02.Shell脚本变量

> 徐亮伟, 江湖人称标杆徐。多年互联网运维工作经验，曾负责过大规模集群架构自动化运维管理工作。擅长Web集群架构与自动化运维，曾负责国内某大型电商运维工作。
> 个人博客"徐亮伟架构师之路"累计受益数万人。
> 笔者Q:552408925、572891887
> 架构师群:471443208

简单的理解变量: 用一个固定的字符串去表示不固定的内容

```
[root@Shell ~]# a=1
[root@Shell ~]# b=2
[root@Shell ~]# echo $a
1
[root@Shell ~]# echo $b
2
```

1.变量常见类型

`Shell` 变量如下几类分为

> 自定义变量
> 系统环境变量
> 位置参数变量
> 预先定义变量

1.自定义变量

> 1.定义变量 `变量名=变量值 ,不允许数字命名, 不能使用横岗命名`
> 2.引用变量 `$变量名 或 ${变量名}`
> 3.查看变量 `echo $变量名 set显示所有变量,包括自定义变量和环境变量`
> 4.取消变量 `unset 变量名 作用范围:仅在当前 shell 中有效`

2.系统环境变量

> 1.定义环境变量`export export 变量 ,将自定义变量转换成环境变量`
> 2.引用环境变量`$变量名 或 ${变量名}`
> 3.查看环境变量`echo $变量名 env |grep Name`
> 4.取消环境变量`unset 变量名`
> 5.变量作用范围`在当前shell和子shell有效`

3.位置参数变量

脚本参数传参: `$1 $2 $3 $4 $5 $6 $7 $8 $9 ${10}`

4.预先定义变量

> `$0`脚本文件名
> `$*`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

```
//例，演示每个参数的作用
echo " 第 3 个位置参数是$3"
echo " 第 2 个位置参数是$2"
echo " 第 1 个位置参数是$1"
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: $$"
echo '$4='$4
echo '$5='$5
echo '$6='$6
echo '$*='$*
echo '$@='$@
echo '$#='$#
echo '$$='$$
```

2.变量赋值方式

1.显式赋值(变量名=变量值)

```
//例
ip1=192.168.56.11
school="Wuhan HuQuan"
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
# 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

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

3.注意事项: 定义或引用变量时注意事项: `" "`弱引用`' '`强引用

```
[root@bgx ~]# school=Iphone
[root@bgx ~]# echo "${school} is good"
Iphone is good
[root@bgx ~]# echo '${school} is good'
${school} is good
```

4.`命令替换等价于`$()`反引号中的`shell`命令会被先执行

```
[root@bgx ~]# touch `date +%F`_file1.txt
[root@bgx ~]# touch $(date +%F)_file2.txt
```

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

3.变量数值运算

1.整数运算 `expr + - \* / %`

```
expr 1 + 2
expr $num1 + $num2
```

2.整数运算 `$(()) + - * / %`

```
echo $(($num1+$num2))
echo $((num1+num2))
echo $((5-3*2))
echo $(((5-3)*2))
echo $((2**3))
sum=$((1+2)); echo $sum
```

3.整数运算 `$[] + - * / %`

```
echo $[5+2]
echo $[5**2]
```

4.整数运算 `let + - * / %`

```
let sum=2+3;
echo $sum
```

5.小数运算 `bc + - * / %`

```
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
```

4.变量删除替换

1.从前往后删除变量内容

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

//获取变量值的长度
[root@bgx ~]# echo ${#url}

//输出变量的长度
[root@bgx ~]# echo ${url}

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

//从前往后，最长匹配(贪婪匹配)
[root@bgx ~]# echo ${url##*.}
```

2.从后往前删除变量内容

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

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

[root@bgx ~]# echo $HOSTNAME
www.bgx.com
[root@bgx ~]# echo ${HOSTNAME%%.*}
bgx.com
```

3.索引及切片

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

4.变量内容替换

```
[root@bgx ~]# url=www.sina.com.cn
[root@bgx ~]# echo ${url/sina/baidu}

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

5.变量替代

> ${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

```
//例1
[root@Shell day01]# unset url
[root@Shell day01]# echo $url
[root@Shell day01]# url=www.sina.com
[root@Shell day01]# echo $url
www.sina.com
[root@Shell day01]# echo ${url-www.baidu.com}
www.sina.com
[root@Shell day01]# unset url
[root@Shell day01]# echo ${url-www.baidu.com}
www.baidu.com

//例2
[root@Shell day01]# url2=
[root@Shell day01]# echo ${url2-www.baidu.com}
```

6.变量替代

> ${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

```
//例1
[root@Shell day01]# unset url
[root@Shell day01]# echo $url

[root@Shell day01]# url=www.sina.com
[root@Shell day01]# echo $url
www.sina.com
[root@Shell day01]# echo ${url:-www.baidu.com}
```

```
www.sina.com
[root@Shell day01]# unset url
[root@Shell day01]# echo ${url:-www.baidu.com}
www.baidu.com

//例2
[root@Shell day01]# url2=
[root@Shell day01]# echo $url2

[root@Shell day01]# echo ${url2:-www.baidu.com}
www.baidu.com
```

5.变量自增

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

//对变量值不会产生任何影响
[root@bgx ~]# echo $i
2
[root@bgx ~]# echo $j
2

//对表达式的值的影响
[root@bgx ~]# echo $x
1
[root@bgx ~]# echo $y
2

对变量的值的影响：
[root@bgx ~]# i=1
[root@bgx ~]# let i++
[root@bgx ~]# echo $i 2

//实际案例
[root@Shell day01]# cat i++.sh
#!/usr/bin/bash

ip=192.168.56.11
i=1
while [ $i -le 5 ];do
        ping -c1 $ip &>/dev/null
        if [ $? -eq 0 ];then
```

```
                echo "$ip is up..."
        fi
        let i++
done
```