目录

[1、 rz工具 2](#_Toc518919278)

[2、 History命令 3](#_Toc518919279)

[2.1 History命令的变量说明 3](#_Toc518919280)

[2.2 参数说明 4](#_Toc518919281)

[3、 Head 5](#_Toc518919282)

[4、 Tail命令 6](#_Toc518919283)

[5、 Sort 7](#_Toc518919284)

[6、 Uniq 9](#_Toc518919285)

[7、 tr 11](#_Toc518919286)

[8、 grep 12](#_Toc518919287)

[9、 Find 15](#_Toc518919288)

[10、 awk 15](#_Toc518919289)

[11、 sed 15](#_Toc518919290)

[12、 sed、awk统合应用 15](#_Toc518919291)

## 变量使用总结

1、··和$()都可以用于将命令执行结果给变量或用于命令行参数

[root@mysqldb ~]# DIR=`pwd`

[root@mysqldb ~]# echo $DIR

/root

[root@mysqldb ~]# DIR1=$(pwd)

[root@mysqldb ~]# echo $DIR1

/root

[root@mysqldb ~]#

2、[]的两端必须要有一个空格

[root@mysqldb scripts]# [ -f /root/ab.py ] && { echo "/root/ab.py is file"; }

/root/ab.py is file

3、awk中引用shell变量使用”’$A’”双引号包含单引号

[root@mysqldb scripts]# DIR=`pwd`

[root@mysqldb scripts]# awk 'BEGIN{print "'$DIR'"}'

/server/scripts

[root@mysqldb scripts]# awk 'BEGIN{print "$DIR"}'

$DIR

[root@mysqldb scripts]#

4、$@和@\*都是显示所有参数，区别在于

[root@mysqldb scripts]# cat tt1.sh

#!/bin/bash

echo '$@ display:'

for i in "$@" ;do echo "$i";done;

echo

echo '$\* Display:'

for a in "$\*" ;do echo "$a";done;

[root@mysqldb scripts]#

[root@mysqldb scripts]# sh /server/scripts/tt1.sh tang hong tao "t h t"

$@ display:

tang

hong

tao

t h t

$\* Display: ##对$\*变量加双引号后将会把所有参数当成一个参数

tang hong tao t h t

[root@mysqldb scripts]#

[root@mysqldb scripts]# set -- tang hong tao "t h t"

[root@mysqldb scripts]# for i in $@; do echo $i ; done; ##对$@变量不加双引号后将会把所有参数按空格切分

tang

hong

tao

t

h

t

[root@mysqldb scripts]# for i in $\*; do echo $i ; done; ##对$\*变量不加双引号后将会把所有参数按空格切分

tang

hong

tao

t

h

t

[root@mysqldb scripts]# for i in "$\*"; do echo $i ; done; ##对$\*变量加双引号后将会把所有参数当成一个参数

tang hong tao t h t

[root@mysqldb scripts]# for i in "$@"; do echo $i ; done; ##对$@变量加双引号后才会按要求输出

tang

hong

tao

t h t

[root@mysqldb scripts]#

5、$$用于获当前脚本的PID，此变量用于保证脚本只有个进行在执行的场景

#!/bin/bash

if [ -f /tmp/pid1.pid ]

then

kill `cat /tmp/pid1.pid` >/dev/null 2>&1

rm -f /tmp/pid1.pid

fi

echo $$ >/tmp/pid1.pid

sleep 300

6、$?用于确认上一条命令是否执行成功，如成功返回0，不成功返回非0

[root@mysqldb scripts]# cd /etc/

[root@mysqldb etc]# echo $?

0

[root@mysqldb etc]# cd /abc

-bash: cd: /abc: No such file or directory

[root@mysqldb etc]# echo $?

1

[root@mysqldb etc]#

7、$\_

8、$!

9、$#

10、$0

## 数学运算

shell中的赋值和操作默认都是字符串处理，在此记下shell中进行数学运算的几个特殊方法，以后用到的时候可以来看，呵呵。  
   1、错误方法举例  
   a)  
   var=1+1  
   echo $var  
   输出的结果是1+1，悲剧，呵呵  
     
   b)  
   var=1  
   var=$var+1  
   echo $var  
   输出结果是1+1，依然悲剧，呵呵  
     
   2、正确方法  
   1)使用let  
   var=1  
   let "var+=1"  
   echo $var  
   输出结果为2，这次没有悲剧  
   注意：  
   a)经我测试let几乎支持所有的运算符，在网上看到一篇文章说“let不支持++、--和逗号、(、)”,但经我测试自加、自减、以及括号的优先级都得到了很好的支持  
   b)方幂运算应使用“\*\*”  
   c)参数在表达式中直接访问，不必加$  
   d)一般情况下算数表达式可以不加双引号，但是若表达式中有bash中的关键字则需加上  
   e)let后的表达式只能进行整数运算  
  
   2)使用(())  
   var=1  
   ((var+=1))  
   echo $var  
   输出结果为2  
   注意：  
   (())的使用方法与let完全相同  
     
   3)使用$[]  
   var=1  
   var=$[$var+1]  
   echo $var  
   输出结果位2  
   注意：  
   a)$[]将中括号内的表达式作为数学运算先计算结果再输出  
   b)对$[]中的变量进行访问时前面需要加$  
   c)$[]支持的运算符与let相同，但也只支持整数运算  
     
   4)使用expr  
   var=1  
   var=`expr $var + 1`  
   echo $var  
   输出结果为2  
   注意：  
   a)expr后的表达式个符号间需用空格隔开  
   b)expr支持的操作符有： |、&、<、<=、=、!=、>=、>、+、-、\*、/、%  
   c)expr支持的操作符中所在使用时需用\进行转义的有：|、&、<、<=、>=、>、\*  
   e)expr同样只支持整数运算  
     
   5)使用bc(可以进行浮点数计算)  
   var=1  
   var=`echo "$var+1"|bc`  
   echo $var  
   输出结果为2  
   介绍：  
   bc是linux下的一个简单计算器，支持浮点数计算，在命令行下输入bc即进入计算器程序，而我们想在程序中直接进行浮点数计算时，利用一个简单的管道即可解决问题。  
   注意：  
   1)经我测试bc支持除位操作运算符之外的所有运算符。  
   2)bc中要使用scale进行精度设置  
   3)浮点数计算实例  
   var=3.14  
   var=`echo "scale=2;$var\*3"|bc`  
   echo $var  
   输出结果为9.42  
     
   6)使用awk(可已进行浮点数计算)  
   var=1  
   var=`echo "$var 1"|awk '{printf("%g",$1\*$2)}'`  
   echo $var  
   输出结果为2  
   介绍：  
   awk是一种文本处理工具，同时也是一种程序设计语言，作为一种程序设计语言，awk支持多种运算，而我们可以利用awk来进行浮点数计算，和上面bc一样，通过一个简单的管道，我们便可在程序中直接调用awk进行浮点数计算。  
   注意：  
   1)awk支持除微操作运算符之外的所有运算符  
   2)awk内置有log、sqr、cos、sin等等函数  
   3)浮点数计算实例  
   var=3.14  
   var=`echo "$var 2"|awk '{printf("%g",sin($1/$2))}'`  
   echo $var  
   输出结果为1