# 常用命令

## cd 切换目录

### 意义

切换当前目录 至 另一个目录

### 命令格式

cd [参数] [dirname]

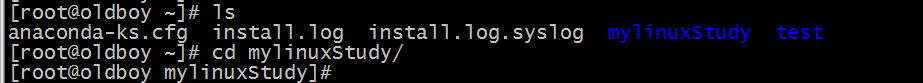
切换到 dirname 目录

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
|  |  |  |

### 例子

cd mylinuxStudy/ 切换到mylinuxStudy目录



### 使用技巧

cd 进入用户主目录；

cd ~ 进入用户主目录；

cd - 返回进入此目录之前所在的目录；

cd .. 返回上级目录（若当前目录为“/“，则执行完后还在“/"；".."为上级目录的意思）；

cd ../.. 返回上两级目录；

cd !$ 把上个命令的参数作为cd参数使用。

## ls 显示文件列表

### 意义

显示文件列表

### 命令格式

ls [参数]

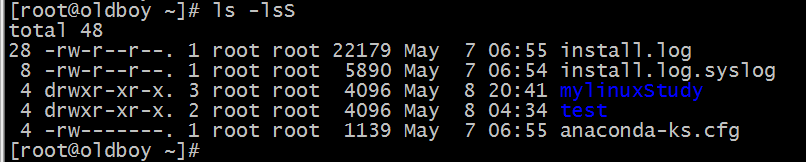
显示当前目录下的所有文件

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 | 注意 |
| -a | 列出文件下所有的文件，包括以“.“开头的隐藏文件 | （linux下文件隐藏文件是以.开头的，如果存在..代表存在着父目录）。 |
| -l | 列出文件的详细信息，如创建者，创建时间，文件的读写权限列表等等。 |  |
| -F | 在每一个文件的末尾加上一个字符说明该文件的类型。 | "@"表示符号链接、"|"表示FIFOS、"/"表示目录、"="表示套接字。 |
| -s | 在每个文件的后面打印出文件的大小 | size(大小) |
| -t | 按时间进行文件的排序 | Time(时间) |
| -p | 为目录添加上后缀”/” |  |
| -A | 列出除了"."和".."以外的文件。 |  |
| -R | 将目录下所有的子目录的文件都列出来，相当于我们编程中的“递归”实现 |  |
| -L | 列出文件的链接名。 | Link（链接） |
| -S | 以文件的大小进行排序 |  |
| -u | -l：显示访问时间，并按名字排序，否则按访问时间排序  -lt：排序 通过修改时间 |  |
| --time | --time-style=long-iso | 显示我们常用的时间格式 |

### 例子

ls -lsS 显示当前目录下的所有文件 显示内容包括 文件的属性（l） 文件的大小（s） 并且按照文件大小排序（S）

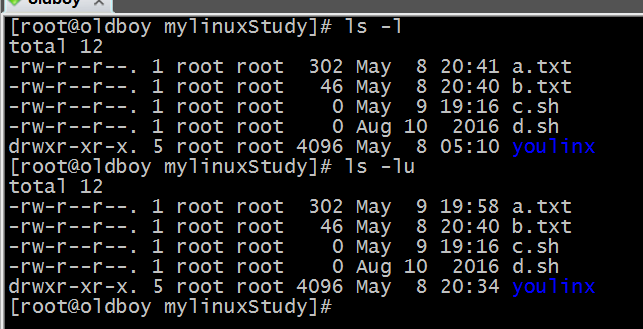


ls-l

显示文件的权限 修改时间

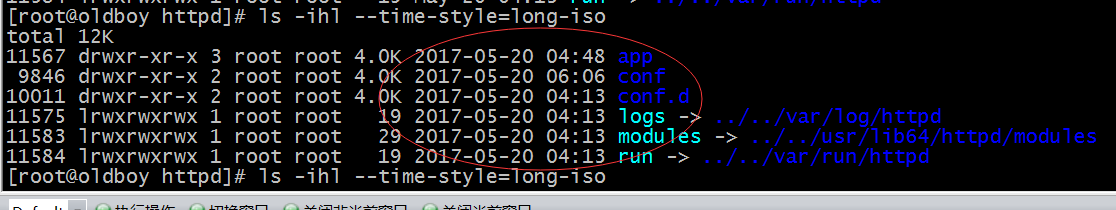
ls -lu

显示文件的权限 访问时间



显示我们常用的时间格式

ls -ihl --time-style=long-iso



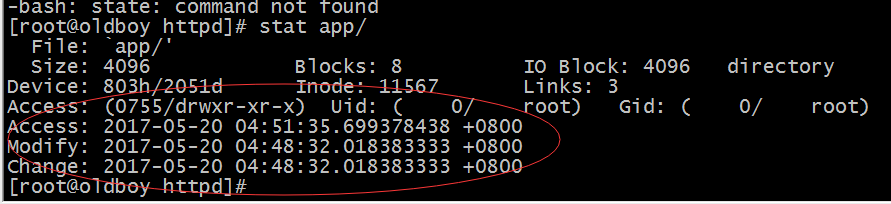
查看文件的时间属性

文件的时间属性有三个

访问时间

修改时间

改变时间（属性改变）



## cat 文本显示

### 意义

cat 是一个文本文件查看和连接工具。查看一个文件的内容，用cat比较简单，就是cat 后面直接接文件名。

### 简略说明

cat主要有三大功能：

1.一次显示整个文件。$ cat filename

2.从键盘创建一个文件。$ cat > filename

只能创建新文件,不能编辑已有文件.

3.将几个文件合并为一个文件： $cat file1 file2 > file

参数：

-n 或 --number 由 1 开始对所有输出的行数编号

-b 或 --number-nonblank 和 -n 相似，只不过对于空白行不编号

-s 或 --squeeze-blank 当遇到有连续两行以上的空白行，就代换为一行的空白行

-v 或 --show-nonprinting

例：

把 textfile1 的档案内容加上行号后输入 textfile2 这个档案里

cat -n textfile1 > textfile2

把 textfile1 和 textfile2 的档案内容加上行号（空白行不加）之后将内容附加到 textfile3 里。

cat -b textfile1 textfile2 >> textfile3

把test.txt文件扔进垃圾箱，赋空值test.txt

cat /dev/null > /etc/test.txt

### 命令格式

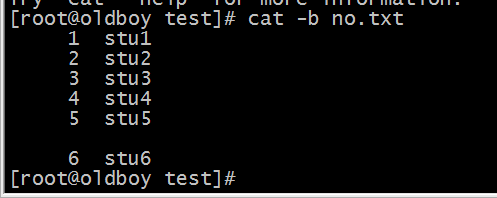
de>cat [参数选项] [文件]...de>

### 参数

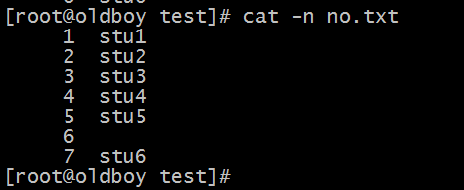
|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -A | -A, --show-all 等价于 -vET |  |
| -b | -b, --number-nonblank 对非空输出行编号 |  |
| -e | -e 等价于 -vE |  |
| -E | --show-ends 在每行结束处显示 $ |  |
| -n | --number 对输出的所有行编号 |  |
| -s | --squeeze-blank 不输出多行空行 | 如果有多个空行，合并成一行 |
| -t | 与 -vT 等价 |  |
| -T | --show-tabs 将跳 字符显示为 ^I |  |
| -u | -u (被忽略) |  |
| -v | --show-nonprinting 使用 ^ 和 M- 引用，除了 LFD 和 TAB 之外 |  |

### 例子

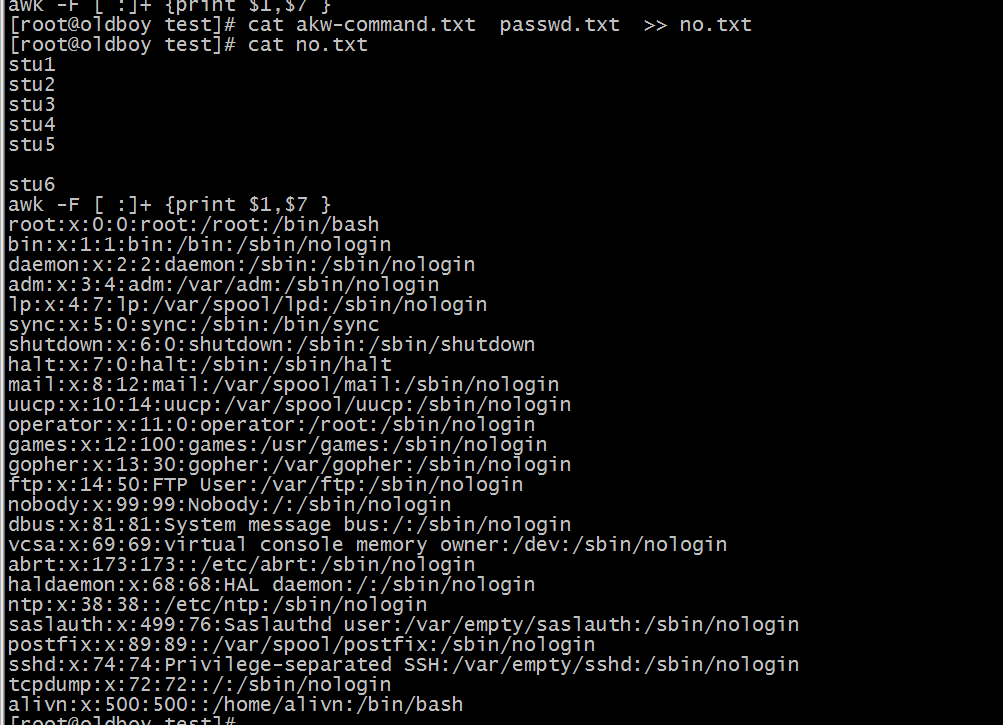
对no.txt文件的 非空行进行显示行号



对no.txt文件 显示行号 对非空行也进行编号



将多个文件追加到no.txt文件中去



## touch 建立文件

### 意义

创建文件 或者修改文件时间

### 命令格式

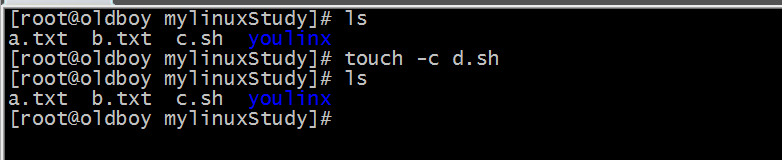
touch [参数] [filename]

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -a | 只更新访问时间，不改变修改时间 |  |
| -c | 不创建不存在的文件 |  |
| -m | 只更新修改时间，不改变访问时间 |  |
| -r file | 使用file文件的时间更新文件的时间 |  |
| -t | 将时间修改为参数指定的时间 | 07081556代表7月8号15点56分 |

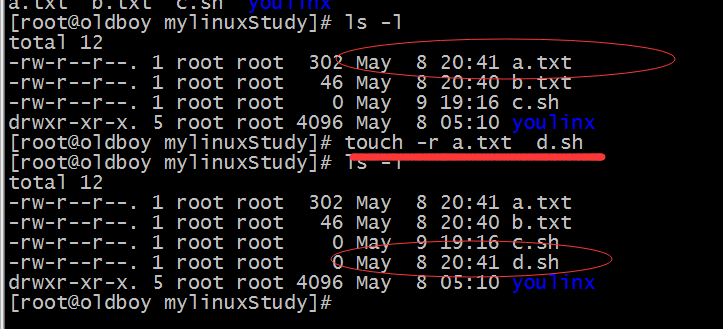
### 例子

touch -c d.sh 不会创建 d.sh



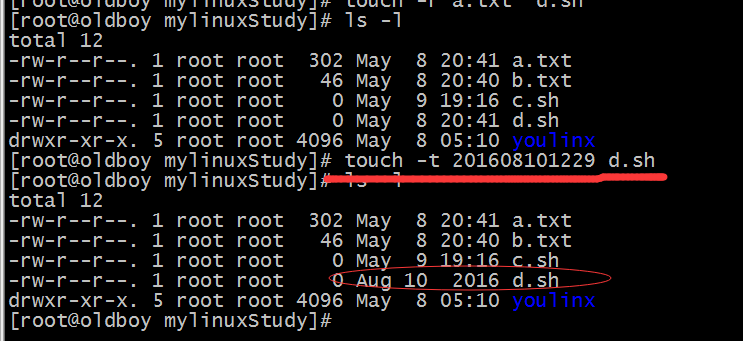
Touch -r a.txt d.sh

创建d.sh文件 并且 以把a.txt的文件日期作为d.sh的日期



touch -t 201608101229 d.sh

修改文件的日期，以 201608101229格式修改



## echo 打印 显示文字

### 意义

echo命令的功能是在显示器上显示一段文字，一般起到一个提示的作用。

### 命令格式

echo [参数] 字符串

将字符串输出到屏幕

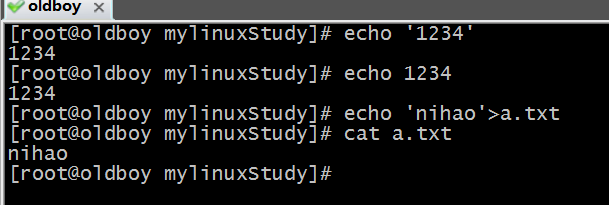
### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -n | 输出文字后 不换行 |  |
| -e | 若字符串中出现以下字符，则特别加以处理，而不会将它当成一般文字输出： | \a 发出警告声；  \b 删除前一个字符；  \c 最后不加上换行符号；  \f 换行但光标仍旧停留在原来的位置；  \n 换行且光标移至行首；  \r 光标移至行首，但不换行；  \t 插入tab；  \v 与\f相同；  \\ 插入\字符；  \nnn 插入nnn（八进制）所代表的ASCII字符； |

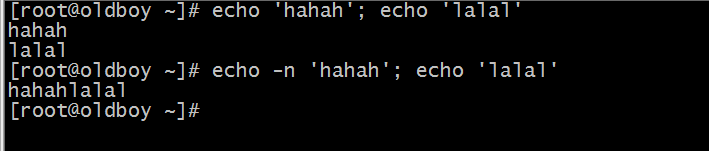
### 例子

echo ‘nihao’>a.txt

将nihao 重定向到 a.txt 文件



echo ‘hahah’; echo‘lalal’输出不换行



## mkdir 创建文件夹

### 意义

mkdir 创建目录 使用-p 参数可以创建多级目录

### 命令格式

mkdir 【参数】【目录】

创建目录

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -m | 设定权限<模式> (类似 chmod) | 一般不在创建目录的时候分配权限 |
| -p | 递归的创建目录，如果当前目录不存在，不会报错，而是创建目录 |  |
| -v | 每次创建目录都显示信息 |  |
|  |  |  |

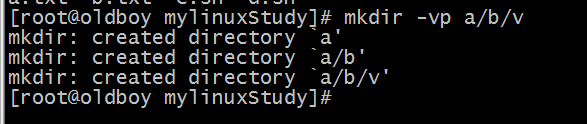
### 例子

mkdir -p a/b/c

创建a目录 b目录 c目录 如果a b目录不存在，不报错，直接建立

mkdir -vp a/b/c

显示目录创建过程



## mv 移动文件

### 意义

mv命令是move的缩写，可以用来移动文件或者将文件改名（move (rename) files），是Linux系统下常用的命令，经常用来备份文件或者目录。

### 命令格式

mv [参数] 源文件或目录 目标文件或目录创建目录

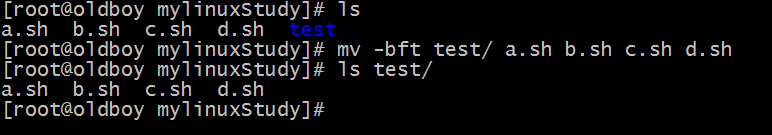
### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -b | 若需覆盖文件，则覆盖前先行备份 |  |
| -f | force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖 |  |
| -i | 若目标文件 (destination) 已经存在时，就会询问是否覆盖！ |  |
| -u | 若目标文件已经存在，且 source 比较新，才会更新(update) |  |
| -t | 即指定mv的目标目录，该选项适用于移动多个源文件到一个目录的情况，此时目标目录在前，源文件在后。 |  |

### 例子

mv -tbf test/ a.sh b.sh

将a.sh b.sh移动到test目录下，指定移动的目标路径(-t) 若覆盖文件先备份(-b) 强制复制（-f）



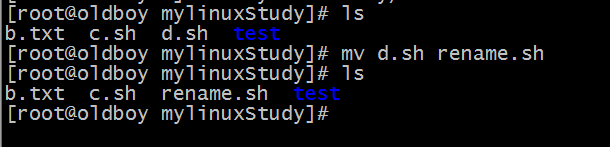
### 使用技巧

视mv命令中第二个参数类型的不同（是目标文件还是目标目录），mv命令将文件重命名或将其移至一个新的目录中。

当第二个参数类型是文件时，mv命令完成文件重命名，此时，源文件只能有一个（也可以是源目录名），它将所给的源文件或目录重命名为给定的目标文件名。

mv d.sh rename.sh

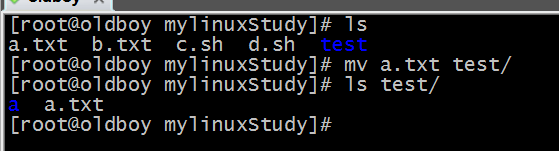
重命名



当第二个参数是已存在的目录名称时，源文件或目录参数可以有多个，mv命令将各参数指定的源文件均移至目标目录中。在跨文件系统移动文件时，mv先拷贝，再将原有文件删除，而链至该文件的链接也将丢失。

mv a.txt test/

将a.txt分配到test目录下



## rm 删除文件 文件夹

### 意义

用户可以用rm命令删除不需要的文件。该命令的功能为删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件，只是断开了链接，原文件保持不变。

### 命令格式

rm [参数] 目标文件或目录创建目录

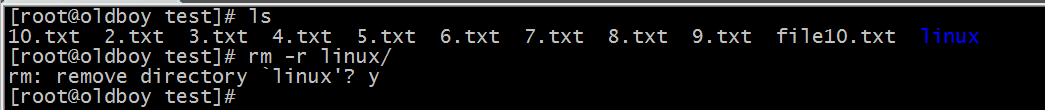
### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -d | directory 删除可能仍有数据的目录 (只限超级用户) |  |
| -f | 略过不存在的文件，不显示任何信息 |  |
| -i | --interactive 进行任何删除操作前必须先确认 |  |
| -r/R | --recursive 同时删除该目录下的所有目录层 |  |
| -v | --verbose 详细显示进行的步骤 |  |

### 例子

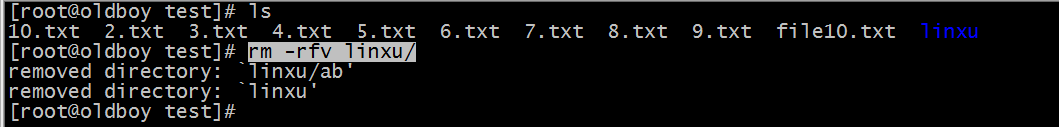
rm -r linux

删除linux目录，-r 递归删除，删除前会询问



rm -rfv linxu/

递归删除linxu目录 不询问(-f) 并且打印删除信息（-v）



### 使用技巧

要删除第一个字符为‘-’的文件 (例如‘-foo’)，请使用以下其中一种方法：

rm -- -foo

rm ./-foo

请注意，如果使用 rm 来删除文件，通常仍可以将该文件恢复原状。如果想保证

该文件的内容无法还原，请考虑使用 shred。

范例：

删除所有rb语言程序档；删除前逐一询问确认 :

rm -i \*.rb

将 Finished 子目录及子目录中所有档案删除 :

rm -r Finished

将 Finished 子目录及子目录中所有档案删除,并且不用--确认 :

rm -rf Finished

## find 查找文件

### 意义

find命令用于：在一个目录（及子目录）中搜索文件，你可以指定一些匹配条件，如按文件名、文件类型、用户甚至是时间戳查找文件。

### 命令格式

find [path...] [expression]

path：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

expression：expression可以分为——“-options [-print -exec -ok ...]”

-options，指定find命令的常用选项，

-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shell命令。相应命令的形式为'command' { } \;，注意{ }和\；之间的空格

find ./ -size 0 -exec rm {} \; 删除文件大小为零的文件 （还可以以这样做：rm -i `find ./ -size 0` 或 find ./ -size 0 | xargs rm -f &）

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：find . -type f -exec ls -l { } \;

在/logs目录中查找更改时间在5日以前的文件并删除它们：find /logs -type f -mtime +5 -exec rm { } \;

-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shell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。

find . -name "\*.conf" -mtime +5 -ok rm { } \; 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

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -name | 按文件名查找 | find ./ -name ‘1.txt’ |
| -perm | 按照文件权限查找 | find . -perm 755 –print 在当前目录下查找文件权限位为755的文件，即文件属主可以读、写、执行，其他用户可以读、执行的文件 |
| -prune | 使用这一选项可以使find命令不在当前指定的目录中查找，如果同时使用-depth选项，那么-prune将被find命令忽略。 | find /apps -path "/apps/bin" -prune -o –print 在/apps目录下查找文件，但不希望在/apps/bin目录下查找  find /usr/sam -path "/usr/sam/dir1" -prune -o –print 在/usr/sam目录下查找不在dir1子目录之内的所有文件 |
| -user | 根据文件的属主去查 | find ~ -user alivn –print 在$HOME目录中查找文件属主为alivn的文件 |
| -group | 按照文件所属的组来查找文件。 | find /apps -group gem –print 在/apps目录下查找属于gem用户组的文件 |
| -mtime -n +n | 按照文件的更改时间来查找文件， - n表示文件更改时间距现在n天以内，+ n表示文件更改时间距现在n天以前。 | find / -mtime -5 –print 在系统根目录下查找更改时间在5日以内的文件  find /var/adm -mtime +3 –print 在/var/adm目录下查找更改时间在3日以前的文件 |
| -nogroup | 查找无有效所属组的文件，即该文件所属的组在/etc/groups中不存在。 | find / –nogroup -print |
| -nouser | 查找无有效属主的文件，即该文件的属主在/etc/passwd中不存在。 | find /home -nouser –print |
| -newer | -newer file1 ! file2 | 查找更改时间比文件file1新但比文件file2旧的文件 |
| -type | 查找某一类型的文件 | 诸如：  b - 块设备文件。  d - 目录。  c - 字符设备文件。  p - 管道文件。  l - 符号链接文件。  f - 普通文件。 |
| -size n：[c] | 查找文件长度为n块的文件，带有c时表示文件长度以字节计。 | find . -size +1000000c –print 在当前目录下查找文件长度大于1 M字节的文件  find /home/apache -size 100c –print 在/home/apache目录下查找文件长度恰好为100字节的文件  find . -size +10 –print 在当前目录下查找长度超过10块的文件（一块等于512字节） |
| -depth | 在查找文件时，指定目录文件的深度 | find / -name "CON.FILE" -depth –print 它将首先匹配所有的文件然后再进入子目录中查找 |
| -mount | -mount：在查找文件时不跨越文件系统mount点 | find . -name "\*.XC" -mount –print 从当前目录开始查找位于本文件系统中文件名以XC结尾的文件（不进入其他文件系统） |
| -follow： | 如果find命令遇到符号链接文件，就跟踪至链接所指向的文件。 |  |

### 例子

find与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。但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xargs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。

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。

find . -type f -print | xargs file 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

find / -name "core" -print | xargs echo "" >/tmp/core.log 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 . -type f -print | xargs grep "hostname" 用grep命令在所有的普通文件中搜索hostname这个词

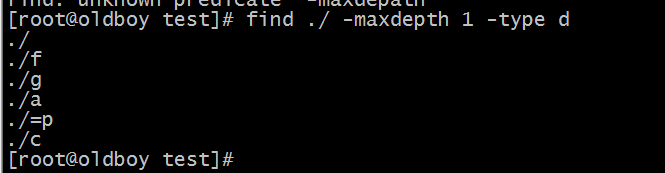
find ./ -mtime +3 -print|xargs rm -f –r 删除3天以前的所有东西 （find . -ctime +3 -exec rm -rf {} \;）

find ./ -size 0 | xargs rm -f & 删除文件大小为零的文件

find命令配合使用exec和xargs可以使用户对所匹配到的文件执行几乎所有的命令。

匹配当前目录下的一级目录，不包含 隐藏目录

find ./ -maxdepth 1 -type d



## sort 排序

### 意义

sort 命令对 File 参数指定的文件中的行排序，并将结果写到标准输出。如果 File 参数指定多个文件，那么 sort 命令将这些文件连接起来，并当作一个文件进行排序。

### 命令格式

sort [参数] [file or stdin]

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -f | 忽略大小写的差异，例如 A 与 a 视为编码相同； |  |
| -b | 忽略最前面的空格符部分； |  |
| -M | 以月份的名字来排序，例如 JAN, DEC 等等的排序方法 |  |
| -n | 使用『纯数字』进行排序(默认是以文字型态来排序的)； |  |
| -r | 反向排序； |  |
| -u | 就是 uniq ，相同的数据中，仅出现一行代表； |  |
| -t | 分隔符，默认是用 [tab] 键来分隔； |  |
| -k | 以那个区间 (field) 来进行排序的意思 |  |

### 例子

对/etc/passwd 的账号进行排序

[root@www ~]# cat /etc/passwd | sort

adm:x:3:4:adm:/var/adm:/sbin/nologin

apache:x:48:48:Apache:/var/www:/sbin/nologin

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

sort 是默认以第一个数据来排序，而且默认是以字符串形式来排序,所以由字母 a 开始升序排序。

/etc/passwd 内容是以 : 来分隔的，我想以第三栏来排序，该如何

[root@www ~]# cat /etc/passwd | sort -t ':' -k 3

root:x:0:0:root:/root:/bin/bash

uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

bin:x:1:1:bin:/bin:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

默认是以字符串来排序的，如果想要使用数字排序：

cat /etc/passwd | sort -t ':' -k 3n

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

默认是升序排序，如果要倒序排序，如下

cat /etc/passwd | sort -t ':' -k 3nr

nobody:x:65534:65534:nobody:/nonexistent:/bin/sh

ntp:x:106:113::/home/ntp:/bin/false

messagebus:x:105:109::/var/run/dbus:/bin/false

sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin

如果要对/etc/passwd,先以第六个域的第2个字符到第4个字符进行正向排序，再基于第一个域进行反向排序。

cat /etc/passwd | sort -t':' -k 6.2,6.4 -k 1r

sync:x:4:65534:sync:/bin:/bin/sync

proxy:x:13:13:proxy:/bin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

查看/etc/passwd有多少个shell:对/etc/passwd的第七个域进行排序，然后去重:

cat /etc/passwd | sort -t':' -k 7 -u

root:x:0:0:root:/root:/bin/bash

syslog:x:101:102::/home/syslog:/bin/false

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

sync:x:4:65534:sync:/bin:/bin/sync

sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin

## uniq 去除重复行

### 意义

uniq命令可以去除排序过的文件中的重复行，因此uniq经常和sort合用。也就是说，为了使uniq起作用，所有的重复行必须是相邻的。

### 命令格式

uniq [-icu]

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -i | 忽略大小写字符的不同； |  |
| -c | 进行计数 |  |
| -u | 只显示唯一行 |  |
| -d | 显示存在重复行的 行数据 |  |

### 例子

testfile的内容如下

[复制代码](javascript:void(0);)

cat testfile

hello

world

friend

hello

world

hello

[复制代码](javascript:void(0);)

直接删除未经排序的文件，将会发现没有任何行被删除

[复制代码](javascript:void(0);)

#uniq testfile

hello

world

friend

hello

world

hello

[复制代码](javascript:void(0);)

排序文件，默认是去重

#cat words | sort |uniq

friend

hello

world

排序之后删除了重复行，同时在行首位置输出该行重复的次数

#sort testfile | uniq -c

1 friend

3 hello

2 world

仅显示存在重复的行，并在行首显示该行重复的次数

#sort testfile | uniq -dc

3 hello

2 world

仅显示不重复的行

sort testfile | uniq -u

friend

## cut 提取文本列

### 意义

cut命令可以从一个文本文件或者文本流中提取文本列。

### 命令格式

cut -d'分隔字符' -f fields <==用于有特定分隔字符

cut -c 字符区间 <==用于排列整齐的信息

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -d | -d ：后面接分隔字符。与 -f 一起使用； |  |
| -f | 依据 -d 的分隔字符将一段信息分割成为数段，用 -f 取出第几段的意思； |  |
| -c | 以字符 (characters) 的单位取出固定字符区间； |  |

### 例子

PATH 变量如下

[root@www ~]# echo $PATH

/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/usr/X11R6/bin:/usr/games

# 1 | 2 | 3 | 4 | 5 | 6 | 7

将 PATH 变量取出，我要找出第五个路径。

#echo $PATH | cut -d ':' -f 5

/usr/local/bin

将 PATH 变量取出，我要找出第三和第五个路径。

#echo $PATH | cut -d ':' -f 3,5

/sbin:/usr/local/bin

将 PATH 变量取出，我要找出第三到最后一个路径。

echo $PATH | cut -d ':' -f 3-

/sbin:/usr/sbin:/usr/local/bin:/usr/X11R6/bin:/usr/games

将 PATH 变量取出，我要找出第一到第三个路径。

#echo $PATH | cut -d ':' -f 1-3

/bin:/usr/bin:/sbin:

将 PATH 变量取出，我要找出第一到第三，还有第五个路径。

echo $PATH | cut -d ':' -f 1-3,5

/bin:/usr/bin:/sbin:/usr/local/bin

实用例子:只显示/etc/passwd的用户和shell

#cat /etc/passwd | cut -d ':' -f 1,7

root:/bin/bash

daemon:/bin/sh

bin:/bin/sh

## wc 统计文本

### 意义

统计文件里面有多少单词，多少行，多少字

### 命令格式

wc [-lwm]

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -l | 列出有多少行 |  |
| -w | 列出有多少字（英文单词） |  |
| -m | 列出有多少字符 |  |

#### 例子

默认使用wc统计/etc/passwd

#wc /etc/passwd

40 45 1719 /etc/passwd

40是行数，45是单词数，1719是字节数

wc的命令比较简单使用，每个参数使用如下：

[复制代码](javascript:void(0);)

#wc -l /etc/passwd #统计行数，在对记录数时，很常用

40 /etc/passwd #表示系统有40个账户

#wc -w /etc/passwd #统计单词出现次数

45 /etc/passwd

#wc -m /etc/passwd #统计文件的字节数

1719

[复制代码](javascript:void(0);)

## grep 文本搜索

### 意义

**g**rep (global search regular expression\_r(RE) and print out the line,全面搜索正则表达式并把行打印出来)是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。

### 命令格式

grep [OPTIONS] PATTERN [FILE...]

grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

grep命令用于搜索由Pattern参数指定的模式，并将每个匹配的行写入标准输出中。这些模式是具有限定的正则表达式，它们使用ed或egrep命令样式。如果在File参数中指定了多个名称，grep命令将显示包含匹配行的文件的名称。对 shell 有特殊含义的字符 ($, \*, [, |, ^, (, ), \ ) 出现在 Pattern参数中时必须带双引号。如果 Pattern参数不是简单字符串，通常必须用单引号将整个模式括起来。在诸如 [a-z], 之类的表达式中，-（减号）cml 可根据当前正在整理的序列来指定一个范围。整理序列可以定义等价的类以供在字符范围中使用。如果未指定任何文件，grep会假定为标准输入。

### grep正则表达式元字符集(基本集)

^ 锚定行的开始 如：'^grep'匹配所有以grep开头的行。

$ 锚定行的结束 如：'grep$'匹配所有以grep结尾的行。

. 匹配一个非换行符的字符 如：'gr.p'匹配gr后接一个任意字符，然后是p。

\* 匹配零个或多个先前字符 如：'\*grep'匹配所有一个或多个空格后紧跟grep的行。 .\*一起用代表任意字符。

[] 匹配一个指定范围内的字符，如'[Gg]rep'匹配Grep和grep。

[^] 匹配一个不在指定范围内的字符，如：'[^A-FH-Z]rep'匹配不包含A-R和T-Z的一个字母开头，紧跟rep的行。

.. 标记匹配字符，如：'love'，love被标记为1。

\< 锚定单词的开始，如：'\<grep'匹配包含以grep开头的单词的行。

\> 锚定单词的结束，如'grep\>'匹配包含以grep结尾的单词的行。

x\{m\} 连续重复字符x，m次，如：'o\{5\}'匹配包含连续5个o的行。

x\{m,\} 连续重复字符x,至少m次，如：'o\{5,\}'匹配至少连续有5个o的行。

x\{m,n\} 连续重复字符x，至少m次，不多于n次，如：'o\{5,10\}'匹配连续5--10个o的行。

\w 匹配一个文字和数字字符，也就是[A-Za-z0-9]，如：'G\w\*p'匹配以G后跟零个或多个文字或数字字符，然后是p。

\W w的反置形式，匹配一个非单词字符，如点号句号等。\W\*则可匹配多个。

\b 单词锁定符，如: '\bgrep\b'只匹配grep，即只能是grep这个单词，两边均为空格。

参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -n | 同时匹配匹配行的上下n行 |  |
| -b，--byte-offset | 打印匹配行前面打印该行所在的块号码。 |  |
| -c,--count | 只打印匹配的行数，不显示匹配的内容。 |  |
| -f File，--file=File | 从文件中提取模板。空文件中包含0个模板，所以什么都不匹配。 |  |
| -h，--no-filename | 当搜索多个文件时，不显示匹配文件名前缀。 |  |
| -i，--ignore-case | 忽略大小写差别。 |  |
| -q，--quiet | 取消显示，只返回退出状态。0则表示找到了匹配的行 |  |
| -l，--files-with-matches | 打印匹配模板的文件清单。 |  |
| -L，--files-without-match | 打印不匹配模板的文件清单。 |  |
| -n，--line-number | 在匹配的行前面打印行号。 |  |
| -s，--silent | 不显示关于不存在或者无法读取文件的错误信息 |  |
| -v，--revert-match | 反检索，只显示不匹配的行 |  |
| -w，--word-regexp | 如果被\<和\>引用，就把表达式做为一个单词搜索。 |  |
| -V，--version | 显示软件版本信息 |  |

### 例子

ls -l | grep '^a' 通过管道过滤ls -l输出的内容，只显示以a开头的行。

grep 'test' d\* 显示所有以d开头的文件中包含test的行。

grep 'test' aa bb cc 显示在aa，bb，cc文件中匹配test的行。

grep '[a-z]' aa 显示所有包含每个字符串至少有5个连续小写字符的字符串的行。

grep 'w(es)t.\*' aa 如果west被匹配，则es就被存储到内存中，并标记为1，然后搜索任意个字符(.\*)，这些字符后面紧跟着另外一个es()，找到就显示该行。如果用egrep或grep -E，就不用""号进行转义，直接写成'w(es)t.\*'就可以了。

grep -i pattern files ：不区分大小写地搜索。默认情况区分大小写

grep -l pattern files ：只列出匹配的文件名，

grep -L pattern files ：列出不匹配的文件名，

grep -w pattern files ：只匹配整个单词，而不是字符串的一部分(如匹配‘magic’，而不是‘magical’)，

grep -C number pattern files ：匹配的上下文分别显示[number]行，

grep pattern1 | pattern2 files ：显示匹配 pattern1 或 pattern2 的行，

grep pattern1 files | grep pattern2 ：显示既匹配 pattern1 又匹配 pattern2 的行。

## awk 文本分析

### 意义

awk是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

awk有3个不同版本: awk、nawk和gawk，未作特别说明，一般指gawk，gawk 是 AWK 的 GNU 版本。

### 命令格式

awk '{pattern + action}' {filenames}

尽管操作可能会很复杂，但语法总是这样，其中 pattern 表示 AWK 在数据中查找的内容，而 action 是在找到匹配内容时所执行的一系列命令。花括号（{}）不需要在程序中始终出现，但它们用于根据特定的模式对一系列指令进行分组。 pattern就是要表示的正则表达式，用斜杠括起来。

awk语言的最基本功能是在文件或者字符串中基于指定规则浏览和抽取信息，awk抽取信息后，才能进行其他文本操作。完整的awk脚本通常用来格式化文本文件中的信息。

通常，awk是以文件的一行为处理单位的。awk每接收文件的一行，然后执行相应的命令，来处理文本。

### 调用awk方式

1.命令行方式

awk [-F field-separator] 'commands' input-file(s)

其中，commands 是真正awk命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件。

在awk中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格。

2.shell脚本方式

将所有的awk命令插入一个文件，并使awk程序可执行，然后awk命令解释器作为脚本的首行，一遍通过键入脚本名称来调用。

相当于shell脚本首行的：#!/bin/sh

可以换成：#!/bin/awk

3.将所有的awk命令插入一个单独文件，然后调用：

awk -f awk-script-file input-file(s)

其中，-f选项加载awk-script-file中的awk脚本，input-file(s)跟上面的是一样的。

### awk工作流程

读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域,$1表示第一个域,$n表示第n个域。默认域分隔符是"空白键" 或 "[tab]键",所以$1表示登录用户，$3表示登录用户ip,以此类推。

### ****awk内置变量****

ARGC 命令行参数个数

ARGV 命令行参数排列

ENVIRON 支持队列中系统环境变量的使用

FILENAME awk浏览的文件名

FNR 浏览文件的记录数

FS 设置输入域分隔符，等价于命令行 -F选项

NF 浏览记录的域的个数

NR 已读的记录数

OFS 输出域分隔符

ORS 输出记录分隔符

RS 控制记录分隔符

$0变量是指整条记录。$1表示当前行的第一个域,$2表示当前行的第二个域,......以此类推。

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
|  |  |  |

### awk编程

**变量和赋值**

除了awk的内置变量，awk还可以自定义变量。

下面统计/etc/passwd的账户人数

awk '{count++;print $0;} END{print "user count is ", count}' /etc/passwd

root:x:0:0:root:/root:/bin/bash

......

user count is  40

count是自定义变量。之前的action{}里都是只有一个print,其实print只是一个语句，而action{}可以有多个语句，以;号隔开。

这里没有初始化count，虽然默认是0，但是妥当的做法还是初始化为0:

awk 'BEGIN {count=0;print "[start]user count is ", count} {count=count+1;print $0;} END{print "[end]user count is ", count}' /etc/passwd

[start]user count is 0

root:x:0:0:root:/root:/bin/bash

...

[end]user count is 40

统计某个文件夹下的文件占用的字节数

ls -l |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ", size}'  
[end]size is  8657198

如果以M为单位显示:

ls -l |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ", size/1024/1024,"M"}'   
[end]size is  8.25889 M

注意，统计不包括文件夹的子目录。

awk中的条件语句是从C语言中借鉴来的，见如下声明方式：

[复制代码](javascript:void(0);)

if (expression) {

statement;

statement;

... ...

}

if (expression) {

statement;

} else {

statement2;

}

if (expression) {

statement1;

} else if (expression1) {

statement2;

} else {

statement3;

}

[复制代码](javascript:void(0);)

统计某个文件夹下的文件占用的字节数,过滤4096大小的文件(一般都是文件夹):

ls -l |awk 'BEGIN {size=0;print "[start]size is ", size} {if($5!=4096){size=size+$5;}} END{print "[end]size is ", size/1024/1024,"M"}'   
[end]size is  8.22339 M

#### 循环语句

awk中的循环语句同样借鉴于C语言，支持while、do/while、for、break、continue，这些关键字的语义和C语言中的语义完全相同。

#### 数组

  因为awk中数组的下标可以是数字和字母，数组的下标通常被称为关键字(key)。值和关键字都存储在内部的一张针对key/value应用hash的表格里。由于hash不是顺序存储，因此在显示数组内容时会发现，它们并不是按照你预料的顺序显示出来的。数组和变量一样，都是在使用时自动创建的，awk也同样会自动判断其存储的是数字还是字符串。一般而言，awk中的数组用来从记录中收集信息，可以用于计算总和、统计单词以及跟踪模板被匹配的次数等等。

显示/etc/passwd的账户

[复制代码](javascript:void(0);)

awk -F ':' 'BEGIN {count=0;} {name[count] = $1;count++;}; END{for (i = 0; i < NR; i++) print i, name[i]}' /etc/passwd

0 root

1 daemon

2 bin

3 sys

4 sync

5 games

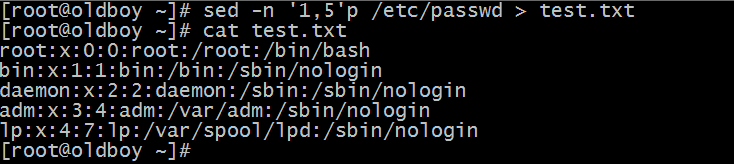
......

[复制代码](javascript:void(0);)

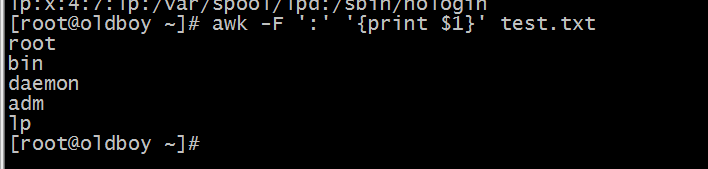
这里使用for循环遍历数组

### 例子

取出用户表的前5行 重定向test.txt文件

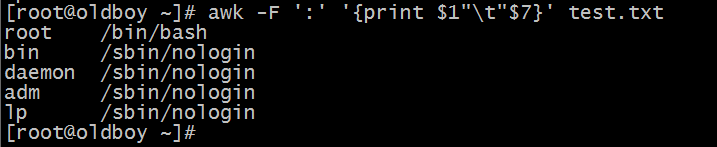


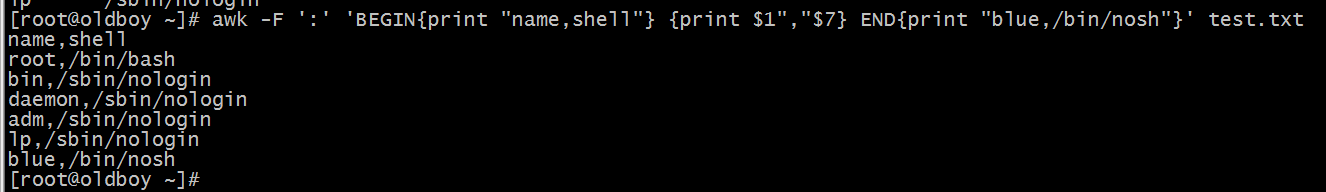
只显示用户名



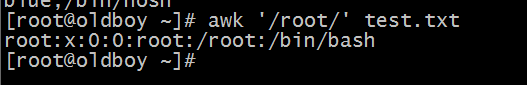
-F指定域分隔符为':'。

如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以tab键分割



如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以逗号分割,而且在所有行添加列名name,shell,在最后一行添加"blue,/bin/nosh"。

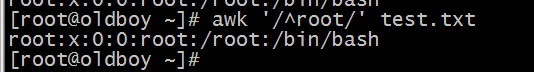
awk工作流程是这样的：先执行BEGING，然后读取文件，读入有/n换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域,$1表示第一个域,$n表示第n个域,随后开始执行模式所对应的动作action。接着开始读入第二条记录······直到所有的记录都读完，最后执行END操作。



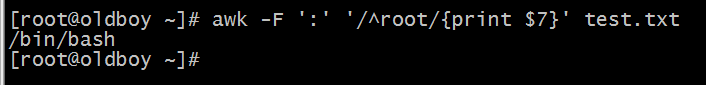
搜索/etc/passwd有root关键字的所有行

这种是pattern的使用示例，匹配了pattern(这里是root)的行才会执行action(没有指定action，默认输出每行的内容)。

搜索支持正则，例如找root开头的: awk -F: '/^root/' /etc/passwd

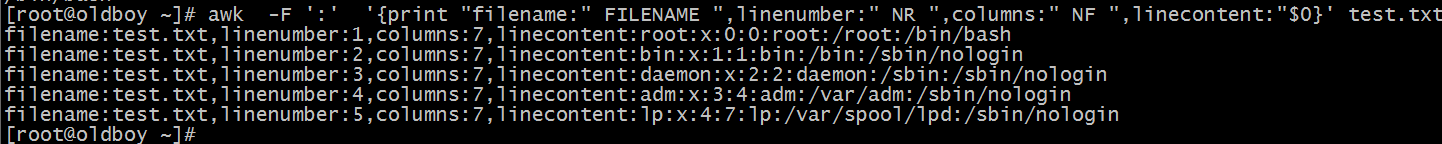


搜索/etc/passwd有root关键字的所有行，并显示对应的shell

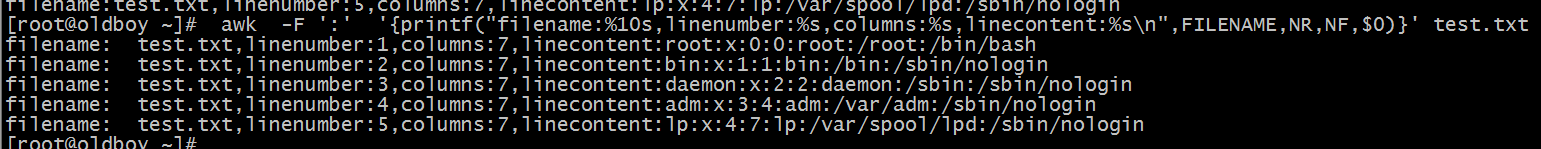


这里指定了action{print $7}

统计/etc/passwd:文件名，每行的行号，每行的列数，对应的完整行内容:



使用printf替代print,可以让代码更加简洁，易读



print和printf

awk中同时提供了print和printf两种打印输出的函数。

其中print函数的参数可以是变量、数值或者字符串。字符串必须用双引号引用，参数用逗号分隔。如果没有逗号，参数就串联在一起而无法区分。这里，逗号的作用与输出文件的分隔符的作用是一样的，只是后者是空格而已。

printf函数，其用法和c语言中printf基本相似,可以格式化字符串,输出复杂时，printf更加好用，代码更易懂。

## sed 流式文本编辑器

### 意义

### sed 是一种在线编辑器，它一次处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有 改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。

### 命令格式

cd [参数] [dirname]

切换到 dirname 目录

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| -n | 使用安静(silent)模式。在一般 sed 的用法中，所有来自 STDIN 的数据一般都会被列出到终端上。但如果加上 -n 参数后，则只有经过sed 特殊处理的那一行(或者动作)才会被列出来。 |  |
| -e | 直接在命令列模式上进行 sed 的动作编辑 |  |
| -f | 直接将 sed 的动作写在一个文件内， -f filename 则可以运行 filename 内的 sed 动作； |  |
| -r | sed 的动作支持的是延伸型正规表示法的语法。(默认是基础正规表示法语法) |  |
| -i | 直接修改读取的文件内容，而不是输出到终端。 | 比较常用，一般用于替换文件中的某个文本信息 |
|  |  |  |

### 动作说明

[n1[,n2]]function

n1, n2 ：不见得会存在，一般代表『选择进行动作的行数』，举例来说，如果我的动作是需要在 10 到 20 行之间进行的，则『 10,20[动作行为] 』

function：

a ：新增， a 的后面可以接字串，而这些字串会在新的一行出现(目前的下一行)～

c ：取代， c 的后面可以接字串，这些字串可以取代 n1,n2 之间的行！

d ：删除，因为是删除啊，所以 d 后面通常不接任何咚咚；

i ：插入， i 的后面可以接字串，而这些字串会在新的一行出现(目前的上一行)；

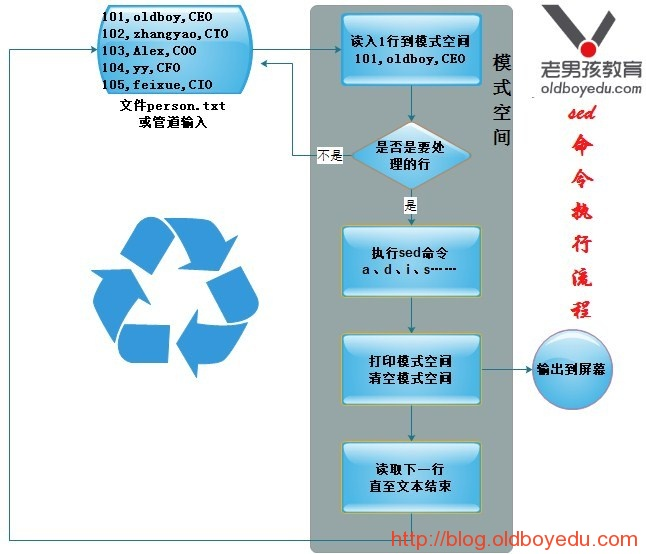
p ：列印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运行～

s ：取代，可以直接进行取代的工作哩！通常这个 s 的动作可以搭配正规表示法！例如 1,20s/old/new/g 就是啦！

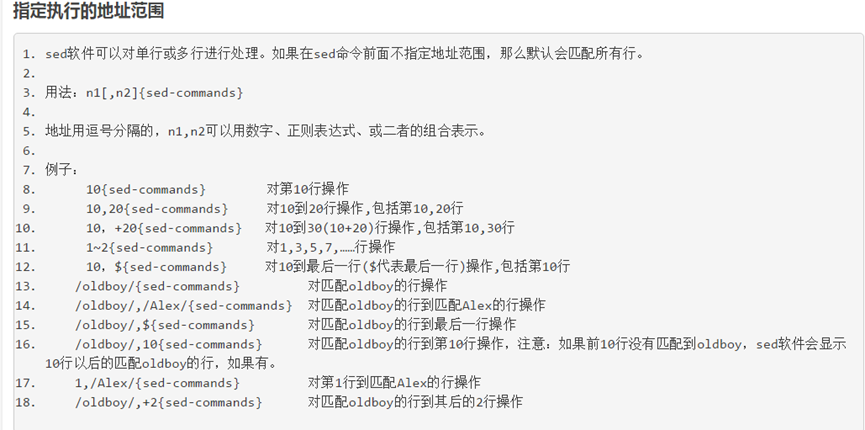
### 命令执行流程

概括流程：Sed软件从文件或管道中读取一行，处理一行，输出一行；再读取一行，再处理一行，再输出一行……

**模式空间:sed软件内部的一个临时缓存，用于存放读取到的内容。**



### 例子



#### 以行为单位的新增/删除

将 /etc/passwd 的内容列出并且列印行号，同时，请将第 2~5 行删除！

[root@www ~]# nl /etc/passwd | sed '2,5d'

1 root:x:0:0:root:/root:/bin/bash

6 sync:x:5:0:sync:/sbin:/bin/sync

7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

.....(后面省略).....

sed 的动作为 '2,5d' ，那个 d 就是删除！因为 2-5 行给他删除了，所以显示的数据就没有 2-5 行罗～ 另外，注意一下，原本应该是要下达 sed -e 才对，没有 -e 也行啦！同时也要注意的是， sed 后面接的动作，请务必以 '' 两个单引号括住喔！

只要删除第 2 行

nl /etc/passwd | sed '2d'

要删除第 3 到最后一行

nl /etc/passwd | sed '3,$d'

在第二行后(亦即是加在第三行)加上『drink tea?』字样！

[root@www ~]# nl /etc/passwd | sed '2a drink tea'

1 root:x:0:0:root:/root:/bin/bash

2 bin:x:1:1:bin:/bin:/sbin/nologin

drink tea

3 daemon:x:2:2:daemon:/sbin:/sbin/nologin

.....(后面省略).....

那如果是要在第二行前

nl /etc/passwd | sed '2i drink tea'

如果是要增加两行以上，在第二行后面加入两行字，例如『Drink tea or .....』与『drink beer?』

[复制代码](javascript:void(0);)

[root@www ~]# nl /etc/passwd | sed '2a Drink tea or ......\

> drink beer ?'

1 root:x:0:0:root:/root:/bin/bash

2 bin:x:1:1:bin:/bin:/sbin/nologin

Drink tea or ......

drink beer ?

3 daemon:x:2:2:daemon:/sbin:/sbin/nologin

.....(后面省略).....

[复制代码](javascript:void(0);)

每一行之间都必须要以反斜杠『 \ 』来进行新行的添加喔！所以，上面的例子中，我们可以发现在第一行的最后面就有 \ 存在。

#### 以行为单位的替换与显示

将第2-5行的内容取代成为『No 2-5 number』呢？

[root@www ~]# nl /etc/passwd | sed '2,5c No 2-5 number'

1 root:x:0:0:root:/root:/bin/bash

No 2-5 number

6 sync:x:5:0:sync:/sbin:/bin/sync

.....(后面省略).....

透过这个方法我们就能够将数据整行取代了！

仅列出 /etc/passwd 文件内的第 5-7 行

[root@www ~]# nl /etc/passwd | sed -n '5,7p'

5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

6 sync:x:5:0:sync:/sbin:/bin/sync

7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

可以透过这个 sed 的以行为单位的显示功能， 就能够将某一个文件内的某些行号选择出来显示。

#### 数据的搜寻并显示

搜索 /etc/passwd有root关键字的行

[复制代码](javascript:void(0);)

nl /etc/passwd | sed '/root/p'

1 root:x:0:0:root:/root:/bin/bash

1 root:x:0:0:root:/root:/bin/bash

2 daemon:x:1:1:daemon:/usr/sbin:/bin/sh

3 bin:x:2:2:bin:/bin:/bin/sh

4 sys:x:3:3:sys:/dev:/bin/sh

5 sync:x:4:65534:sync:/bin:/bin/sync

....下面忽略

[复制代码](javascript:void(0);)

如果root找到，除了输出所有行，还会输出匹配行。

使用-n的时候将只打印包含模板的行。

nl /etc/passwd | sed -n '/root/p'

1 root:x:0:0:root:/root:/bin/bash

#### 数据的搜寻并删除

删除/etc/passwd所有包含root的行，其他行输出

nl /etc/passwd | sed '/root/d'

2 daemon:x:1:1:daemon:/usr/sbin:/bin/sh

3 bin:x:2:2:bin:/bin:/bin/sh

....下面忽略

#第一行的匹配root已经删除了

#### 数据的搜寻并执行命令

找到匹配模式eastern的行后，

搜索/etc/passwd,找到root对应的行，执行后面花括号中的一组命令，每个命令之间用分号分隔，这里把bash替换为blueshell，再输出这行：

nl /etc/passwd | sed -n '/root/{s/bash/blueshell/;p}'  
 1  root:x:0:0:root:/root:/bin/blueshell

如果只替换/etc/passwd的第一个bash关键字为blueshell，就退出

nl /etc/passwd | sed -n '/bash/{s/bash/blueshell/;p;q}'

1 root:x:0:0:root:/root:/bin/blueshell

最后的q是退出。

#### 数据的搜寻并替换

s：单独使用→将每一行中**第一处**匹配的字符串进行替换 ==>sed命令   
g：每一行进行全部替换 ==>sed命令s的替换标志之一，非sed命令

表示替换全局的，不加g表示只替换匹配到的第一个  
-i：修改文件内容 ==>sed软件的选项

除了整行的处理模式之外， sed 还可以用行为单位进行部分数据的搜寻并取代。基本上 sed 的搜寻与替代的与 vi 相当的类似！他有点像这样：

sed 's/要被取代的字串/新的字串/g'

先观察原始信息，利用 /sbin/ifconfig 查询 IP

[root@www ~]# /sbin/ifconfig eth0

eth0 Link encap:Ethernet HWaddr 00:90:CC:A6:34:84

inet addr:192.168.1.100 Bcast:192.168.1.255 Mask:255.255.255.0

inet6 addr: fe80::290:ccff:fea6:3484/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

.....(以下省略).....

本机的ip是192.168.1.100。

将 IP 前面的部分予以删除

[root@www ~]# /sbin/ifconfig eth0 | grep 'inet addr' | sed 's/^.\*addr://g'

192.168.1.100 Bcast:192.168.1.255 Mask:255.255.255.0

接下来则是删除后续的部分，亦即： 192.168.1.100 Bcast:192.168.1.255 Mask:255.255.255.0

将 IP 后面的部分予以删除

[root@www ~]# /sbin/ifconfig eth0 | grep 'inet addr' | sed 's/^.\*addr://g' | sed 's/Bcast.\*$//g'

192.168.1.100

#### 多点编辑

一条sed命令，删除/etc/passwd第三行到末尾的数据，并把bash替换为blueshell

nl /etc/passwd | sed -e '3,$d' -e 's/bash/blueshell/'

1 root:x:0:0:root:/root:/bin/blueshell

2 daemon:x:1:1:daemon:/usr/sbin:/bin/sh

-e表示多点编辑，第一个编辑命令删除/etc/passwd第三行到末尾的数据，第二条命令搜索bash替换为blueshell。

#### 直接修改文件内容(危险动作)

sed 可以直接修改文件的内容，不必使用管道命令或数据流重导向！ 不过，由於这个动作会直接修改到原始的文件，所以请你千万不要随便拿系统配置来测试！ 我们还是使用下载的 regular\_express.txt 文件来测试看看吧！

利用 sed 将 regular\_express.txt 内每一行结尾若为 . 则换成 !

[root@www ~]# sed -i 's/\.$/\!/g' regular\_express.txt

利用 sed 直接在 regular\_express.txt 最后一行加入『# This is a test』

[root@www ~]# sed -i '$a # This is a test' regular\_express.txt

由於 $ 代表的是最后一行，而 a 的动作是新增，因此该文件最后新增『# This is a test』！

sed 的『 -i 』选项可以直接修改文件内容，这功能非常有帮助！举例来说，如果你有一个 100 万行的文件，你要在第 100 行加某些文字，此时使用 vim 可能会疯掉！因为文件太大了！那怎办？就利用 sed 啊！透过 sed 直接修改/取代的功能，你甚至不需要使用 vim 去修订！

#### 分组替换（正则）

将密码文件的 用户名（第一行）和对应的shell（最后一行）替换

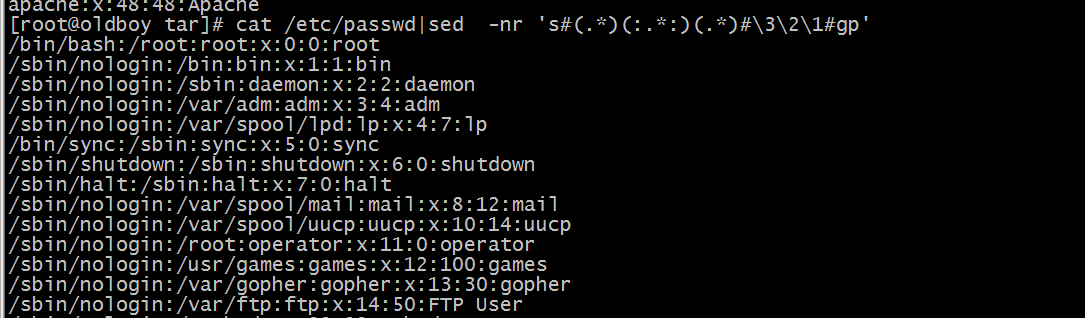
**cat /etc/passwd|sed -nr 's#(.\*)(:.\*:)(.\*)#\3\2\1#gp'**

说明：\3\2\1 分别代表了每个括号里面的正则 所匹配到的内容

\1对应 （.\*）

\2 对应 （：.\*:）

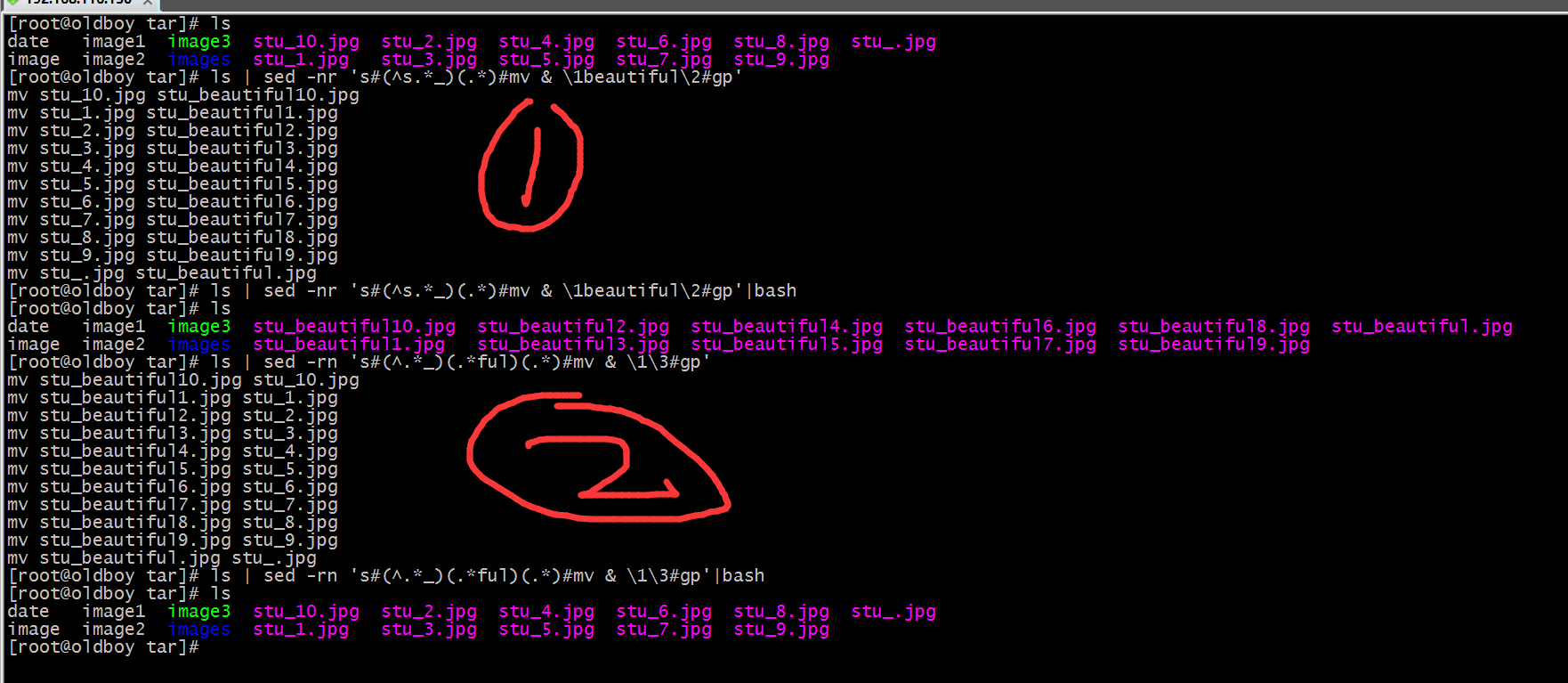
\3对应 (.\*)



#### 特殊符号&代表被替换的内容

&代表被匹配到的内容，我们可以用于获取指定匹配的内容。

如：我们拼接多条 文件改名 命令



## chkconfig 设置系统服务

### linux运行级别

       什么是运行级别呢？简单点来说，运行级别就是[**操作系统**](http://lib.csdn.net/base/operatingsystem)当前正在运行的功能级别。级别是从0到6，具有不同的功能。这些级别定义在/ect/inittab文件中。这个文件是init程序寻找的主要文件，最先运行的服务是那些放在/ect/rc.d目录下的文件。

**Linux下的7个运行级别：**

0系统停机状态，系统默认运行级别不能设置为0，否则不能正常启动，机器关闭。

1单用户工作状态，root权限，用于系统维护，禁止远程登陆，就像Windows下的安全模式登录。

2多用户状态，没有NFS支持。

3完整的多用户模式，有NFS，登陆后进入控制台命令行模式。

4系统未使用，保留一般不用，在一些特殊情况下可以用它来做一些事情。例如在笔记本电脑的电池用尽时，可以切换到这个模式来做一些设置。

5X11控制台，登陆后进入图形GUI模式，X Window系统。

6系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动。运行init 6机器就会重启。

**运行级别原理：**

1.在目录/etc/rc.d/init.d下有许多服务器脚本程序，一般称为服务(service)  
2.在/etc/rc.d下有7个名为rcN.d的目录，对应系统的7个运行级别

3.rcN.d目录下都是一些符号链接文件，这些链接文件都指向init.d目录下的service脚本文件，命名规则为K+nn+服务名或S+nn+服务名，其中nn为两位数字。

4.系统会根据指定的运行级别进入对应的rcN.d目录，并按照文件名顺序检索目录下的链接文件：对于以K开头的文件，系统将终止对应的服； 对于以S开头的文件，系统将启动对应的服务

5.查看运行级别用：runlevel

6.进入其它运行级别用：init N，如果init 3则进入终端模式，init 5则又登录图形GUI模式

7.另外init0为关机，init 6为重启系统

标准的Linux运行级别为3或5，如果是3的话，系统就在多用户状态；如果是5的话，则是运行着X Window系统。

不同的运行级别有不同的用处，也应该根据自己的不同情形来设置。例如，如果丢失了root口令，那么可以让机器启动进入单用户状态来设置。在启动后的lilo提示符下输入：

init=/bin/sh rw

这样就可以使机器进入运行级别1，并把root文件系统挂为读写。它会路过所有系统认证，让你使用passwd程序来改变root口令，然后启动到一个新的运行级。

### 意义

chkconfig命令可以用来检查、设置系统的各种服务

### 命令格式

chkconfig [--add][--del][--list][系统服务] 或 chkconfig [--level <等级代号>][系统服务][on/off/reset]

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
| --add | 增加所指定的系统服务，让chkconfig指令得以管理它，并同时在系统启动的叙述文件内增加相关数据。 |  |
| --del | 删除所指定的系统服务，不再由chkconfig指令管理，并同时在系统启动的叙述文件内删除相关数据。 |  |
| --level | 等级代号>à指定读系统服务要在哪一个执行等级中开启或关毕。 |  |
| --list | 显示系统服务 |  |

### 例子

chkconfig --list      列出所有的系统服务

chkconfig --add httpd 增加httpd服务

chkconfig --del httpd 删除httpd服务

chkconfig --level httpd 2345 on 把httpd在运行级别为2、3、4、5的情况下都是on（开启）的状态。

chkconfig命令提供了一种简单的方式来设置一个服务的运行级别。例如，为了设置MySQL服务器在运行级别3和4上运行，你必须首先将MySQL添加为受chkconfig管理的服务：

chkconfig --add mysql

现在，我们在级别3和5上设定服务为“on”

chkconfig --level 35 mysql on

在其他级别上设为off

chkconfig --level 01246 mysql off

为了确认你的配置被正确的修改了，我们可以列出服务将会运行的运行级别，如下所示：

#chkconfig --list mysql

mysql           0:off       1:off       2:off       3:on 4:off       5:on 6:off

## tar 解压缩文件

### 名词区分：

打包：将一大堆文件或目录变成一个总的文件【tar命令】

压缩：将一个大的文件通过一些压缩算法变成一个小文件【gzip，bzip2等】

Linux中很多压缩程序只能针对一个文件进行压缩，这样当你想要压缩一大堆文件时，你得将这一大堆文件先打成一个包（tar命令），然后再用压缩程序进行压缩（gzip bzip2命令）。

### 意义

打包 解压文件

### 命令格式

tar [主选项+辅选项] 文件或目录

使用该命令时，主选项必须有，它告诉tar要做什么事情，辅选项是辅助使用的，可以选用。

### 主选项

【一条命令以下5个参数只能有一个】

-c: --create 新建一个压缩文档，即打包

-x: --extract,--get解压文件

-t: --list,查看压缩文档里的所有内容

-r:--append 向压缩文档里追加文件

-u:--update 更新原压缩包中的文件

辅助选项：

-z:是否同时具有gzip的属性？即是否需要用gzip压缩或解压？一般格式为xxx.tar.gz或xx.tgz

-j：是否同时具有bzip2的属性？即是否需要用bzip2压缩或解压？一般格式为xx.tar.bz2

-v:显示操作过程！这个参数很常用

-f：使用文档名，注意，在f之后要立即接文档名，不要再加其他参数！

-C:切换到指定目录

--exclude FILE:在压缩过程中，不要将FILE打包

### 例子

#### 打包

习惯上以.tar后缀代表tar包，用xxx.tar.gz或.tgz代表gzip压缩过的tar文件，用.tar.bz2代表bzip2压缩过的tar文件。

将img1和img2两个文件夹打包成img.tar，仅打包不压缩

[复制代码](javascript:void(0);)

liuxiaoyan@development:~/test$ tar -cvf img.tar img1 img2

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ ls

img1 img2 img.tar

[复制代码](javascript:void(0);)

将img1和img2两个文件夹打包成img.tar.gz，打包后，以gzip压缩

[复制代码](javascript:void(0);)

liuxiaoyan@development:~/test$ tar -zcvf img.tar.gz img1 img2

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ ls

img1 img2 img.tar img.tar.gz

[复制代码](javascript:void(0);)

将img1和img2两个文件夹打包成img.tar.bz2，打包后，以bzip2来压缩

[复制代码](javascript:void(0);)

liuxiaoyan@development:~/test$ tar -jcvf img.tar.bz2 img1 img2

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ ls

img1 img2 img.tar img.tar.bz2 img.tar.gz

[复制代码](javascript:void(0);)

#### 不解压的情况下查看

查看img.tar中的所有内容

[复制代码](javascript:void(0);)

liuxiaoyan@development:~/test$ tar -tf img.tar

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

[复制代码](javascript:void(0);)

查看img.tar.gz的所有内容

加-v参数查看详细信息，-z可有可无

[复制代码](javascript:void(0);)

liuxiaoyan@development:~/test$ tar -tf img.tar.gz

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ tar -ztf img.tar.gz

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ tar -ztvf img.tar.gz

drwxrwxr-x liuxiaoyan/liuxiaoyan 0 2015-05-20 22:09 img1/

-rw-r--r-- liuxiaoyan/liuxiaoyan 1952 2015-05-20 22:08 img1/102.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 2344 2015-05-20 22:08 img1/101.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 1995 2015-05-20 22:09 img1/100.png

drwxrwxr-x liuxiaoyan/liuxiaoyan 0 2015-05-20 22:10 img2/

-rw-r--r-- liuxiaoyan/liuxiaoyan 1922 2015-05-20 22:09 img2/105.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 1546 2015-05-20 22:08 img2/104.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 2033 2015-05-20 22:08 img2/103.png

[复制代码](javascript:void(0);)

查看img.tar.bz2内容

加-v参数查看详细信息，-j可有可无

[复制代码](javascript:void(0);)

liuxiaoyan@development:~/test$ tar -tf img.tar.bz2

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ tar -jtf img.tar.bz2

img1/

img1/102.png

img1/101.png

img1/100.png

img2/

img2/105.png

img2/104.png

img2/103.png

liuxiaoyan@development:~/test$ tar -jtvf img.tar.bz2

drwxrwxr-x liuxiaoyan/liuxiaoyan 0 2015-05-20 22:09 img1/

-rw-r--r-- liuxiaoyan/liuxiaoyan 1952 2015-05-20 22:08 img1/102.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 2344 2015-05-20 22:08 img1/101.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 1995 2015-05-20 22:09 img1/100.png

drwxrwxr-x liuxiaoyan/liuxiaoyan 0 2015-05-20 22:10 img2/

-rw-r--r-- liuxiaoyan/liuxiaoyan 1922 2015-05-20 22:09 img2/105.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 1546 2015-05-20 22:08 img2/104.png

-rw-r--r-- liuxiaoyan/liuxiaoyan 2033 2015-05-20 22:08 img2/103.png

[复制代码](javascript:void(0);)

#### 解压

将img.tar解压到temp目录

liuxiaoyan@development:~/test$ cd temp/

liuxiaoyan@development:~/test/temp$ ls

liuxiaoyan@development:~/test/temp$ tar -xf ../img.tar

liuxiaoyan@development:~/test/temp$ ls

img1 img2

只减压img.tar中的img1

liuxiaoyan@development:~/test/temp$ tar -xf ../img.tar img1

liuxiaoyan@development:~/test/temp$ ls

img1

#### 更新

现在给movies文件下新建一个文件movie2.txt，然后更新enjoy.tar

$ tar -cf img.tar img1 img2

给img1增加了一张照片，现在用下面命令更新img.tar

$ tar -uf img.tar img1

#### 追加

$ tar -cf img.tar img1 img2  
img.tar里追加一个文件夹img3

$ tar -rf img.tar img3

#### -C参数

-C(大C参数很有用，改变文件目录，多层时用)  -C, --directory=DIR        change to directory DIR

使用了-C参数改变目录为-C后面 目录，不管是在打包还是解压都很有用。

用法：tar temp.tar **file** -C **fileDir**

$ cd temp

$ ls

img1 img2 img3

$ tar -cf img.tar temp/\*

这样打包会将temp目录也打包进去，使用下面带有-C参数的命令可以去掉路径信息  
$ tar -cf img.tar \* -C temp

 解压img.tar到当前目录下的temp目录

$ tar -xf img.tar **-C temp**

#### 解压方法总结

1、\*.tar 用 tar –xvf 解压

2、\*.gz 用 gzip -d或者gunzip 解压

3、\*.tar.gz和\*.tgz 用 tar –xzf 解压

4、\*.bz2 用 bzip2 -d或者用bunzip2 解压

5、\*.tar.bz2用tar –xjf 解压

6、\*.Z 用 uncompress 解压

7、\*.tar.Z 用tar –xZf 解压

8、\*.rar 用 unrar x 解压

9、\*.zip 用 unzip 解压

**使用unrar注意**：

先安装unrar,使用unrar不带参数运行一下就可以看到帮助信息。

如果一个文件s.rar，里面有1000首歌，用e参数，就直接把1000首歌全解压出来到当前目录。

如果是x参数，解压后，所有内容都放在同名的文件夹里面。

# 文件属性相关

## file 辨识文件类型

Linux file命令用于辨识文件类型。

通过file指令，我们得以辨识该文件的类型。

### 命令格式

file [-beLvz][-f <名称文件>][-m <魔法数字文件>...][文件或目录...]

### 参数

* -b 　列出辨识结果时，不显示文件名称。
* -c 　详细显示指令执行过程，便于排错或分析程序执行的情形。
* -f<名称文件> 　指定名称文件，其内容有一个或多个文件名称，让file依序辨识这些文件，格式为每列一个文件名称。
* -L 　直接显示符号连接所指向的文件的类别。
* -m<魔法数字文件> 　指定魔法数字文件。
* -v 　显示版本信息。
* -z 　尝试去解读压缩文件的内容。
* [文件或目录...] 要确定类型的文件列表，多个文件之间使用空格分开，可以使用shell通配符匹配多个文件。

**实例**

显示文件类型：

[root@localhost ~]# file install.log

install.log: UTF-8 Unicode text

[root@localhost ~]# file -b install.log <== 不显示文件名称

UTF-8 Unicode text

[root@localhost ~]# file -i install.log <== 显示MIME类别。

install.log: text/plain; charset=utf-8

[root@localhost ~]# file -b -i install.log

text/plain; charset=utf-8

显示符号链接的文件类型

[root@localhost ~]# ls -l /var/mail

lrwxrwxrwx 1 root root 10 08-13 00:11 /var/mail -> spool/mail

[root@localhost ~]# file /var/mail

/var/mail: symbolic link to `spool/mail'

[root@localhost ~]# file -L /var/mail

/var/mail: directory

[root@localhost ~]# file /var/spool/mail

/var/spool/mail: directory

[root@localhost ~]# file -L /var/spool/mail

/var/spool/mail: directory

## last 罗列登陆系统的用户信息

### 意义

该命令用来列出目前与过去登录系统的用户相关信息。

指令英文原义：show listing of last logged in users

    执行权限 ：有些需要特殊权限

    指令所在路径：/usr/bin/last

    执行last指令时，它会读取位于/var/log目录下名称为wtmp的文件，并把该给文件的内容记录的登录系统的用户名单全部显示出来。默认是显示wtmp的记录，btmp能显示的更详细，可以显示远程登录，例如ssh登录。

    utmp文件中保存的是当前正在本系统中的用户的信息。

    wtmp文件中保存的是登录过本系统的用户的信息。

**命令输出字段介绍：**

   第一列：用户名

   第二列：终端位置。pts/0 (伪终端) 意味着从诸如SSH或telnet的远程连接的用户.tty (teletypewriter) 意味着直接连接到计算机或者本地连接的用户

   第三列：登录ip或者内核 。如果你看见:0.0 或者什么都没有，这意味着用户通过本地终端连接。除了重启活动，内核版本会显示在状态中。

   第四列：开始时间

   第五列：结束时间（still login in 还未退出 down 直到正常关机 crash 直到强制关机）

   第六列：持续时间

### 命令语法

last [-R] [-num] [ -n num ] [-adiowx] [ -f file ] [ -t YYYYMMDDHHMMSS ] [name...]  [tty...]

### 命令参数

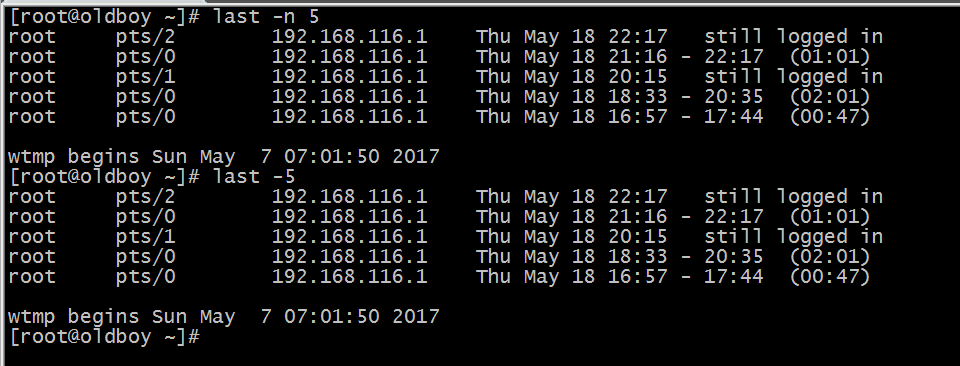
|  |  |  |
| --- | --- | --- |
| 参数 | 长参数 | 描叙 |
| -a |  | 将登录系统的的主机名称或IP地址，显示在最后一行 |
| -d |  | 将IP地址转换成主机名称 |
| -f |  | 指定记录文件，默认是显示/var/log目录下的wtmp文件的记录，但/var/log目录下得btmp能显示的内容更丰富，可以显示远程登录，例如ssh登录 ，包括失败的登录请求。 |
| -i |  | -i显示特定ip登录的情况。跟踪用 -i显示特定ip登录的情况。跟踪用 |
| -o |  | Read an old-type wtmp file (written by linux-libc5 applications). |
| -n |  | -n <显示列数>或-<显示列数> 　设置列出名单的显示列数 |
| -w |  | Display full user and domain names in the output |
| -R |  | 不显示登入系统的主机名称或IP（省略 hostname 的栏位） |
| -t |  | 显示YYYYMMDDHHMMSS之前的信息 |
| -x |  | 显示系统关闭、用户登录和退出的历史 |

**使用示例：**

**1：显示最后登录系统的N条记录**

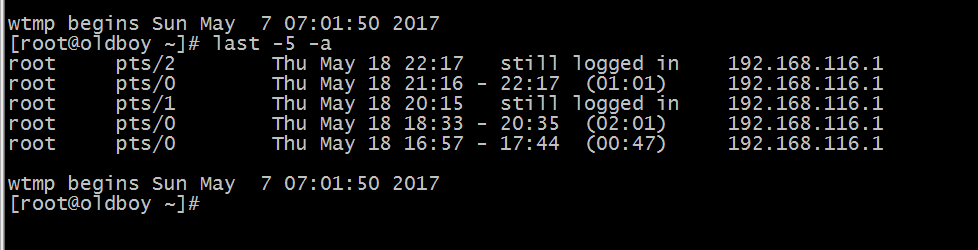
**last -n 5;**

**last -5**



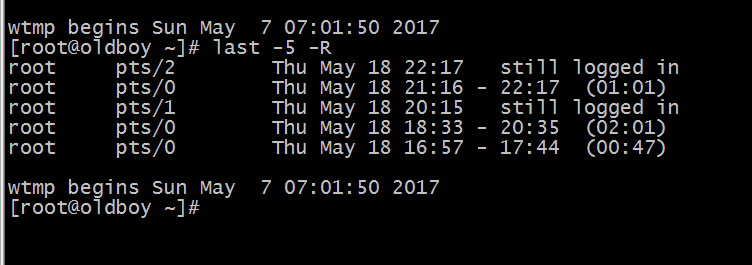
**3：将登录系统的主机名或IP地址显示在最后一行**

**last -5 -a**



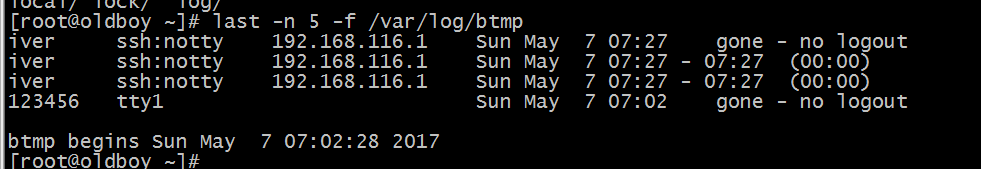
**4：不显示登入系统的主机名称或IP地址**

Last -5 -R



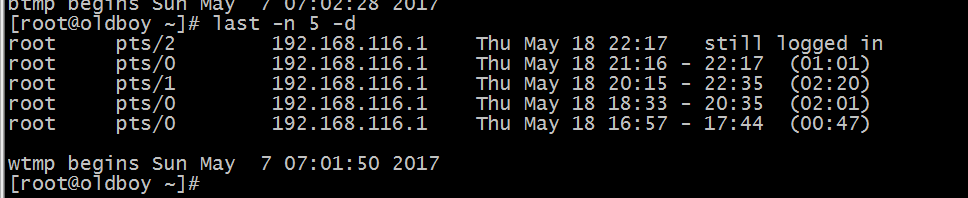
**5: 指定/var/log/btmp文件，查看登录系统的用户相关信息**

last -n 5 -f /var/log/btmp

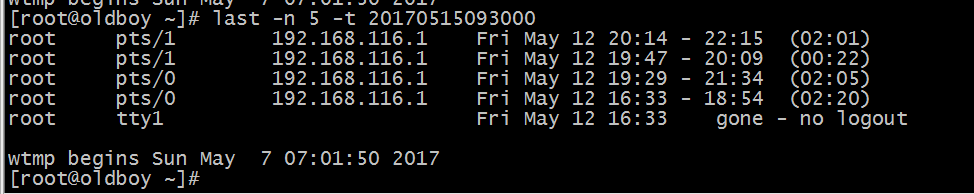


**6: 将IP地址转换成主机名称**

last -10 -d



**7: 显示YYYYMMDDHHMMSS（20170515093000）之前的信息**



## ln创建链接

ln命令是一个非常重要命令，它的功能是为某一个文件在另外一个位置建立一个同步的链接。

当我们需要在不同的目录，用到相同的文件时，我们不需要在每一个需要的目录下都放一个必须相同的文件，我们只要在某个固定的目录，放上该文件，然后在 其它的目录下用ln命令链接（link）它就可以，不必重复的占用磁盘空间。

**命令格式**

ln [参数][源文件或目录][目标文件或目录]

其中参数的格式为

[-bdfinsvF] [-S backup-suffix] [-V {numbered,existing,simple}]

[--help] [--version] [--]

**命令功能**    
Linux文件系统中，有所谓的链接(link)，我们可以将其视为档案的别名，而链接又可分为两种 : 硬链接(hard link)与软链接(symbolic link)，硬链接的意思是一个档案可以有多个名称，而软链接的方式则是产生一个特殊的档案，该档案的内容是指向另一个档案的位置。硬链接是存在同一个文件系统中，而软链接却可以跨越不同的文件系统。

不论是硬链接或软链接都不会将原本的档案复制一份，只会占用非常少量的磁碟空间。

**软链接**：

* 1.软链接，以路径的形式存在。类似于Windows操作系统中的快捷方式
* 2.软链接可以 跨文件系统 ，硬链接不可以
* 3.软链接可以对一个不存在的文件名进行链接
* 4.软链接可以对目录进行链接

**硬链接**：

* 1.硬链接，以文件副本的形式存在。但不占用实际空间。
* 2.不允许给目录创建硬链接
* 3.硬链接只有在同一个文件系统中才能创建

### 命令参数

#### 必要参数：

* -b 删除，覆盖以前建立的链接
* -d 允许超级用户制作目录的硬链接
* -f 强制执行
* -i 交互模式，文件存在则提示用户是否覆盖
* -n 把符号链接视为一般目录
* -s 软链接(符号链接)
* -v 显示详细的处理过程

#### 选择参数

* -S "-S<字尾备份字符串> "或 "--suffix=<字尾备份字符串>"
* -V "-V<备份方式>"或"--version-control=<备份方式>"
* --help 显示帮助信息
* --version 显示版本信息

### 实例

给文件创建软链接，为log2013.log文件创建软链接link2013，如果log2013.log丢失，link2013将失效：

ln -s log2013.log link2013

输出：

[root@localhost test]# ll

-rw-r--r-- 1 root bin 61 11-13 06:03 log2013.log

[root@localhost test]# ln -s log2013.log link2013

[root@localhost test]# ll

lrwxrwxrwx 1 root root 11 12-07 16:01 link2013 -> log2013.log

-rw-r--r-- 1 root bin 61 11-13 06:03 log2013.log

给文件创建硬链接，为log2013.log创建硬链接ln2013，log2013.log与ln2013的各项属性相同

ln log2013.log ln2013

输出：

[root@localhost test]# ll

lrwxrwxrwx 1 root root 11 12-07 16:01 link2013 -> log2013.log

-rw-r--r-- 1 root bin 61 11-13 06:03 log2013.log

[root@localhost test]# ln log2013.log ln2013

[root@localhost test]# ll

lrwxrwxrwx 1 root root 11 12-07 16:01 link2013 -> log2013.log

-rw-r--r-- 2 root bin 61 11-13 06:03 ln2013

-rw-r--r-- 2 root bin 61 11-13 06:03 log2013.log

## date查看修改时间

### 命令格式

date [选项]... [+格式]  
　或：date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]  
以给定的格式显示当前时间，或是设置系统日期。  
  
  -d,--date=字符串              显示指定字符串所描述的时间，而非当前时间  
  -f,--file=日期文件            类似--date，从日期文件中按行读入时间描述  
  -r, --reference=文件          显示文件指定文件的最后修改时间  
  -R, --rfc-2822                以RFC 2822格式输出日期和时间  
                                例如：2006年8月7日，星期一 12:34:56 -0600  
  --rfc-3339=TIMESPEC       以RFC 3339 格式输出日期和时间。  
                                TIMESPEC=`date'，`seconds'，或 `ns'   
                                表示日期和时间的显示精度。  
                                日期和时间单元由单个的空格分开：  
                                2006-08-07 12:34:56-06:00  
  -s, --set=字符串              设置指定字符串来分开时间  
  -u, --utc, --universal        输出或者设置协调的通用时间  
      --help            显示此帮助信息并退出  
      --version         显示版本信息并退出  
时间格式  
给定的格式FORMAT 控制着输出，解释序列如下：  
  
  %%    一个文字的 %  
  %a    当前locale 的星期名缩写(例如： 日，代表星期日)  
  %A    当前locale 的星期名全称 (如：星期日)  
  %b    当前locale 的月名缩写 (如：一，代表一月)  
  %B    当前locale 的月名全称 (如：一月)  
  %c    当前locale 的日期和时间 (如：2005年3月3日 星期四 23:05:25)  
  %C    世纪；比如 %Y，通常为省略当前年份的后两位数字(例如：20)  
  %d    按月计的日期(例如：01)  
  %D    按月计的日期；等于%m/%d/%y  
  %e    按月计的日期，添加空格，等于%\_d  
  %F    完整日期格式，等价于 %Y-%m-%d  
  %g    ISO-8601 格式年份的最后两位 (参见%G)  
  %G    ISO-8601 格式年份 (参见%V)，一般只和 %V 结合使用  
  %h    等于%b  
  %H    小时(00-23)  
  %I    小时(00-12)  
  %c    按年计的日期(001-366)  
  %k    时(0-23)  
  %l    时(1-12)  
  %m    月份(01-12)  
  %M    分(00-59)  
  %n    换行  
  %N    纳秒(000000000-999999999)  
  %p    当前locale 下的"上午"或者"下午"，未知时输出为空  
  %P    与%p 类似，但是输出小写字母  
  %r    当前locale 下的 12 小时时钟时间 (如：11:11:04 下午)  
  %R    24 小时时间的时和分，等价于 %H:%M  
  %s    自UTC 时间 1970-01-01 00:00:00 以来所经过的秒数  
  %S    秒(00-60)  
  %t    输出制表符 Tab  
  %T    时间，等于%H:%M:%S  
  %u    星期，1 代表星期一  
  %U    一年中的第几周，以周日为每星期第一天(00-53)  
  %V    ISO-8601 格式规范下的一年中第几周，以周一为每星期第一天(01-53)  
  %w    一星期中的第几日(0-6)，0 代表周一  
  %W    一年中的第几周，以周一为每星期第一天(00-53)  
  %x    当前locale 下的日期描述 (如：12/31/99)  
  %X    当前locale 下的时间描述 (如：23:13:48)  
  %y    年份最后两位数位 (00-99)  
  %Y    年份  
  %z +hhmm              数字时区(例如，-0400)  
  %:z +hh:mm            数字时区(例如，-04:00)  
  %::z +hh:mm:ss        数字时区(例如，-04:00:00)  
  %:::z                 数字时区带有必要的精度 (例如，-04，+05:30)  
  %Z                    按字母表排序的时区缩写 (例如，EDT)  
  
默认情况下，日期的数字区域以0 填充。  
以下可选标记可以跟在"%"后:  
  
  - (连字符)不填充该域  
  \_ (下划线)以空格填充  
  0 (数字0)以0 填充  
  ^ 如果可能，使用大写字母  
  # 如果可能，使用相反的大小写  
  
在任何标记之后还允许一个可选的域宽度指定，它是一个十进制数字。  
作为一个可选的修饰声明，它可以是E，在可能的情况下使用本地环境关联的  
表示方式；或者是O，在可能的情况下使用本地环境关联的数字符号。

### 时间输出

date是Linux系统里自带的一个系统命令，用来显示当前的系统时间，不过默认显示的结果里包括很多信息，特别是做为文件名输出时，不是很方便  
好在date命令里包含格式化输出的选项

**[python]** [view plain](http://blog.csdn.net/jk110333/article/details/8590746/) [copy](http://blog.csdn.net/jk110333/article/details/8590746/)

1. [root@root ~]# date "+%Y-%m-%d"
2. 2013-02-19
3. [root@root ~]# date "+%H:%M:%S"
4. 13:13:59
5. [root@root ~]# date "+%Y-%m-%d %H:%M:%S"
6. 2013-02-19 13:14:19
7. [root@root ~]# date "+%Y\_%m\_%d %H:%M:%S"
8. 2013\_02\_19 13:14:58
9. [root@root ~]# date -d today
10. Tue Feb 19 13:10:38 CST 2013
11. [root@root ~]# date -d now
12. Tue Feb 19 13:10:43 CST 2013
13. [root@root ~]# date -d tomorrow
14. Wed Feb 20 13:11:06 CST 2013
15. [root@root ~]# date -d yesterday
16. Mon Feb 18 13:11:58 CST 2013

### 时间设置

我们一般使用“date -s”命令来修改系统时间  
比如将系统时间设定成2011年7月13日的命令：date -s 07/13/2011  
将系统时间设定成11点12分0秒的命令：date -s 11:12:00  
**注意**：这里说的是系统时间，是linux由操作系统维护的。  
在系统启动时，Linux操作系统将时间从CMOS中读到系统时间变量中，以后修改时间通过修改系统时间实现。为了保持系统时间与CMOS时间的一致性，Linux每隔一段时间会将系统时间写入CMOS。由于该同步是每隔一段时间（大约是11分钟）进行的，在我们执行date -s后，如果马上重起机器，修改时间就有可能没有被写入CMOS,这就是问题的原因。  
如果要确保修改生效可以执行如下命令。  
#clock –w  
这个命令强制把系统时间写入CMOS。

时间加减

通过-d参数  
date +%Y%m%d         //显示现在天年月日  
date +%Y%m%d –d "+1 day"  //显示后一天的日期  
date +%Y%m%d –d "-1 day"  //显示前一天的日期  
date +%Y%m%d -d "-1 month"  //显示上一月的日期  
date +%Y%m%d –d "+1 month"  //显示下一月的日期  
date +%Y%m%d -d "-1 year"  //显示前一年的日期  
date +%Y%m%d -d "+1 year"  //显示下一年的日期

或者更简单点的  date=`date -d -${t}day '+%Y%m%d'` //为t为前几天

date +%F-%H-%M-%S -d '-2HOUR' //当前日期-2小时

date +%F-%H-%M-%S -d '-20MIN //当前日期-20分

date +%F-%H-%M-%S -d '-20SEC' //当前日期-20秒

## whereis 查找二进制命令路径

### 意义

切换当前目录 至 另一个目录

### 命令格式

cd [参数] [dirname]

切换到 dirname 目录

### 参数

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 |  |
|  |  |  |

### 例子

## Linux用户权限

Linux作为一种多用户多任务操作系统，在日常的使用中不可避免地要划分出一个角色的概念来管理和使用计算机，这个角色与每一个计算机使用者关联，在Linux中称这种角色为用户。而在每一个用户使用计算机的过程中,又必然存在对有限计算机资源使用的限制性，那么操作系统就必须提供一种途径来保证每个用户独立、合理的使用计算机。

### 用户和用户组管理

（一）用户及用户组相关基本概念

用户：泛指计算机的使用者。用计算机可识别的用户ID（UID，user id）标识。

用户组：用户容器，用来将多个用户合并为一个单一的逻辑组件。用计算机可识别的用户组ID（GID，group id）标识。

UID和GID：都是16位二进制数，其范围为0-65535，共计2^16=65536个。

用户类别：

（1）管理员用户：被称为root，UID为0。

（2）系统用户：为系统运行提供服务的非登陆式用户，UID从1-999这个范围进行分配。

（3）登陆用户：真正使用计算机的普通用户，UID从500-60000进行分配。

组类别：Linux系统中用户组可按三种方式划分：

1、与类似用户的分类方式

（1）管理员组：管理员所在组，GID为0。

（2）系统用户组：系统用户所在组，GID为1-999。

（3）登陆用户组：登陆用户所在组，GID为1000-60000。

2、从用户角度出发分类

（1）用户的基本组：每个用户的主要组。

（2）用户的附加组：每个用户加入的其他组。

3、从用户组的角度出发分类

（1）私有组：组名和用户名相同，且只包含一个用户。

（2）公共组：组名包含多个用户。

认证信息：用户名和密码，通过比对事先存储的信息与登陆时用户提供的信息是否一致来认证用户。

那么遵从Linux中“一切皆文件”的哲学理念，认证信息也保存在文件来中。我们来认识三个文件分别是/etc/passwd、/etc/group和/etc/shadow，这三个文件分别保存了用户的信息、组的信息和密码的信息，当用户登陆时就是通过比对这三个文件中的信息来实现认证，接下来，我们对这三个文件的内容格式作出解释。

（1）/etc/passwd 文件字段说明

name:password:UID:GID:GECOS:directory:shell

① name：用户名

② password：可以是加密的密码，也可以是占位符x：

③ UID：用户ID

④ GID：用户所属的基本组ID

⑤ GECOS：可选，用户的注释信息

⑥ directory：用户的家目录

⑦ shell：用户的默认shell（登陆时）

（2）/etc/shadow 文件字段说明

① name:用户名。

② $加密算法$salt$加密密码：salt称为杂质，加密时添加。

③ 最近一次修改密码的时间：表示从1970年1月1日（unix元年）开始的天数；0表示用户下次登录时应该更改密码；空字段表示这个功能被禁止。

④ 最短使用期限：从更改密码开始，下次可更改密码经过的天数。

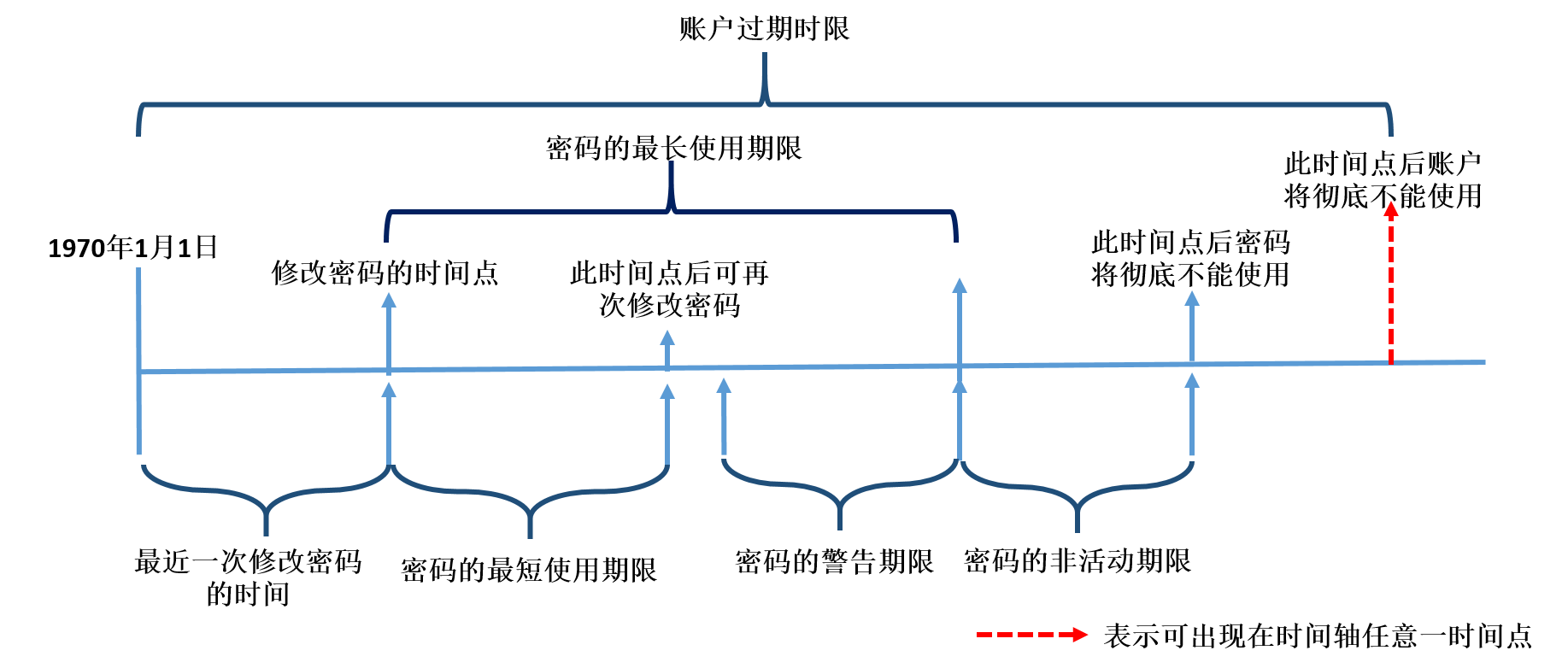
⑤ 最长使用期限：从更改密码开始，可合法使用此密码的天数。

⑥ 警告期限：在最长使用期限过期之前，警告密码应更改的天数。

⑦ 非活动期限：在最长使用期限后，此密码还可被使用的一段时间。

⑧ 账户过期期限：从1970年开始，账户可被使用的天数。

⑨ 保留字段



（3）/etc/group 文件字段说明

group\_name:password:GID:user\_list

① group\_name：用户组名

② passwd：组密码

③ GID:用户组ID

④ user\_list：以该组为其附加组的用户的用户列表；

### 用户和用户组管理相关命令

命令概述：useradd，userdel，usermod，passwd，groupadd，groupdel，groupmod，gpasswd，chage，id，su，chown，chgrp

### groupadd  添加组

使用格式 ：groupadd [options] group

-g，--gid GID  指定GID；如不指定则默认是上一个已存在组的GID加1

-r, --system  创建系统组

# groupadd mygroup

# groupadd -r mygroup

2、groupmod  修改组属性

使用格式 ： groupmod [options] GROUP

-g GID: 修改GID

-n new\_name:修改组名

#  groupmod -g 1002 mygroup

#  groupmod -n mynewgroup mygroup

3、groupdel  删除组

使用格式：    groupdel [options] group

#  groupdel mynewgroup

### useradd  创建用户

使用格式：    useradd [options] LOGIN

-u，--uid UID：指定UID；

-g，--gid GROUP：指定基本组ID，此组需要事先存在

-G, --groupsGROUP1[,GROUP2,...[,GROUPN]]]：指明用户所属的附加组，多个组之

间用逗号隔开

-c，--comment COMMENT：指定注释信息

-d，--homeHOME\_DIR:  以指定的路径为用户的家目录；通过复制/etc/skel此目录并

重命名来实现：指定的家目录路径事先存在，则不会为用户复制环境配置文件；

-s，--shell SHELL：指定用户的默认shell，而可用的所有shell列表存储在/etc/shells文

件当中；

-r，--system：创建系统用户；

-M  不创建用户主目录

-f    密码过期后的使用天数，0表示禁用，-1表示永不禁用

-D  显示创建用户的默认设定

注意：创建用户时的诸多默认设定在/etc/login.defs文件中

#  useradd myuser

### usermod  更改用户属性

使用格式：    usermod [options] LOGIN

-u，--uid UID ：修改用户ID

-g，--gid GROUP  修改用户所属的基本组，

-G, --groups GROUP1[,GROUP2,...[,GROUPN]]]：修改用户所属的附加组；原来的

附加组会被覆盖

-a，--append：与-G一起使用，用于为用户追加新的附加组；

-c，--comment COMMENT：修改注释信息

-d，--home HOME\_DIR: 修改用户的家目录，用户原有的文件不会被转移至新位置；

-m，--move-home：只能与-d一起使用，用于将用户的主目录移动到新位置

-l，--login LOGIN：修改用户登录名

-s，--shell SHELL：修改用户的默认shell

如：usermod -s /usr/bin alivn 修改alivn的shell

-L，--lock：锁定用户密码

-U，--unclock：解锁用户的密码；

#  将用户myuser的家目录更改为/tmp/home/user，并且原来的用户配置不变，且更改用户的默认shell为tcsh

#  usermod -s /bin/tcsh -d /tmp/home/user -m myuser

### userdel  删除用户

使用格式：    userdel[option] login

-r  删除用户时，一并删除其家目录

注意：当删除一个用户时，其基本组也会被删除。

#  删除用户myuser

#  userdel myuser

### passwd  密码管理

使用格式：    passwd [-k] [-l][-u [-f]] [-d] [-e] [-n mindays] [-x maxdays] [-w warndays] [-i

inactivedays] [-S][--stdin] [username]

（1）passwd：修改用户自己的密码

（2）passwd USERNAME：修改指定用户的密码，但仅root用户有此权限，且不需要知道

原密码；

-l，-u：锁定和解锁用户

-d：清除用户密码串

-e DATE：设定过期期限；

-i DAYS：非活动期限；

-n DAYS: 密码的最短使用期限

-x DAYS: 密码的最长使用期限

-w DAYS：警告期限

--stdin：从标准输入读入密码。

通常在脚本中如此使用：echo "PASSWORD" | passwd--stdin USENAME

#  更改用户user的密码

#  passwd myuser

### gpasswd： 更改用户组密码

使用格式：    gpasswd [option]  group

-a USERNAME: 向组中添加用户

-d USERNAME: 从组中移除用户

#  更改用户组密码

#  gpasswd mygroup

#  将用户myuser添加进mygroup组中，即让mygroup成为myuser的附加组。

#  gpasswd myuser

### chage 更改密码过期信息

使用格式：    chage [options] LOGIN

-d：更改上次修改密码的期限

-E：更改账户过期期限

-I：更改密码的非活动期限

-W：更改警告期限

-m：更改密码最短使用期限

-M：更改密码最长使用期限

10、id  显示用户的真实和有效 id 信息

使用格式：    id[OPTION]... [USER]

id：显示自己的ID信息

-u：仅显示有效UID

-g：仅显示有效基本组ID

-G：显示用户所属的所有组的ID

-n：不显ID而显示名称（配合以上显示ID的选项一起用）

11、su  切换用户

登陆时切换：会重新读取目标用户的配置文件来重新初始化

su- USERNAME

su-l USERNAME

非登陆式切换：不会读取目标用户的配置文件来进行初始化

suUSERNAME

-c 'COMMAND':仅以指定用户的身份运行此处指定的命令

注意：管理员可无密码切换至其他任何用户；非管理员必须要给出目标用户的登陆密码

### chown 更改文件属主或者属组或者属主和属组

功能：更改某个文件或目录的属主和属组。这个命令也非常常用。例如root用户把自己的一个文件拷贝给用户xu，为了让用户xu能够存取这个文件，root用户应该把这个文件的属主设为xu，否则，用户xu无法存取这个文件。

语法:

chown [选项] 用户或组 文件

说明:chown将指定文件的拥有者改为指定的用户或组。用户能是用户名或用户ID。组能是组名或组ID。文件是以空格分开的要改动权限的文件列表，支持通配符。

该命令的各选项含义如下：

- R 递归式地改动指定目录及其下的所有子目录和文件的拥有者。

- v 显示chown命令所做的工作。

使用格式：      chown [OPTION]... [OWNER][:|.[GROUP]] FILE...

如：chown alivn(用户).javausers(用户组) /usr/linuxtest

将linuxtest的属主改为alivn 用户组改为 javausers

注意:用户仅能修改属主为自己的那些权限

#  将文件file1的属组和属主更改为mygroup和myuser

#  chown myuser:mygroup file1

### chgrp 仅更改属组

功能：改动文件或目录所属的组。

语法：chgrp [选项] group filename

该命令改动指定指定文件所属的用户组。其中group能是用户组ID，也能是/etc/group文件中用户组的组名。文件名是以空格分开的要改动属组的文件列表，支持通配符。如果用户不是该文件的属主或终极用户，则不能改动该文件的组。

该命令的各选项含义为：

- R 递归式地改动指定目录及其下的所有子目录和文件的属组。

如：chgrp javausers file.txt

将file.txt的属主 改为javausers

### umask默认权限

当我们登录系统之后创建一个文件总是有一个默认权限的，那么这个权限是怎么来的呢？这就是umask干的事情。umask设置了用户创建文件的默认 权限，它与chmod的效果刚好相反，umask设置的是权限“补码”，而chmod设置的是文件权限码。一般在/etc/profile、$ [HOME]/.bash\_profile或$[HOME]/.profile中设置umask值。

　　你的系统管理员必须要为你设置一个合理的 umask值，以确保你创建的文件具有所希望的缺省权限，防止其他非同组用户对你的文件具有写权限。在已经登录之后，可以按照个人的偏好使用umask命 令来改变文件创建的缺省权限。相应的改变直到退出该shell或使用另外的umask命令之前一直有效。一般来说，umask命令是在/etc /profile文件中设置的，每个用户在登录时都会引用这个文件，所以如果希望改变所有用户的umask，可以在该文件中加入相应的条目。如果希望永久 性地设置自己的umask值，那么就把它放在自己$HOME目录下的.profile或.bash\_profile文件中。

umask 命令允许你设定文件创建时的缺省模式，对应每一类用户(文件属主、同组用户、其他用户)存在一个相应的umask值中的数字。

对于文件来说，这一数字的最 大值分别是6。系统不允许你在创建一个文本文件时就赋予它执行权限，必须在创建后用chmod命令增加这一权限。

目录则允许设置执行权限，这样针对目录来 说，umask中各个数字最大可以到7。

手工定义umask值

　　umask nnn

其中nnn为umask置000-777。

**加减法计算文件权限**

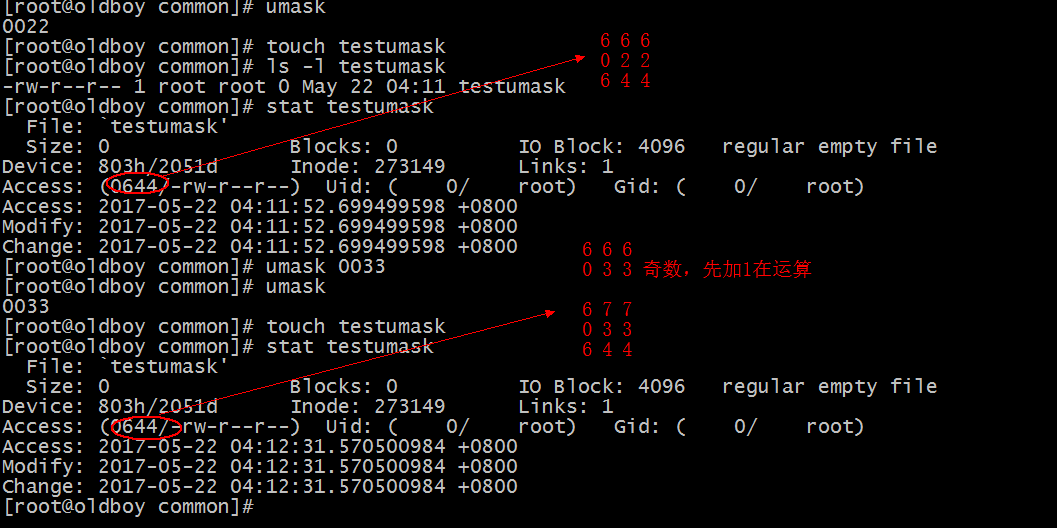
对于文件来说，最大位为6 ，即666

如果umask默认为：022，则对应位相减，得出默认权限是：644

（1）：如果umask位是偶数，直接相减

（2）：如果umask位是基数，对应位+1后再做减法

例如：umask 033 ,则666的后两位+1为677在做运算，得出644



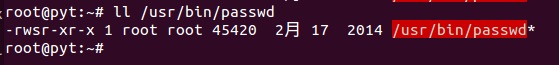
### SETUID SETGID 粘滞位

（1）进程运行时能够访问哪些资源或文件，不取决于进程文件的属主属组，而是取决于运行该命令的用户身份的uid/gid，以该身份获取各种系统资源。

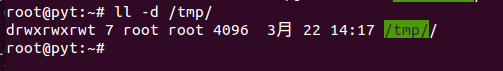
（2）对一个属主为root的可执行文件，如果设置了SUID位，则其他所有普通用户都将可以以root身份运行该文件，获取相应的系统资源。

（3）可以简单地理解为让普通用户拥有可以执行“只有root权限才能执行”的特殊权限。

（4）setuid，setuid的作用是让执行该命令的用户以该命令拥有者的权限去执行，比如普通用户执行passwd时会拥有root的权限，这样就可以修改/etc/passwd这个文件了。它的标志为：s，会出现在x的地方，例：-rwsr-xr-x  。而setgid的意思和它是一样的，即让执行文件的用户以该文件所属组的权限去执行。



（5）我们知道/tmp是系统的临时文件目录，所有的用户在该目录下拥有所有的权限，也就是说在该目录下可以任意创建、修改、删除文件，那如果用户A在该目录下创建了一个文件，用户B将该文件删除了，这种情况我们是不能允许的。为了达到该目的，就出现了stick  bit(粘滞位)的概念。它是针对目录来说的，如果该目录设置了stick  bit(粘滞位)，则该目录下的文件除了该文件的创建者和root用户可以删除和修改/tmp目录下的stuff，别的用户均不能动别人的，这就是粘滞位的作用。



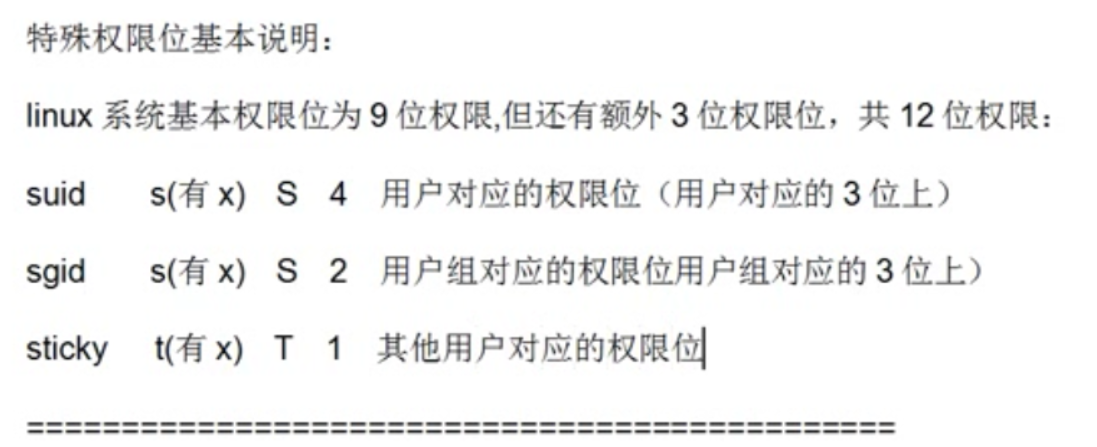
Chmod命令中的特殊权限位含义：

1、S\_ISUID 04000 文件的 (set user-id on execution)位

2、S\_ISGID 02000 文件的 (set group-id on execution)位

3、S\_ISVTX 01000 文件的sticky 位

上面的都是8进制数。



如何设置UID、GID、STICK\_BIT：

SUID：置于 u 的 x 位，原位置有执行权限，就置为 s，没有了为 S .

chmod u+s  xxx # 设置setuid权限

chmod 4551 file // 权限： r-sr-x—x

SGID：置于 g 的 x 位，原位置有执行权限，就置为 s，没有了为 S .

chmod g+s  xxx # 设置setgid权限

chmod 2551 file // 权限： r-xr-s--x

STICKY：粘滞位，置于 o 的 x 位，原位置有执行权限，就置为 t ，否则为T .

chmod o+t  xxx # 设置stick bit权限，针对目录

chmod 1551 file // 权限： r-xr-x--t