1. set –e //这行代码之后的任何代码，如果返回一个非0的值，那么整个脚本立即退出，官方的说明是为了防止错误出现滚雪球的现象
2. set -o pipefail //告诉 bash 返回从右到左第一个以非0状态退出的管道命令的返回值，如果所有命令都成功执行时才返回0
3. 格式: echo "/033[字背景颜色;字体颜色m字符串/033[控制码"

如果单纯显示字体颜色可以固定控制码位0m。

格式: echo "/033[字背景颜色;字体颜色m字符串/033[0m"

字背景颜色范围:40 - 49

40:黑 41:深红 42:绿 43:黄色 44:蓝色 45:紫色 46:深绿 47:白色

字颜色:30 - 39

30:黑 31:红 32:绿 33:黄 34:蓝色 35:紫色 36:深绿 37:白色

ANSI控制码

\33[0m 关闭所有属性 \33[01m 设置高亮度

\33[04m 下划线 \33[05m 闪烁

\33[07m 反显 \33[08m 消隐

\33[30m -- \33[37m 设置前景色 \33[40m -- \33[47m 设置背景色

\33[nA 光标上移n行 \33[nB 光标下移n行

\33[nC 光标右移n行 \33[nD 光标左移n行

\33[y;xH设置光标位置 \33[2J 清屏

\33[K 清除从光标到行尾的内容 \33[s 保存光标位置

\33[u 恢复光标位置 \33[?25l 隐藏光标

\33[?25h 显示光标

如：echo -e "\033[34m\033[05m 请确认您的操作,输入 [Y/N]\033[0m"

1. 条件表达式

if ....; then

....

elif ....; then

....

Else

....

fi

文件表达式：

if [ -f file ] 如果文件存在

if [ -d ... ] 如果目录存在

if [ -h file ] 如果 FILE 存在且是一个符号连接则为真。

if [ -s file ] 如果文件存在且非空

if [ -r file ] 如果文件存在且可读

if [ -w file ] 如果文件存在且可写

if [ -x file ] 如果文件存在且可执行

if [ -e file ] 文件或目录名存在

整数变量表达式：

if [ int1 -eq int2 ] 如果int1等于int2

if [ int1 -ne int2 ] 如果不等于

if [ int1 -ge int2 ] 如果>=

if [ int1 -gt int2 ] 如果>

if [ int1 -le int2 ] 如果<=

if [ int1 -lt int2 ] 如果<

字符串变量表达式：

If [ $a = $b ] 如果string1等于string2

字符串允许使用赋值号做等号

if [ $string1 != $string2 ] 如果string1不等于string2

if [ -n $string ] 如果string 非空(非0），返回0(true)

if [ -z $string ] 如果string 为空

if [ $sting ] 如果string 非空，返回0 (和-n类似)

如：

If空[空-z "$BARCELONA\_ROOT"空];then //空格特别重要,黄色代表空格

echo "Errot:BARCELONA\_ROOT is not defined!"

exit 1

fi

if [ ! -f "$1" ]; then

printf "Error: file does not exist.\n"

exit 1

fi

1. 变量说明:

$$  
Shell本身的PID（ProcessID）  
$!  
Shell最后运行的后台Process的PID  
$?  
最后运行的命令的结束代码（返回值）  
$-  
使用Set命令设定的Flag一览  
$\*  
所有参数列表。如"$\*"用「"」括起来的情况、以"$1 $2 … $n"的形式输出所有参数。  
$@  
所有参数列表。如"$@"用「"」括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。  
$#  
添加到Shell的参数个数  
$0  
Shell本身的文件名  
$1～$n  
添加到Shell的各参数值。$1是第1参数、$2是第2参数…。

1. 整数比较

-eq 等于,如:if ["$a" -eq "$b" ]

-ne 不等于,如:if ["$a" -ne "$b" ]

-gt 大于,如:if ["$a" -gt "$b" ]

-ge 大于等于,如:if ["$a" -ge "$b" ]

-lt 小于,如:if ["$a" -lt "$b" ]

-le 小于等于,如:if ["$a" -le "$b" ]

< 小于(需要双括号),如:(("$a" < "$b"))

<= 小于等于(需要双括号),如:(("$a" <= "$b"))

> 大于(需要双括号),如:(("$a" > "$b"))

>= 大于等于(需要双括号),如:(("$a" >= "$b"))

1. $(( ))的用途

用来作整数运算。在 bash 中，$(( ))的整数运算符号大致有这些：+ - \* / 加、减、乘、%

$ echo $(((a+b)/c))

1. ${ }的一些特异功能：

假设我们定义了一个变量为：

file=/dir1/dir2/dir3/my.file.txt

我们可以用${ }分别替换获得不同的值：

${file#\*/}：拿掉第一条/及其左边的字符串：dir1/dir2/dir3/my.file.txt

${file##\*/}：拿掉最后一条/及其左边的字符串：my.file.txt

${file#\*.}：拿掉第一个. 及其左边的字符串：file.txt

${file##\*.}：拿掉最后一个. 及其左边的字符串：txt

${file%/\*}：拿掉最后条/及其右边的字符串：/dir1/dir2/dir3

${file%%/\*}：拿掉第一条/及其右边的字符串：(空值)

${file%.\*}：拿掉最后一个. 及其右边的字符串：/dir1/dir2/dir3/my.file

${file%%.\*}：拿掉第一个. 及其右边的字符串：/dir1/dir2/dir3/my

记忆的方法为：

#是去掉左边(在鉴盘上#在$之左边)

%是去掉右边(在鉴盘上%在$之右边)

单一符号是最小匹配﹔两个符号是最大匹配。

${file:0:5}：提取最左边的5个字节：/dir1

${file:5:5}：提取第5个字节右边的连续5个字节：/dir2

1. cd -是返回到上一次的工作目录
2. shell读文件的方法

1）管道的方式：

cat /sites/linuxpig.com.txt |while read LINE

do

echo $LINE

done

2）利用重定向符<

while read LINE

do

echo $LINE

done < /sites/linuxpig.com.txt

1. 数值计算

方法1：declare –i

#declare -i sum=100+300+50

# echo $sum ==>#结果为：450

方法2，var=$ (( expression ))

foo=1

foo=$(($foo+1))

1. return=`[linux命令](https://www.baidu.com/s?wd=linux%E5%91%BD%E4%BB%A4&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4PH9WuH-BnvDsmhf1PWmz0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzPW0Yn1RsPWf)` 会将[linux命令](https://www.baidu.com/s?wd=linux%E5%91%BD%E4%BB%A4&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4PH9WuH-BnvDsmhf1PWmz0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzPW0Yn1RsPWf)的结果赋值给return  
    return=$([linux命令](https://www.baidu.com/s?wd=linux%E5%91%BD%E4%BB%A4&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4PH9WuH-BnvDsmhf1PWmz0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzPW0Yn1RsPWf))也是将linux命令的结果赋值给return的一种方式
2. Linux系统脚本的三种执行方式

1)、source命令用法：   source FileName

作用:在当前bash环境下读取并执行FileName中的命令。该filename文件可以无"执行权限"

注：该命令通常用命令“.”来替代。

如：source bash\_profile   . bash\_profile两者等效。

2)、sh和bash命令用法： sh FileName bash FileName

3)、./的命令用法：   ./FileName

作用:打开一个子shell来读取并执行FileName中命令。 该文件必须有可执行的权限。

source filename 与 sh filename 及./filename执行脚本的区别在那里呢？

1)、当shell脚本具有可执行权限时，用sh filename与./filename执行脚本是没有区别得。./filename是因为当前目录没有在PATH中，所有"."是用来表示当前目录的。

2)、sh filename 重新建立一个子shell，在子shell中执行脚本里面的语句，该子shell继承父shell的环境变量，但子shell新建的、改变的变量不会被带回父shell，除非使用export。

3)、source filename：这个命令其实只是简单地读取脚本里面的语句依次在当前shell里面执行，没有建立新的子shell。那么脚本里面所有新建、改变变量的语句都会保存在当前shell里面。

1. 位置参数可以用shift命令左移。比如shift 3表示原来的$4现在变成$1，原来的$5现在变成$2等等，原来的$1、$2、$3丢弃，$0不移动。不带参数的shift命令相当于shift 1。

若用户要求 Shell 在不知道位置变量个数的情况下，还能逐个的把参数一一处理，也就是在 $1 后为 $2,在 $2 后面为 $3 等。在 shift 命令执行前变量 $1 的值在 shift 命令执行后就不可用了。

until [ $# -eq 0 ]

do

echo "第一个参数为: $1 参数个数为: $#"

shift

done

执行以上程序x：$./shift.sh 1 2 3 4

第一个参数为: 1 参数个数为: 4

第一个参数为: 2 参数个数为: 3

第一个参数为: 3 参数个数为: 2

第一个参数为: 4 参数个数为: 1

1. EOF … End of File

EOD … End of Document

EOT … End of Text

EOS … End of String

1. other