%5))
10
# 可以使用 C 语言的 += 等类似的运算符
$ echo $((a+=2))
12
# 可以使用 C 语言的自增/自减运算符
# 注意 a++ 和 ++a 的区别
$ echo $((a++))
12
$ echo $((++a))
14
# 可以进行关系、逻辑运算，真为1，假为0
$ echo $((2+3**2>1001%5))
1
$ echo $((2+3**2<1001%5&&a))
0
```

## Shell 变量的输入

Shell 变量除了可以直接赋值之外，其值还可以使用内置的 read 命令从标准输入获得。 read 命令的格式为：

```
read [参数] [<变量名> ...]
```

常用的参数有：

- **-p prompt** ： 设置提示信息
- **-t timeout** ： 设置输入的等待秒数

下面给出几个使用例子：

```
# 提示输入三个变量的值
$ read -p "Please input 3 numbers: " n1 n2 n3
Please input 3 numbers: 1 22 333
# 显示这三个变量的值
$ echo $n1 $n2 $n3
1 22 333
# 提示输入两个变量的值
$ read -p "Please input 2 strings: " s1 s2
Please input 2 strings: centos ubuntu
# 显示这两个变量的值
$ echo $s1 $s2
centos ubuntu
# 提示输入一行字符串
$ read -p "Please input a line: " myline
Please input a line: I love Linux and all FLOSS.
# 显示这个变量的值
$ echo $myline
I love Linux and all FLOSS.
# 要求在 10 秒内输入一个变量的值
$ read -t 10 nowaiting
# 等待 10 秒后退出
# 显示这个变量的值
$ echo $nowaiting

# 值为空
# 重新输入变量的值
$ read -t 10 nowaiting
wait me!
# 显示这个变量的值
$ echo $nowaiting
wait me!
$
```

- 显示源文件
- 登录