A+

shell变量详解

这篇文章总结了怎样在bash shell中创建变量、使用变量、删除变量，并且演示了本地变量（全局变量和局部变量）、环境变量、只读变量和特殊变量的用法。

创建变量

如何在当前bash中创建一个变量呢，直接使用 "变量名=变量值"方式即可，比如

```
[www.zsythink.net]#zsy=zsythink
[www.zsythink.net]#
[www.zsythink.net]#
```


我们创建了一个变量，变量名叫zsy，变量值为zsythink

引用变量

我们已经创建了一个变量，那么怎么引用这个变量的值呢。

引用变量：就是使用变量的意思，"${变量名}" 表示使用变量，示例如下

```
[www.zsythink.net]#zsy=zsythink
[www.zsythink.net]#echo ${zsy}
zsythink
[www.zsythink.net]#
[www.zsythink.net]#
```


上图中，我们先在命令行中创建了一个变量，然后引用了它，可以看到，我们使用echo命令输出了变量值。

大多数情况下，引用变量时，"${变量名}"中的"{}""可以省略，但是某些情况下不行。

我们先看看省略时是什么样子的。

```
[www.zsythink.net]#zsy="www.zsythink.net"
[www.zsythink.net]#
[www.zsythink.net]#echo $zsy
www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#
```


上图中，使用echo输出变量名时，我们并没有使用"${变量名}"的方法引用变量值，而是省略了"{}"，使用"$变量名"的方式引用了变量值，但是，某些情况不能省略"{}"呢，我来描述一下，比如，我们声明了一个animal变量，然后将animal变量的值设置为了pig，如下

```
[www.zsythink.net]#animal=pig
[www.zsythink.net]#
[www.zsythink.net]#echo $animal
pig
[www.zsythink.net]#
[www.zsythink.net]#
```


当我们引用animal变量的值的时候，其值为pig，而现在，我们想要使用echo命令，输出pigs ，那么，我们能不能直接在应用animal变量时直接加上s呢，我们试试

```
[www.zsythink.net]#animal=pig
[www.zsythink.net]#
[www.zsythink.net]#echo $animal
pig
[www.zsythink.net]#echo $animals

[www.zsythink.net]#
[www.zsythink.net]#
```


可以看到，这样是不行的，因为当我们这样引用变量时，系统会认为我们在引用一个新的变量"animals",如果想要达到我们的效果，在屏幕上输出pigs，我们则必须

名}"的方式引用变量值，示例如下

```
[www.zsythink.net]#animal=pig
[www.zsythink.net]#
[www.zsythink.net]#echo $animal
pig
[www.zsythink.net]#echo ${animal}s
pigs
[www.zsythink.net]#
[www.zsythink.net]#
```


撤销变量

如何撤销变量、释放变量呢。

我们可以把撤销变量理解成删除一个变量，我们使用unset撤销变量，语法如下

unset VARNAME

撤销变量示例如下：

```
[www.zsythink.net]#zsy=zsythink
[www.zsythink.net]#echo ${zsy}
zsythink
[www.zsythink.net]#unset zsy
[www.zsythink.net]#echo ${zsy}

[www.zsythink.net]#
[www.zsythink.net]#
```


可以看到，撤销变量后，再次使用echo命令输出变量值时，输出为"空"，因为对应的变量已经不存在了。

如果我们在脚本中定义了变量，当脚本执行完成后，脚本中的变量自动会被撤销，虽然说脚本在执行完成后，脚本中生成的变量会被释放，但是，最好还是养成及时
并且无再次使用可能的变量，及时撤销变量是良好的变成习惯，就像其他语言的开发中，关闭已经不再使用的数据库连接一样，如果某些脚本需要常驻后台执行，最
量是否及时释放。

以后我们还会用到函数，unset也可以撤销函数，当某个函数使用过后，不需要再次使用的时候，可以撤销此函数，使用如下语法撤销函数

unset -f 函数名

好了，创建变量、使用变量、撤销变量的操作我们已经说完了，那么我们说说变量的类型。

bash是弱类型的语言，默认会把变量值当做字符串处理，所以，我们可以认为bash中的变量都是"字符串"。

变量是存在作用域的概念的，按照作用域划分，一般可以划分为本地变量，环境变量，除了这两种变量，还有只读变量和一些特殊变量，那么，它们有什么不同呢，
下。

本地变量

下图的方式其实就是在当前bash中创建一个本地变量，我们之间创建的变量，也是本地变量。

```
[www.zsythink.net]#zsythink='www.zsythink.net'
[www.zsythink.net]#
[www.zsythink.net]#
```


本地变量的作用域：本地变量只在在当前bash进程中有效，对当前shell之外的其它shell进程，包括当前shell的子shell进程均无效。

本地变量中还可以细分为 局部变量 和 全局变量。

在以后的总结中，我们还会学习到函数，可能会在函数中定义局部变量，局部变量创建方法如下。

local varname=value

局部变量的作用域：局部变量只对当前函数或代码段有效。

如果脚本中存在全局变量var（注意：是脚本范围内的全局变量，不是bash命令行中），同时，脚本中某个函数中如果也有一个变量命名为var并且被赋值，那么函数
将会覆盖全局变量var的值，如果函数中声明var变量时，使用了local关键字，那么，函数中的local var的值 ，将不会覆盖全局变量var的值，而且在函数中使用的va
local var的值，函数外使用var变量的值仍是全局变量的var的值。

那么我们用一段代码，演示上面的理论总结，如果你无法理解示例中的代码，没有关系，我们只是用它印证上面的话，不能理解就跳过此示例即可。

```
[www.zsythink.net]#cat test.sh
#!/bin/bash
#
zsythink="www.zsythink.net"
echo ${zsythink}

testvarzone() {
        zsythink="zsy's blog @ www.zsythink.net";
        echo $zsythink;
}

testvarzone
echo $zsythink
[www.zsythink.net]#
[www.zsythink.net]#./test.sh
www.zsythink.net
zsy's blog @ www.zsythink.net
zsy's blog @ www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#vim test.sh
[www.zsythink.net]#cat test.sh
#!/bin/bash
#
zsythink="www.zsythink.net"
echo ${zsythink}

testvarzone() {
        local zsythink="zsy's blog @ www.zsythink.net";
        echo $zsythink;
}

testvarzone
echo $zsythink
[www.zsythink.net]#
[www.zsythink.net]#./test.sh
www.zsythink.net
zsy's blog @ www.zsythink.net
www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#
[www.zsythink.net]#
```


从上图中的实验可以看出，如果当zsythink变量并不是局部变量时，当我们第二次对zsythink变量赋值以后，相当于重新赋值了，而当一个变量被设置为局部变量时量的作用域只限制在当前函数内，所以，局部变量zsythink并不会覆盖脚本内的zsythink全局变量。这就是在脚本范围内，局部变量与全局变量的区别。

当我们在当前shell执行一个脚本的时候，其实是在当前shell进程中生成了一个子进程，在子进程中运行的脚本。

举个例子：

我们直接在当前bash下运行test.sh脚本

```
[www.zsythink.net]#cat ./test.sh
#!/bin/bash
#
sleep 50
[www.zsythink.net]#
[www.zsythink.net]#./test.sh
```


打开另一个终端，使用pstree命令查看进程树，发现如下图

```
├─sshd─┬─sshd───bash───test.sh───sleep
      └─sshd───bash───pstree
```

上图中篮筐中的bash为刚才执行./test.sh命令的bash，当我们执行test.sh的时候，则生成一个子进程（红框中的test.sh），在子进程中运行test.sh

如果我们在"篮筐中的bash"中定义了本地变量，那么使用上述方式直接运行test.sh的时候，test.sh在运行时则不能够直接使用"篮筐中bash"定义的本地变量，因为地变量只在在当前bash进程中有效，对当前shell之外的其它shell进程，包括当前shell的子shell进程均无效，虽然红框中的test.sh进程为篮筐中bash进程的子进程法使用篮筐中bash进程的本地变量，我们来做个试验。

在当前bash中创建一个本地变量vtest，在当前bash中直接运行test.sh

```
[www.zsythink.net]#zsythink="www.zsythink.net"
[www.zsythink.net]#echo $zsythink
www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#cat test.sh
#!/bin/bash
#
echo ${zsythink}
[www.zsythink.net]#./test.sh

[www.zsythink.net]#
[www.zsythink.net]#
```


红色横线以上的部分表示我们在当前bash进程中输出了本地变量，而在test.sh脚本中，我们同样写了一条echo命令，输出zsythink变量，我们发现，test.sh并没有变量中的内容，因为test.sh实际上是在当前bash的子进程中运行，而zsythink是定义在当前bash中的本地变量，验证了前面的本地变量作用域的概念。

如果想要test.sh脚本可以使用到当前bash中定义的变量，怎么办呢，有两种方式

1、使用另一种不是本地变量的"变量"，它被称作"环境变量"，我们可以在当前bash中定义环境变量，环境变量的具体演示在后面，不要着急。

2、在当前bash中，仍然使用"本地变量"，但是改变执行test.sh脚本的方式。

这里面牵扯到两个新概念"环境变量" 和 "执行脚本的方式"，我们向下看，慢慢聊。

环境变量

环境变量这个名词搞IT的同学应该都比较熟悉，我就不啰嗦了

环境变量的作用域：环境变量的生效范围为当前shell进程及其子进程

使用export关键字指明对应的变量为环境变量，方法如下

export varname=value

也可以先声明为本地变量，然后再导出为环境变量,步骤如下

zsythink="www.zsythink.net"

export zsythink

命令行中直接执行的shell脚本在执行时会启动一个子shell进程。

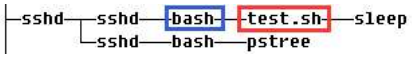
命令行中直接执行的shell脚本会继承当前shell的环境变量

系统自动执行的shell脚本（非命令行中执行）就需要自我定义需要的各种环境变量，或者导入一些已经存在的环境变量。

我们已经知道，本地变量只能被当前shell进程使用，而环境变量可以被子进程使用到。

我们还使用刚才在本地变量中使用过的例子，来套用到环境变量身上，看看，实际效果。

我们直接在当前bash下执行test.sh脚本的时候，其实是在当前shell进程中生成了一个子进程，在子进程中运行的test.sh脚本。



那么，我们在当前bash中定义一个环境变量，然后直接在当前bash中执行test.sh脚本，看看test.sh脚本能不能识别到，继承到环境变量。

```
[www.zsythink.net]#export zsythink="www.zsythink.net"
[www.zsythink.net]#cat ./test.sh
#!/bin/bash
#
echo ${zsythink}

[www.zsythink.net]#./test.sh
www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#
```


我们发现，test.sh输出了zsythink变量中的内容，因为test.sh实际上是在当前bash的子进程中运行，而zsythink是定义在当前bash中的环境变量，验证了环境变量概念。

有的人就是记性好，你是不是还记得，我们说过，想要让test.sh可以使用到当前bash中定义的变量，除了定义环境变量这种方法，还有另一种方法，就是改变执行t方式，这句话是什么意思，怎么做，我们向下看。

告诉你一个秘密，在当前bash中，除了使用"路径+名称脚本"的方式直接运行脚本以外，还能使用另一种方法运行脚本，就是使用source关键字，我们只要在原有的source关键字即可，如下图。

```
[www.zsythink.net]#cat test.sh
#!/bin/bash
#
echo "www.zsythink.net";
sleep 50
[www.zsythink.net]#
[www.zsythink.net]#source ./test.sh
www.zsythink.net
```


细心的你已经发现，脚本已经执行了，而且按照我们的指令，睡眠50秒，趁着他睡眠的功夫，我们打开另一个终端，查看一下当前系统中的进程树

```
├─sshd─┬─sshd───bash───sleep
      └─sshd───bash───pstree
```

我们发现，这样执行脚本，似乎和不加source直接执行脚本时显示的进程不太一样。

我们来对比一下：

加source关键字的时候，执行脚本时，进程树如下

```
├─sshd─┬─sshd───bash───sleep
      └─sshd───bash───pstree
```

不加source关键字，直接运行脚本的时候，进程树如下，

```
├─sshd─┬─sshd───bash───test.sh───sleep
      └─sshd───bash───pstree
```

细心的你又发现了，不加source关键字的时候，执行脚本，相当于创建了一个子进程，然后在子进程中运行脚本，如果加了source关键字，相当于没有创建子进程当前bash中运行了脚本。

我似乎想到了什么……

本地变量只能在当前shell进程中生效，而当我们在执行脚本的时候如果加入了source关键字，那么脚本就相当于在当前进程中执行，而不是在子进程中执行，那么rce的方式执行脚本，理论上来说，本地变量是可以被脚本使用的，因为它们都在一个进程中，我们来验证一下。

```
[www.zsythink.net]#zsy="www.zsythink.net"
[www.zsythink.net]#
[www.zsythink.net]#cat test.sh
#!/bin/bash
#
echo "$zsy";
[www.zsythink.net]#
[www.zsythink.net]#./test.sh

[www.zsythink.net]#source ./test.sh
www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#
```


如上图，使用两种方式执行脚本，得到的结果不一样，验证了我们的想法。

此处还要说明一点source /some/file相当于 ". /some/file"

". /some/file"用文字描述就是"点 空格 脚本路径"，这样说是为了让你区分出如下两种写法：

"./some/file" 和 ". /some/file"

他们长得非常像，所以此处向新手同学强调一下。

其实 source /some/file 可以理解为将/some/file文件中的内容包含到当前文件或者进程中。

"."的作用和source的作用相同。

需要注意一点，先看示例，如下。

```
[www.zsythink.net]#export zsy="zsy"
[www.zsythink.net]#zsy="www.zsythink.net"
[www.zsythink.net]#echo $zsy
www.zsythink.net
[www.zsythink.net]#
[www.zsythink.net]#
```


如上写法表示声明一个环境变量，然后重新对这个环境变量赋值，并不能表示它从一个环境变量变成了本地变量，如果想把它从本地变量变成环境变量，需要先uns新声明为本地变量并赋值。

只读变量

只读变量，顾名思义，就是设置了以后只能读不能改。

可以把只读变量理解成一个常量。

使用如下方法设置一个只读变量

readonly rovarname=value

例如

readonly pai=3.1415926

只读变量设置后不可修改，不可unset（撤销），如果想要只读变量失效需要退出当前shell。

```
[www.zsythink.net]#readonly zsy='www.zsythink.net'
[www.zsythink.net]#unset zsy
-bash: unset: zsy: cannot unset: readonly variable
[www.zsythink.net]#
[www.zsythink.net]#
```

zsythink.net 朱双印博客

只读变量的只对当前bash生效，"子bash"无法继承当前bash的只读变量，如果想要当前bash的"子bash"或者"孙bash"也能继承当前bash的只读变量，那么需要将环境变量的一种，变成"环境只读变量",使用如下方法，声明一个"环境只读变量"

export readonly varname=value

"环境只读变量"可以被子bash继承到后直接使用。

特殊变量

特殊变量往往在脚本中使用，我们会在以后的脚本示例总结中对它们演示，此处只是总结出来，方便有经验的朋友回忆，看不懂没关系，我们会在以后的应用中应用候自然会明白。

特殊变量：$?

上述特殊变量保存了上一个命令的执行状态返回值。

命令执行后，可能有两类状态返回值（0 - 255）

如果返回值为0：表示上一条命令正确执行

如果返回值为1-255中的任意一个：表示上一条命令执行错误

1到255中，1、2、127 为系统预留的错误状态码，其他状态码可自定义。

位置变量

脚本中的特殊变量

$# 表示传入脚本的参数个数，参数数量

$*参数列表 同 $@

$@参数列表，获取到所有参数

${@:起点} 表示由起点开始（包括起点），取得后面的所有的位置参数

${@:起点:个数} 表示由起点开始（包括起点），取得指定个数的位置参数

当我们直接使用$*和$@的时候，这两种写法没有任何区别，

但是，在对$* 和$@加引号后，变成了"$*"和"$@", 这两种写法就会产生区别

$@ $* 只在被双引号包起来的时候才会有差异

"$*": 传递给脚本的所有参数，全部参数合为一个字符串

"$@": 传递给脚本的所有参数，每个参数为独立字符串

$0表示脚本本身，相当于basename输出的内容

如下特殊变量用来在脚本内引用传入脚本的参数值。

$1 , $2 .....

使用shift 可剔除1个参数

shift -n 一次同时剔除最前面的n个参数，踢出头后面的参数自动向前排

shift表示上档

Shift 命令还有另外一个重要用途, Bash 定义了9个位置变量，从 $1 到 $9,这并不意味着用户在命令行只能使用9个参数，借助 shift 命令可以访问多于9个的参数。S
移动参数的个数由其所带的参数指定。例如当 shell 程序处理完前九个命令行参数后，可以使用 shift 9 命令把 $10 移到 $1 。

如果直接使用shift 不加任何数字，那么代表最前面的$1被剔出，$1被踢出后，当前的$2变成了$1,我们可以通过这种方法，每次只处理第一个所谓的"$1"。

如果你直接使用 $25 那么不会输出第25个参数，因为一共就9个参数位，如果直接输入$25，会输出第2个参数的内容后多出一个5，假如$2的值是a,那么，直接使用
5。

告诉你一个秘密，使用如下写法不用上档

```
[root@cos72ini testdir]#
[root@cos72ini testdir]# cat argnum.sh
#!/bin/bash
#
echo "1arg is" ${1}
echo "2arg is" $2
echo "3arg is" $3
echo "4arg is" $4
echo "5arg is" $5
echo "6arg is" $6
echo "7arg is" $7
echo "8arg is" $8
echo "9arg is" $9
echo "10arg is" ${10}
echo "11arg is" ${11}
echo "12arg is" ${12}
echo "13arg is" ${13}
echo "14arg is" ${14}
echo "15arg is" ${15}
echo "16arg is" ${16}
echo "17arg is" ${17}
echo "18arg is" ${18}
echo "19arg is" ${19}
echo "20arg is" ${20}
echo "21arg is" ${21}
echo "22arg is" ${22}
echo "23arg is" ${23}
echo "24arg is" ${24}
echo "25arg is" ${25}
echo "26arg is" ${26}
[root@cos72ini testdir]# ./argnum.sh a b c d e f e h i g k l m n o p q r s t u v w x y z
1arg is a
2arg is b
3arg is c
4arg is d
5arg is e
6arg is f
7arg is e
8arg is h
9arg is i
10arg is g
11arg is k
12arg is l
13arg is m
14arg is n
15arg is o
16arg is p
17arg is q
18arg is r
19arg is s
20arg is t
21arg is u
22arg is v
23arg is w
24arg is x
25arg is y
26arg is z
[root@cos72ini testdir]#
```


位置变量使用${}的方式表示，不用上档。

位置变量不能被继承 ，只有环境变量可以被继承。

其他创建变量的方法

也可以使用declare声明一个变量

declare abc=123

-i 表示为声明的变量为整形,此处说的整形与前面提到的若类型并不矛盾，它本质仍然是字符串，只是方便我们以后把它当做数字，对它进行运算。

declare -i sum=0

-x 表示为声明的变量为环境变量

declare -x expvarname=

declare -x expvarname= 这样的写法相当于 export expvarname=

declare -r name 声明一个只读变量

这些选项可以混合使用，比如声明一个可以被继承的"环境只读变量"

declare -rx zsy="www.zsythink.net"

我的微信公众号

关注"实用运维笔记"微信公众号，当博客中有新文章时，可第一时间得知哦~

Shell (bash)