详解Shell脚本中“$”符号的多种用法

通常情况下，在工作中用的最多的有如下几项：

- $0：Shell 的命令本身
- 1到9：表示 Shell 的第几个参数
- $? ：显示最后命令的执行情况
- $#：传递到脚本的参数个数
- $$：脚本运行的当前进程 ID 号
- $*：以一个单字符串显示所有向脚本传递的参数
- $!：后台运行的最后一个进程的 ID 号
- $-：显示 Shell 使用的当前选项
- ......

今天将通过以上几种选项并做进一步的操作案例；

1、引用变量

引用变量时，使用 $ 符号直接来进行引用，以及包括循环变量；

```
[root@localhost ~]# x=1024
[root@localhost ~]# echo $x
1024
```

利用双引号 " 将括起来的字符串支持变量插值。

```
[root@localhost ~]# x=1024
[root@localhost ~]# echo "x = $x"
x = 1024
```

使用 ${ } 作为单词边界。

```
[root@localhost ~]# x=1024
[root@localhost ~]# echo "x = ${x}xy"
x = 1024xy
```

使用 ${#} 获取变量字符串长度。

```
[root@localhost etc]# s=helloworld
[root@localhost etc]# echo "s.length = ${#s}"
s.length = 10
```

2、引用脚本或函数参数

基于引用脚本的方式，1 表示 Shell 脚本文件名，n 从 2 开始表示第 n 个参数，第 2 个参数是 $2；

```
[root@localhost ~]# echo 'echo $1 $2 $3' > ping.sh
[root@localhost ~]# cat ping.sh
echo $1 $2 $3
[root@localhost ~]# sh ping.sh 1 2 3
1 2 3
```

单引号 ' 括起来的字符串不会进行插值，并使用 $# 获取脚本或函数参数的个数；

```
[root@localhost ~]# echo 'echo $#' > ping.sh
[root@localhost ~]# sh ping.sh 1 2 3
3
```

3、上条命令的返回值

使用 $? 上条命令的返回值。

0：表示没有错误，其他任何数值：表示有错误。

```
[root@localhost ~]# true 1024
[root@localhost ~]# echo $?
0
[root@localhost ~]# false 2048
[root@localhost ~]# echo $?
1
```

4、执行并获取命令输出

使用 $() 执行并获取命令输出赋值给变量，等于双引号的功能。

```
[root@localhost ~]# echo `date`
2016年 06月 05日 星期日 12:39:08 CST
[root@localhost ~]# echo $(date)
2016年 06月 05日 星期日 12:39:34 CST
```

5、表达式求值

使用 []对表达式进行求值，与命令expr不同的是：[ ] 用于插值，则 expr 用于将值进行输出。

```
[root@localhost ~]# echo $[1024 + 2048]
3072
[root@localhost ~]# expr 1024 + 2048
3072
[root@localhost ~]# a=1024
[root@localhost ~]# b=2048
[root@localhost ~]# echo $[ a + b ]
3072
```

6、获取当前进程 ID

使用 $$ 来进行获取当前进程的 ID 号。

```
[root@localhost ~]# echo $$
55580
```

7、后台运行的最后一个进程 ID

使用 $! 来进行获取后台运行的最后一个进程 ID。

在命令结尾使用 & 可创建后台进程。

执行命令 kill !然后在输入echo! 将终止该 ping.sh 脚本。

```
[root@localhost ~]# tail -f /root/ping.sh &
[2] 55848
[root@localhost ~]# echo $!
55848
[root@localhost ~]# kill $!
[root@localhost ~]# echo $!
55848
[2]+ 已终止 tail -f /root/ping.sh
```

8、获取 Shell 选项

使用 $- 来进行获取当前 Shell 的选项。

```
[root@localhost ~]# echo $-
himBH
```

到此这篇关于详解Shell 脚本中 '$' 符号的多种用法的文章就介绍到这了,更多相关Shell $用法内容请搜索以前的文章或继续浏览下面的相关文章希望大家以后多多支持!