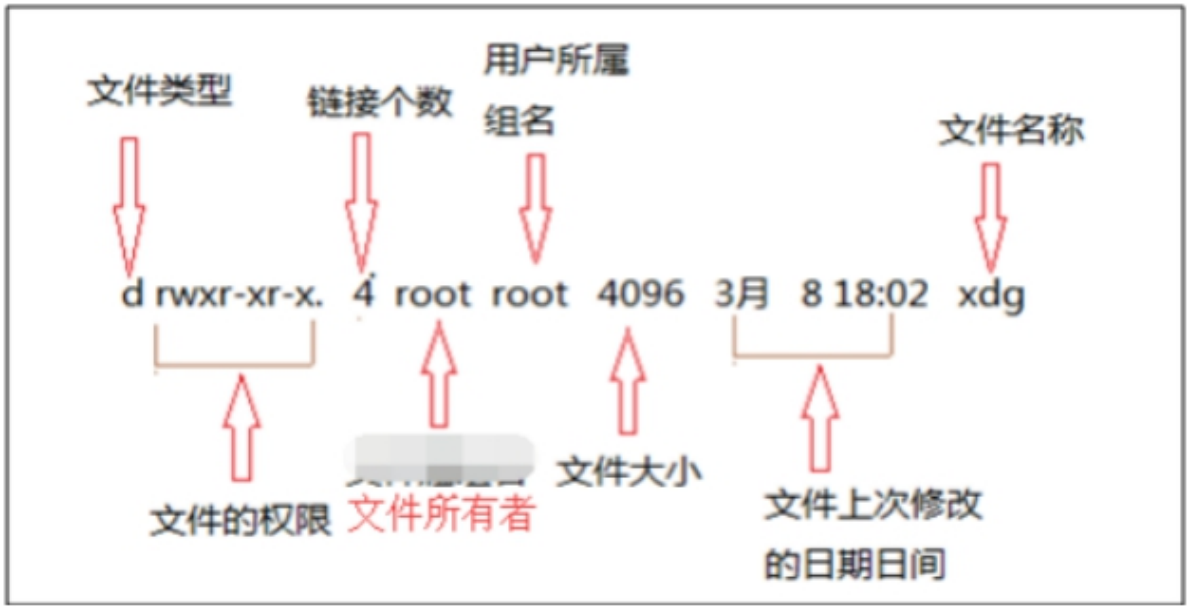
文件的属性

1.查看权限

```
[root@exercise1 ~]# ll
总用量 16
-rw-------.  1 root root 1390 1月   9 09:31 anaconda-ks.cfg
drwxr-xr-x.  3 root root   18 1月  11 08:36 boot
drwxr-xr-x. 74 root root 8192 1月  12 23:17 etc
drwxr-xr-x.  3 root root   18 1月  11 08:36 opt
[root@exercise1 ~]#
```



2.第一个字符文件类型中

d： 目录文件　　l： 链接文件　　b： 块设备文件　　c： 字符设备文件　　p： 管道文件　　-: 表示普通文件

为什么有的目录文件有不同的颜色呢?

linux 系统中不同的颜色代表了不同的文件类型

```
颜色                代表内容                举例
蓝色                目录                    /etc
黑色                文件                    /etc/passwd
浅蓝色              链接                    /etc/grub2.cfg
红色                压缩包                  boot.tar.gz
绿色                可执行文件              /etc/init.d/network
黄色                设备文件                /dev/sda
```

3.file 查看文件 file 命令

作用： file - determine file type　#确定文件类型

用法： file /etc/passwd

注：linux 系统不根据后缀名识别文件类型

```
[root@exercise1 ~]# vim /opt/a.txt
[root@exercise1 ~]# file /opt/a.txt
/opt/a.txt: ASCII text
```

权限基本概述

1.什么是权限?

权限主要用来约束用户能对系统所做的操作。或者说，权限是指某个特定的用户具有特定的系统资源使用权力。

2.为什么要有权限?

因为系统中不可能只存在一个root用户，一定会有多个用户，为了保护每个登陆用户的隐私和工作环境，所以就有了权限。

3.权限与用户之间的关系?

在linux系统中，权限是用来定义用户能做什么，不能做什么。

比如：针对文件定义了三种身份，分别是属主(owner）、属组（group）、其他人(others)，

而为每一种身份又对应三种权限下分别是可读（readable)、可写（writable）、可执行（excutable)。

1.UGO： 所有者--用户组--其它用户

所有者： 就是创建文件的用户，这个用户拥有对它所创建的文件的一切权限，所有者可以允许其所在的用户组可以访问所有者文件。

用户组： 用户组是具有相同特征用户的逻辑集合，有时我们需要让多个用户具有相同的权限，比如查看、修改 某一个文件的权限，

一种方法是分别对多个用户进行文件访问授权，如果有10个用户的话，就需要授权 10次，显然这种方法不太合理；

另一种方法是建立一个组，让这个组具有查看、修改此文件的权限，然后将所有需要访问此文件的用户放入这个组中，那么所有用户就具有了和组一样的权限。这就是用户组。

其它用户：系统内的其他所有者用户就是 other 用户类

当一个用户访问文件流程如下：

1)判断用户是否为文件所有者，如果是则按所有者的权限进行访问

2)判断用户是否为文件用户组成员，如果是则按组的权限进行访问

3)如果不是所有者，也不是该文件所属组，则按（其它人）匿名权限进行访问

```
例子：
#每个用户都拥有自己的专属目录，通常放置/home下
[root@exercise1 ~]# ll /home/
```

```
总用量 0
drwx------. 2 lin05  lin05  83 1月  17 19:37 lin05
drwx------. 2 oracle oracle 62 1月  17 17:29 oracle

#注：  [rwx------]表示目录所有者本身拥有的权限，其它用户是无法进入的。 root 可以。

#你以什么用户身份登录，那么你创建的文件或目录，自动成为该文件的所属主和组
[abc@exercise1 ~]$ touch a.txt
[abc@exercise1 ~]$ ll a.txt
-rw-r--r--. 1 abc lin02 0 1月  18 19:11 a.txt
[abc@exercise1 ~]$
```

2.更改文件的属主和属组

改变文件的所属关系用到命令：

chown： 可以用来改变文件(或目录)的属主

chgrp： 可以用来改变文件（或目录）的默认属组 如果你要对目录进行操作，加参数 -R

chown

语法：

```
chown user:group filename      比如：chown hr:san a.txt 把文件的属主和属组改为 hr,san
chown user filename   比如：chown san a.txt   把文件的属主改为 san 用户
chown :group filename   比如： chown :miao a.txt     把文件的属组改为 miao 这个组
chown user: filename 比如：chown san: a.txt   自动继承这个用户所有的组
```

例子：

```
[root@exercise1 opt]# touch {a..c}.txt
```

```
[root@exercise1 opt]# ll
总用量 0
-rw-r--r--. 1 root root 0 1月  18 19:31 a.txt
-rw-r--r--. 1 root root 0 1月  18 19:31 b.txt
-rw-r--r--. 1 root root 0 1月  18 19:31 c.txt
[root@exercise1 opt]# chown lin05 a.txt
[root@exercise1 opt]# ll a.txt
-rw-r--r--. 1 lin05 root 0 1月  18 19:31 a.txt
[root@exercise1 opt]# chown lin05:lin05 a.txt
[root@exercise1 opt]# ll a.txt
-rw-r--r--. 1 lin05 lin05 0 1月  18 19:31 a.txt
[root@exercise1 opt]# chown :root a.txt
[root@exercise1 opt]# ll a.txt
-rw-r--r--. 1 lin05 root 0 1月  18 19:31 a.txt
[root@exercise1 opt]#
```

chgrp

语法:

chgrp hr filename 比如： chgrp hr f.txt

-R ： 递归（目录下的所有内容都更改，否则只修改目录）

实战：一个文件只有读的权限，拥有者是否可以写这个文件?

```
[root@exercise1 opt]# su - lin05
上一次登录：一 1月 17 19:33:24 CST 2022pts/1 上
[lin05@exercise1 ~]$ touch a.txt
[lin05@exercise1 ~]$ ll a.txt
-rw-rw-r--. 1 lin05 lin05 0 1月  18 19:35 a.txt
#在另一个终端上，以 root 身份登录：
[root@exercise1 ~]# chmod 000 /home/lin05/a.txt   #修改成 000 权限
[root@exercise1 ~]# ll /home/lin05/a.txt
----------. 1 lin05 lin05 0 1月  18 19:35 /home/lin05/a.txt
#回到以 lin05 身份登录的终端：
[lin05@exercise1 ~]$ vi a.txt   # 写入 aaa   ,   :wq! 保存
#在另一个终端上，以 root 身份登录：
[root@exercise1 ~]# cat /home/lin05/a.txt
```

```
14  aaa
```

实验结果：文件所有者一定可以写文件。 就像 root 可以对 shadow 强制写。 因shadow 的拥有者是 root

3.修改权限

使用字符设定

修改权限用的命令：chmod 作用：修改文件，目录的权限

语法：chmod [对谁操作] [操作符] [赋于什么权限] 文件名

```
对谁操作：
u----> 用户user ， 表示文件或目录的所有者
g---->用户组 group ， 表示文件或目录所属的用户组
o---->其它用户 others
a---->所有用户 all 操作符
+    #添加权限
-    # 减少权限
=    #直接给定一个权限

权限：r w x
```

列举部分组合：

对谁操作权限	对谁操作	对谁操作意思
u-w	user	拥有者
g+x	group	组
o=r	other	其他人
a+x	all	所有人

权限对文件和目录的影响

有三种权限可以应用：读取，写入不执行，这些权限对访问文件和目录的影响如下：

权限	对文件的影响	对目录的影响
r(读取)	可以读取文件的内容(cat)	可以列出目录的内容(文件名)ls
w(写入)	可以更改文件的内容(vi,echo)	可以创建或删除目录中的任意文件 touch,mkdir,rm
x(执行)	可以作为命令执行文件(sh)	可以访问目录的内容（取决于目录中文件的权限）cd

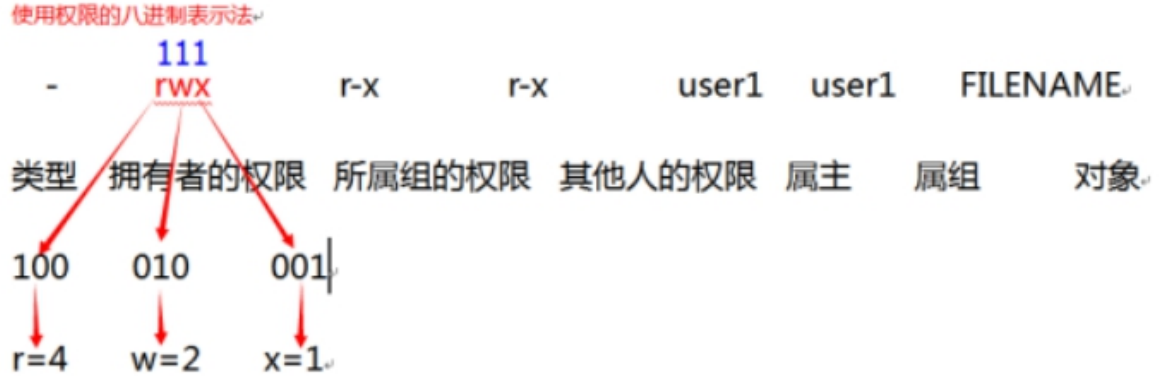
例子：chmod 修改权限

```
[root@exercise1 opt]# touch /opt/1.txt
[root@exercise1 opt]# ll /opt/1.txt
-rw-r--r--. 1 root root 0 1月  18 20:01 /opt/1.txt
[root@exercise1 opt]# chmod u-w /opt/1.txt
[root@exercise1 opt]# ll /opt/1.txt
-r--r--r--. 1 root root 0 1月  18 20:01 /opt/1.txt
[root@exercise1 opt]# chmod g+x /opt/1.txt
[root@exercise1 opt]# ll /opt/1.txt
-r--r-xr--. 1 root root 0 1月  18 20:01 /opt/1.txt
[root@exercise1 opt]# chmod a+x /opt/1.txt   # 后期给shell 脚本加一个可执行权限
[root@exercise1 opt]# ll /opt/1.txt
-r-xr-xr-x. 1 root root 0 1月  18 20:01 /opt/1.txt
[root@exercise1 opt]# chmod a=rwx /opt/1.txt
[root@exercise1 opt]# ll /opt/1.txt
-rwxrwxrwx. 1 root root 0 1月  18 20:01 /opt/1.txt
[root@exercise1 opt]#
```

4使用八进制（0-7）数字表示权限法

权限	二进制	八进制	描述

--- | 000 | 0 | 没有任何权限
--x | 001 | 1 | 只有执行权限
-w- | 010 | 2 | 只有写入权限
-wx | 011 | 3 | 有写入和执行权限
r-- | 100 | 4 | 只有读取权限
r-x | 101 | 5 | 有读取权限和执行权限
rw- | 110 | 6 | 有读取和写入权限
rwx | 111 | 7 | 有全部权限



互动：rw- 的值是多少？ 答： 4+2=6

rwx r-x r-x 的值是多少？ 答： rwx=4+2+1=7 ； r-x=4+1=5 rwx r-x r-x=7 5 5

语法：

```
chmod 755 文件或文件夹名字
chmod a=rwx b.txt   等于 chmod 777 b.txt
```

例子：

```
[root@exercise1 ~]# touch /opt/dd.txt
[root@exercise1 ~]# ll /opt/dd.txt
-rw-r--r--. 1 root root 0 1月  18 22:35 /opt/dd.txt
[root@exercise1 ~]# chmod 755 /opt/dd.txt
[root@exercise1 ~]# ll /opt/dd.txt
-rwxr-xr-x. 1 root root 0 1月  18 22:35 /opt/dd.txt
[root@exercise1 ~]# chmod 700 /opt/dd.txt
[root@exercise1 ~]# ll /opt/dd.txt
-rwx------. 1 root root 0 1月  18 22:35 /opt/dd.txt
```

5.补码

为什么我们创建的文件的权限是 644 呢?

我们创建文件的默认权限是怎么来的?

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

文件默认权限 = 666 ， 目录默认权限 = 777

我们一般在/etc/profile、$[HOME]/.bash_profile$或[HOME]/.profile 中设置 umask 值。
永久生效，编辑用户的配置文件vim .bash_profile

```
[root@exercise1 ~]# vi /etc/profile
```

```
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
    umask 002
else
    umask 022
fi
```

注： UID 大于 199 且用户的组名和用户名一样，那么 umask 值为002，否则为022.

注： -gt 在shell中表示大于； id -g 显示用户组 ID ， id -gn 显示组名。

```
临时生效： umask 权限补码
[root@exercise1 ~]# umask 044
[root@exercise1 ~]# touch /opt/ss.txt
[root@exercise1 ~]# ll /opt/ss.txt
-rw--w--w-. 1 root root 0 1月  18 22:54 /opt/ss.txt
```

权限的算法:

一般情况是：目录默认权限-umask 值

666-022=644

777-022=755

#这是一个好的记忆方法，但不严谨。

互动：umask 掩码为033 创建普通文件后，权限是什么？ 666-033=633 (rw- -wx -wx) ?

```
[root@exercise1 ~]# umask 033
[root@exercise1 ~]# touch /opt/k.txt
[root@exercise1 ~]# ll /opt/k.txt
-rw-r--r--. 1 root root 0 1月  18 22:56 /opt/k.txt
[root@exercise1 ~]#
```

答：结果为： 644

权限科学的计算方法:

1、将默认权限（目录777，文件 666）和umask 值都转换为2 进制

2、对 umask 取反

3、将默认权限和umask 取反后的值做与运算

4、将得到的二进制值再转换 8 进制，即为权限

```
例 1： umask为022
6   6   6                  umask  0 2 2
110 110 110            000 010 010 #转成二进制
                       111 101 101  # umask 取反的值
110 110 110            与                          #第二步，默认权限和umask 取反后的值做与运算
111 101 101            # umask 取反的值
110 100 100
6      4      4              #转成8进制
```

例 2： umask为033
6 6 6 umask 0 3 3
110 110 110 000 011 011 #转成二进制
111 100 100 # umask 取反的值
110 110 110 与 #第二步，默认权限和umask 取反后的值做与运算
111 100 100 # umask 取反的值
110 100 100
6 4 4 #转成8进制

6.特殊权限

文件的特殊权限：**suid sgid sticky**

1、SUID（set uid 设置用户 ID）：

限定：只能设置在二进制可执行程序上面。对目录设置无效

功能：程序运行时的权限从执行者变更成程序所有者的权限

2、SGID：

限定：既可以给二进制可执行程序设置，也可以对目录设置

功能：在设置了 SGID 权限的目录下建立文件时，新创建的文件的所属组会，继承上级目录的所属组

3、Stickybit：粘滞位权限是针对目录的，对文件无效，也叫防删除位

这 3 个特殊权限对应的数值为

SUID	SGID	Stickybit
u+s 或 u=4	g+s 或 g=2	o+t 或o=1

SUID 属性一般用在可执行文件上，当用户执行该文件时，会临时拥有该执行文件的所有者权限。使用”ls -l” 或者”ll” 命令浏览文件时，如果可执行文件所有者权限的第三位是一个小写的”s”，就表明该执行文件拥有SUID 属性。比如/usr/bin/passwd 文件

```
[root@exercise1 ~]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 6月  10 2014 /usr/bin/passwd
```

互动： 普通用户 abc，没有对 shadow 文件写入的权限， 但是 abc 用户使用 passwd 修改自己密码时，可以 修改 shadow 文件中的内容，这是什么原因?

```
[root@exercise1 ~]# ll /etc/passwd
-rw-r--r--. 1 root root 979 1月  17 19:30 /etc/passwd
[root@exercise1 ~]# su - lin05
上一次登录：二 1月 18 19:35:14 CST 2022pts/0 上
[lin05@exercise1 ~]$ passwd
更改用户 lin05 的密码 。
为 lin05 更改 STRESS 密码。
（当前）UNIX 密码：123456
新的 密码：Xuegod*666
重新输入新的 密码：Xuegod*666
passwd ：所有的身份验证令牌已经成功更新。
[root@base2 ~]# vim /etc/shadow   #查看 shadow 文件已经被 lin05 用户修改成功。
因为 lin05 用户执行 passwd 命令时，权限会提升成 root 用户，所以可以修改成功。
```

SUID（set uid 设置用户 ID）：

限定：只能设置在二进制可执行程序上面。对目录设置无效

功能：程序运行时的权限从执行者变更成程序所有者的权限

```
[root@exercise1 ~]# useradd abc
[root@exercise1 ~]# su - abc
[abc@exercise1 ~]$ less /etc/shadow   #看不到内容
/etc/shadow: 权限不够
[abc@exercise1 ~]$ 登出   #切换到 root ，给一个 suid 权限
[root@exercise1 ~]# chmod u+s /usr/bin/less
[root@exercise1 ~]# su - abc
上一次登录：二 1月 18 23:14:12 CST 2022pts/0 上
[abc@exercise1 ~]$ less /etc/shadow   #看到 查看 u+s 后的效果
[abc@exercise1 ~]$ ll /usr/bin/less
-rwsr-xr-x. 1 root root 158240 7月  31 2015 /usr/bin/less
[root@exercise1 ~]# chmod 4755 /usr/bin/less   # 等同于 chmod u+s /usr/bin/less
```

SGID：

限定：既可以给二进制可执行程序设置，也可以给目录设置。

功能：在设置了 SGID 权限的目录下建立文件时，新创建的文件的所属组会继承上级目录的权限。

```
[root@exercise1 ~]# mkdir /opt/test
[root@exercise1 ~]# ll /opt/
总用量 0
-rwxrwxrwx. 1 root root 0 1月  18 20:01 1.txt
-rwx------. 1 root root 0 1月  18 22:35 dd.txt
-rw-r--r--. 1 root root 0 1月  18 22:56 k.txt
-rw--w--w-. 1 root root 0 1月  18 22:54 ss.txt
drwxr--r--. 2 root root 6 1月  18 23:21 test
[root@exercise1 opt]# chmod g+s /opt/test/
[root@exercise1 opt]# !ll
ll -d test/
drwxr-Sr--. 2 root root 6 1月  18 23:21 test/
[root@exercise1 opt]#
```

Sticky

限定：只作用于目录

功能：目录下创建的文件只有 root、文件创建者、目录所有者才能删除，附属组的也删不了。

```
例：  系统中的 tmp 目录就是这样
[root@exercise1 opt]# ll -d /tmp/
drwxrwxrwt. 10 root root 4096 1月  18 22:27 /tmp/
[root@exercise1 opt]#

用法：
chmod o+t /tmp/test/
```

7.文件扩展权限 ACL

扩展 ACL　：access control list

setfacl [-bkRd] [{-m|-x} acl 参数] 目标文件名

|参数| 作用|

|-m | 设置 acl 参数|

|-x | 删除某项 acl 参数|

|-b | 删除所有 acl 参数|

|-k | 删除预设的 acl 参数|

|-R | 递归设置 acl 参数|

|-d | 设置『预设 acl 参数』只对目录有效 , 在该目录新建的数据会引用|

命令格式 :

『 u:[使用者账号列表]:[rwx] 』 <= = 针对特定用户

『 g:[群组列表]:[rwx] 』 <= = 针对特定用户组

『 m:[rwx] 』 <= = 针对有效权限 mask　　setfacl -m mask::rwx file/dir

『 d:[ug]: 使用者列表 :[rwx] 』 针对预设权限 , 属性将继承到次目录

```
例：设置用户 abc 对文件 a.txt 拥有的 rwx 权限 ，abc 不属于 a.txt 的所属主和组，abc 是 other。

[root@exercise1 opt]# touch /opt/test2.txt
[root@exercise1 opt]# getfacl /opt/test2.txt
getfacl: Removing leading '/' from absolute path names
# file: opt/test2.txt
# owner: root
# group: root
user::rw-
group::r--
other::r--

[root@exercise1 opt]# setfacl -m u:lin05:rwx /opt/test2.txt   # u   :  设置某个用户拥有的权限
[root@exercise1 opt]# getfacl /opt/test2.txt
getfacl: Removing leading '/' from absolute path names
# file: opt/test2.txt
# owner: root
# group: root
user::rw-
user:lin05:rwx
group::r--
mask::rwx
other::r--
[root@exercise1 opt]# su - lin05
上一次登录：二 1月 18 23:05:37 CST 2022pts/0 上
[lin05@exercise1 ~]$ ll /opt/test2.txt
-rw-rwxr--+ 1 root root 0 1月  18 23:27 /opt/test2.txt
```

例 2： 给目录加扩展权限

[root@exercise1 opt]# mkdir /opt/test3

[root@exercise1 opt]# setfacl -m d:u:lin05:rwx /opt/test3/ # -d default设置默认 acl ， 对目录有效，此目录下新建的目录或文件都继承 此 acl 权限

[root@exercise1 opt]# getfacl /opt/test3/
getfacl: Removing leading '/' from absolute path names
file: opt/test3/
owner: root
group: root
user::rwx
group::r--
other::r--
default:user::rwx
default:user:lin05:rwx
default:group::r--
default😳:rwx
default:other::r--

[root@exercise1 opt]# cd /opt/test3/
[root@exercise1 test3]# touch /opt/test3/a.txt
[root@exercise1 test3]# mkdir /opt/test3/data
[root@exercise1 test3]# getfacl /opt/test3/a.txt #因为-d参数，所以test下所有创建的文件和目录都继承了默认的 acl 权限
getfacl: Removing leading '/' from absolute path names
file: opt/test3/a.txt
owner: root
group: root
user::rw-
user:lin05:rwx   #effective:rw-
group::r--
mask::rw-
other::r--

[root@exercise1 test3]# getfacl /opt/test3/data/
getfacl: Removing leading '/' from absolute path names
file: opt/test3/data/
owner: root
group: root
user::rwx
user:lin05:rwx

group::r--
mask::rwx
other::r--
default:user::rwx
default:user:lin05:rwx
default:group::r--
default😅:rwx
default:other::r--

```
例 3 ：给目录下所有文件都加扩展权限
[root@exercise1 test3]# setfacl -R -m u:lee:rw-
testdirectory/      #-R 一定要在-m 前面，表示目录下所有文件
[root@exercise1 test3]# setfacl -x u:abc /tmp/a.txt
                    # 去掉单个权限
[root@exercise1 test3]# setfacl -b   /tmp/a.txt
                      # 去掉所有 acl 权限
```

实战：创建一个让 root 都无法删除的文件

发现 windows 中 有文件删除不了，怎么办？ 使用 360 强制删除，粉碎文件 那么在 Linux 下怎么办?

```
[root@exercise1 ~]# touch hack.sh aa.sh
[root@exercise1 ~]# ll hack.sh aa.sh
-rw-r--r-- 1 root root 0 May 24 21:29 aa.sh    -rw-r--r-
- 1 root root 0 May 24 21:29 hack.sh
[root@exercise1 ~]# rm -rf aa.sh
黑客使用 xshell 悄悄执行在后台添加 attr 扩展属性
[root@exercise1 ~]# chattr   +i hack.sh
删除文件：
[root@exercise1 ~]# rm -rf   hack.sh
#发现删除不了 为什么删除不了？
```

8.从 REHL6 开始，新增加文件系统扩展属性

命令：chattr

参数：　a　只能追加内容　　；　i　不能被修改

+a: 只能追加内容　如：　echo aaa　>> hack.sh

+i：即Immutable，系统不允许对这个文件进行任何的修改。如果目录具有这个属性，那么任何的进程只能修 改目录之下的文件，不允许建立和删除文件。

注：immutable　[ɪˈmju:təbl]　不可改变的　；　Append [əˈpend] 追加

-i　：移除 i 参数。　-a :移除 a 参数

```
[root@exercise1 test3]# lsattr a.txt
---------------- a.txt
[root@exercise1 test3]# chattr +a a.txt
[root@exercise1 test3]# lsattr a.txt
-----a---------- a.txt
[root@exercise1 test3]# echo aa >> a.txt
[root@exercise1 test3]# vim a.txt
```


```
[root@exercise1 test3]# rm -rf a.txt
rm: 无法删除"a.txt": 不允许的操作
```

#i参数同理操作