Shell Script是一种弱类型语言，使用变量的时候无需首先声明其类型。**新的变量会在本地数据区分配内存进行存储**，这个变量归当前的Shell所有，任何子进程都不能访问本地变量。这些变量与环境变量不同，**环境变量被存储在另一内存区，叫做用户环境区**，这块内存中的变量可以被子进程访问。

chmod 改变一个或者多个文件的存取模式

-c，--changes只输出被改变文件的信息

-f，--silent，--quiet 当chmod不能改变文件模式时，不通知文件的用户

--help 输出帮助信息。

-R，--recursive可递归遍历子目录，把修改应到目录下所有文件和子目录

--reference=filename 参照filename的权限来设置权限

-v，--verbose 无论修改是否成功，输出每个文件的信息

--version 输出版本信息。

who

u用户

g组

o其它

a所有用户(默认)

+ 表示增加权限、- 表示取消权限、= 表示唯一设定权限。 = 重新分配权限

permission

r读

w写

x执行

s设置用户(或组)的ID号

t设置粘着位(sticky bit)，防止文件或目录被非属主删除

u 用户的当前权限

g 组的当前权限

o 其他用户的当前权限

$ chmod u+x file 给file的属主增加执行权限

$ chmod 751 file 给file的属主分配读、写、执行(7)的权限，给file的所在组分配读、执行(5)的权限，给其他用户分配执行(1)的权限

$ chmod u=rwx,g=rx,o=x file 上例的另一种形式

总结：

两种修改权限的方式：

1. 数字形式：4(读)、2(写)、1(执行)
2. 字母形式：r读 w写x执行

指令名称 : chown

使用权限 : root

chown 用户名:组名 文件路径（可以是绝对路径也可以是相对路径）

使用方式 : chown [-cfhvR] [--help] [--version] user[:group] file...

说明 : Linux/Unix 是多人多工作业系统，所有的档案皆有拥有者。利用 chown 可以将档案的拥有者加以改变。一般来说，这个指令只有是由系统管理者(root)所使用，一般使用者没有权限可以改变别人的档案拥有者，也没有权限可以将自己的档案拥有者改设为别人。只有系统管理者(root)才有这样的权限。

例1：chown root:root /tmp/tmp1

就是把tmp下的tmp1的用户名和用户组改成root和root（只修改了tmp1的属组）.

例2：chown -R root:root /tmp/tmp1

就是把tmp下的tmp1下的所有文件的属组都改成root和root。

chomd 和 chown

chown用来更改某个目录或文件的用户名和用户组的

chmod功能要比chown要强大。可更改文件所有属性和权限。只有管理员账户才有权限用此命令。

使用find和xargs

F i n d命令的一般形式为：

find pathname -options [-print -exec -ok]

让我们来看看该命令的参数：

pathname find命令所查找的目录路径。例如用.来表示当前目录，用/来表示系统根目录。

-print find命令将匹配的文件输出到标准输出。

-exec find命令对匹配的文件执行该参数所给出的s h e l l命令。相应命令的形式为' command' {} \;，注意{ }和\；之间的空格。

-ok 和-exec的作用相同，只不过以一种更为安全的模式来执行该参数所给出的s h e l l命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

使用exec或ok来执行shell命令

当匹配到一些文件以后，可能希望对其进行某些操作，这时就可以使用-exec选项。一旦find命令匹配到了相应的文件，就可以用-exec选项中的命令对其进行操作（在有些操作系统中只允许- exec选项执行诸如ls或ls -l这样的命令）。大多数用户使用这一选项是为了查找旧文件并删除它们。这里我强烈地建议你在真正执行r m命令删除文件之前，最好先用ls命令看一下，确认它们是所要删除的文件。

exec选项后面跟随着所要执行的命令，然后是一对儿{}，一个空格和一个\，最后是一个分号。

为了使用exec选项，必须要同时使用print选项。如果验证一下find命令，会发现该命令只输出从当前路径起的相对路径及文件名。

为了用ls -l命令列出所匹配到的文件，可以把ls -l命令放在find命令的-exec选项中，例如：

上面的例子中，find命令匹配到了当前目录下的所有普通文件，并在-exec选项中使用ls -l

命令将它们列出。

为了在/logs目录中查找更改时间在5日以前的文件并删除它们，可以用：

$ find logs -type f -mtime +5 -exec rm {} \;

记住，在shell中用任何方式删除文件之前，应当先查看相应的文件，一定要小心！

当使用诸如mv或rm命令时，可以使用-exec选项的安全模式。它将在对每个匹配到的文件进行操作之前提示你。在下面的例子中， find命令在当前目录中查找所有文件名以.LOG结尾、更改时间在5日以上的文件，并删除它们，只不过在删除之前先给出提示。

按y键删除文件，按n键不删除。

任何形式的命令都可以在-exec选项中使用。在下面的例子中我们使用grep命令。

首先匹配所有文件名为“passwd\*”的文件，例如passwd、passwd.old、passwd.bak，然后执行grep命令看看在这些文件中是否存在一个rounder用户。

find命令的例子

我们已经介绍了find命令的基本选项，下面给出find命令的一些其他的例子。

为了匹配$HOME目录下的所有文件，下面两种方法都可以使用：

$ find $HOME -print

$ find ~ -print

为了在当前目录中查找suid置位，文件属主具有读、写、执行权限，并且文件所属组的用

户和其他用户具有读和执行的权限的文件，可以用：

$ find . -type f -perm 4755 -print

为了查找系统中所有文件长度为0的普通文件，并列出它们的完整路径，可以用：

$ find / -type f -size 0 -exec ls -l {} \;

为了查找/var/logs目录中更改时间在7日以前的普通文件，并删除它们，可以用：

$ find /var/logs -type f -mtime +7 -exec rm {} \;

为了查找系统中所有属于audit组的文件，可以用：

$find / -name -group audit -print

xargs

在使用find命令的-exec选项处理匹配到的文件时，find命令将所有匹配到的文件一起传递给exec执行。不幸的是，**有些系统对能够传递给exec的命令长度有限制，**这样在find命令运行几分钟之后，就会出现溢出错误。错误信息通常是“参数列太长”或“参数列溢出”。这就是xargs命令的用处所在，特别是与find命令一起使用。**Find命令把匹配到的文件传递给xargs命令，而xargs命令每次只获取一部分文件而不是全部，不像-exec选项那样**。这样它可以先处理最先获取的一部分文件，然后是下一批，并如此继续下去。在有些系统中，**使用-exec选项会为处理每一个匹配到的文件而发起一个相应的进程**，并非将匹配到的文件全部作为参数一次执行；这样在有些情况下就会出现进程过多，系统性能下降的问题，因而效率不高；**而使用args命令则只有一个进程**。

另外，在使用xargs命令时，究竟是一次获取所有的参数，还是分批取得参数，以及每一次获取参数的数目都会根据该命令的选项及系统内核中相应的可调参数来确定。

让我们来看看xargs命令是如何同find命令一起使用的，并给出一些例子。

下面的例子查找系统中的每一个普通文件，然后使用xargs命令来测试它们分别属于哪类

文件：

下面的例子在整个系统中查找内存信息转储文件(core dump) ，然后把结果保存到

/tmp/core.log 文件中：

$ find . -name "core" -print | xargs echo "" >/tmp/core.log

下面的例子在/apps/audit目录下查找所有用户具有读、写和执行权限的文件，并收回相应的写权限：

$ find /apps/audit -perm -7 -print | xargs chmod o-w

在下面的例子中，我们用grep命令在所有的普通文件中搜索device这个词：

$ find / -type f -print | xargs grep "device"

在下面的例子中，我们用grep命令在当前目录下的所有普通文件中搜索DBO这个词：

$ find . -name **＊\**-type f -print | xargs grep "DBO"

注意，在上面的例子中，\用来取消find命令中的\*在shell中的特殊含义。

取值的时候要在变量名前加$，$variable\_name可以在引号中使用

a = "hello world"

echo "Hi, $as"

就不会输出“Hi, hello worlds”，而是输出“Hi，”。这是因为Shell**把$as当成一个变量**，而$as未被赋值，其值为空。正确的方法是：

echo "Hi, ${a}s"

输出“Hi, hello worlds”

单引号中的变量不会进行变量替换操作。