shell 变量

什么是 shell 变量?
变量的类型
变量的定义方式
变量的运算
变量"内容"的删除和替换

shell 变量? 用一个固定的字符串去表示不固定的内容

变量的类型：
1. 自定义变量
定义变量： 变量名=变量值 变量名必须以字母或下划线开头，区分大小写
ip1=192.168.2.115
引用变量： $变量名 或 ${变量名}
查看变量： echo $变量名 set(所有变量：包括自定义变量和环境变量)
取消变量： unset 变量名
作用范围： 仅在当前 shell 中有效

2. 环境变量
定义环境变量： 方法一 export back_dir2=/home/backup
方法二 export back_dir1 将自定义变量转换成环境变量
引用环境变量： $变量名 或 ${变量名}
查看环境变量： echo $变量名 env 例如 env |grep back_dir2
取消环境变量： unset 变量名
变量作用范围： 在当前 shell 和子 shell 有效
==========
C 语言 局部变量 vs 全局变量
SHELL 自定义变量 vs 环境变量
==========

3. 位置变量
$1 $2 $3 $4 $5 $6 $7 $8 $9 ${10}

4. 预定义变量
$0 脚本名
$* 所有的参数
$@ 所有的参数
$# 参数的个数
$$ 当前进程的 PID
$! 上一个后台进程的 PID
$? 上一个命令的返回值 0 表示成功

示例 1：

```
# vim test.sh
echo "第 2 个位置参数是$2"
echo "第 1 个位置参数是$1"
echo "第 4 个位置参数是$4"

echo "所有参数是:$*"
echo "所有参数是:$@"
echo "参数的个数是:$#"
echo "当前进程的 PID 是:$$"

echo '$1='$1
echo '$2='$2
echo '$3='$3
echo '$*='$*
echo '$@='$@
echo '$#='$#
echo '$$='$$
```

了解$*和$@区别

示例 2：

```
# vim ping.sh
#!/bin/bash
ping -c2 $1 &>/dev/null
if [ $? = 0 ];then
echo "host $1 is ok"
else
echo "host $1 is fail"
fi
# chmod a+x ping.sh
# ./ping.sh 192.168.2.25
```

变量的赋值方式：

1. 显式赋值

变量名=变量值

示例：

```
ip1=192.168.1.251
school="BeiJing 1000phone"
```

today1=`date +%F`
today2=$(date +%F)

2. read 从键盘读入变量值
read 变量名
read -p "提示信息:" 变量名
read -t 5 -p "提示信息:" 变量名
read -n 2 变量名

示例 3：
# vim first.sh
back_dir1=/var/backup
read -p "请输入你的备份目录:" back_dir2
echo $back_dir1
echo $back_dir2
sh first.sh

示例 4：
vim ping2.sh
#!/bin/bash
read -p "Input IP:" ip
ping -c2 $ip &>/dev/null
if [$? = 0];then
echo "host $ip is ok"
else
echo "host $ip is fail"
fi
chmod a+x ping2.sh
./ping.sh

定义或引用变量时注意事项：
" " 弱引用
' ' 强引用
[root@tianyun ~]# school=1000phone
[root@tianyun ~]# echo "${school} is good"
1000phone is good
[root@tianyun ~]# echo '${school} is good'
${school} is good

`` 命令替换 等价于 $() 反引号中的 shell 命令会被先执行
[root@tianyun ~]# touch `date +%F`_file1.txt
[root@tianyun ~]# touch $(date +%F)_file2.txt

[root@tianyun ~]# disk_free3="df -Ph |grep '/$' |awk '{print $4}'" 错误
[root@tianyun ~]# disk_free4=$(df -Ph |grep '/$' |awk '{print $4}')
[root@tianyun ~]# disk_free5=`df -Ph |grep '/$' |awk '{print $4}'`

变量的运算：

1. 整数运算

方法一：expr

expr 1 + 2

expr $num1 + $num2 + - * / %

方法二：$(())

echo $(($num1+$num2)) + - * / %

echo $((num1+num2))

echo $((5-3*2))

echo $(((5-3)*2))

echo $((2**3))

sum=$((1+2)); echo $sum

方法三：$[]

echo $[5+2] + - * / %

echo $[5**2]

方法四：let

let sum=2+3; echo $sum

let i++; echo $i

2. 小数运算

echo "2*4" |bc

echo "2^4" |bc

echo "scale=2;6/4" |bc

awk 'BEGIN{print 1/2}'

echo "print 5.0/2" |python

变量"内容"的删除和替换（扩展）

==="内容"的删除===

[root@tianyun ~]# url=www.sina.com.cn

[root@tianyun ~]# echo ${#url} 获取变量值的长度

15

[root@tianyun ~]# echo ${url} 标准查看

www.sina.com.cn

[root@tianyun ~]# echo ${url#*.} 从前往后，最短匹配

sina.com.cn

[root@tianyun ~]# echo ${url##*.} 从前往后，最长匹配 贪婪匹配
cn

[root@tianyun ~]# url=www.sina.com.cn
[root@tianyun ~]# echo ${url}
www.sina.com.cn
[root@tianyun ~]# echo ${url%.*} 从后往前，最短匹配
www.sina.com
[root@tianyun ~]# echo ${url%%.*} 从后往前，最长匹配 贪婪匹配
www

[root@tianyun ~]# url=www.sina.com.cn
[root@tianyun ~]# echo ${url#a.}
www.sina.com.cn
[root@tianyun ~]# echo ${url#*sina.}
com.cn

[root@tianyun ~]# echo $HOSTNAME
tianyun.1000phone.com
[root@tianyun ~]# echo ${HOSTNAME%%.*}
tianyun

索引及切片

[root@tianyun ~]# echo ${url:0:5}
[root@tianyun ~]# echo ${url:5:5}

[root@tianyun ~]# echo ${url:5}

==="内容"的替换===
[root@tianyun ~]# url=www.sina.com.cn
[root@tianyun ~]# echo ${url/sina/baidu}
www.baidu.com.cn

[root@tianyun ~]# url=www.sina.com.cn
[root@tianyun ~]# echo ${url/n/N}
www.siNa.com.cn
[root@tianyun ~]# echo ${url//n/N} 贪婪匹配

www.siNa.com.cN

===变量的替代===

```
[root@tianyun ~]# unset var1
[root@tianyun ~]#
[root@tianyun ~]# echo ${var1}
[root@tianyun ~]# echo ${var1-aaaaa}
aaaaa

[root@tianyun ~]# var2=111
[root@tianyun ~]# echo ${var2-bbbbb}
111
[root@tianyun ~]#
[root@tianyun ~]# var3=
[root@tianyun ~]# echo ${var3-ccccc}
```

${变量名-新的变量值}

变量没有被赋值：会使用“新的变量值“ 替代

变量有被赋值（包括空值）： 不会被替代

```
[root@tianyun ~]# unset var1
[root@tianyun ~]# unset var2
[root@tianyun ~]# unset var3
[root@tianyun ~]#
[root@tianyun ~]# var2=
[root@tianyun ~]# var3=111
[root@tianyun ~]# echo ${var1:-aaaa}
aaaa
[root@tianyun ~]# echo ${var2:-aaaa}
aaaa
[root@tianyun ~]# echo ${var3:-aaaa}
111
```

${变量名:-新的变量值}

变量没有被赋值（包括空值）：都会使用“新的变量值“ 替代

变量有被赋值： 不会被替代

```
[root@tianyun ~]# echo ${var3+aaaa}
[root@tianyun ~]# echo ${var3:+aaaa}

[root@tianyun ~]# echo ${var3=aaaa}
[root@tianyun ~]# echo ${var3:=aaaa}
```

```
[root@tianyun ~]# echo ${var3?aaaa}
[root@tianyun ~]# echo ${var3:?aaaa}
```

i++ 和 ++i （了解）

对变量的值的影响：

```
[root@tianyun ~]# i=1
[root@tianyun ~]# let i++
[root@tianyun ~]# echo $i
2
[root@tianyun ~]# j=1
[root@tianyun ~]# let ++j
[root@tianyun ~]# echo $j
2
```

对表达式的值的影响：

```
[root@tianyun ~]# unset i
[root@tianyun ~]# unset j
[root@tianyun ~]#
[root@tianyun ~]# i=1
[root@tianyun ~]# j=1
[root@tianyun ~]#
[root@tianyun ~]# let x=i++  先赋值，再运算
[root@tianyun ~]# let y=++j  先运算，再赋值
[root@tianyun ~]#
[root@tianyun ~]# echo $i
2
[root@tianyun ~]# echo $j
2
[root@tianyun ~]#
[root@tianyun ~]# echo $x
1
[root@tianyun ~]# echo $y
2
```