Linux 云计算集群架构师

学神 IT 教育：从零基础到实战，从入门到精通！

联系方式：

学神 IT 教育官方网站: http://www.xuegod.cn

Linux 云计算架构师进阶学习群 QQ 群: 1072932914

学习顾问：小语老师　　学习顾问：边边老师　　学神微信公众号

微信扫码添加学习顾问微信，同时扫码关注学神公众号了解最新行业动态，获取更多学习资料及答疑就业服务！

第七章　Centos8-文件权限管理

本节所讲内容：

7.1　文件的基本权限：r w x　（UGO）

7.2　文件的特殊权限：suid sgid sticky 和文件扩展权限 ACL

7.3　实战：创建一个让 root 都无法删除的文件

7.1　文件的基本权限

7.1.1　权限的作用

通过对文件设定权限可以达到以下三种访问限制权限：

只允许用户自己访问；

允许一个预先指定的用户组中的用户访问；

允许系统中的任何用户访问。

7.1.2　查看权限

[root@xuegod63 ~]# ll /etc/passwd

-rw-r--r--. 1 root root 2053 9 月　19 2017 /etc/passwd

文件权限基本解释：

-　　rw-　　r--　　r--.　　1 root root 2053 9 月　19 2017 /etc/passwd

-	rwx	r-x	r-x	user1	user1	time	FILENAME
文件类型	拥有者的权限	所属组的权限	其他人的权限	拥有者	属组	最后修改时间	对象

其中：文件类型,可以为 p、d、l、s、c、b 和 –

p 表示命名管道文件

d 表示目录文件

l 表示符号链接文件

-表示普通文件

s 表示 socket 套接口文件，比如我们启用 mysql 时，会产生一个 mysql.sock 文件

c 表示字符设备文件，例：　虚拟控制台　或 tty0

b 表示块设备文件　　例：　sda，　cdrom

例：

[root@xuegod63 ~]# ll /dev/sda /dev/cdrom　/etc/passwd /dev/tty0

lrwxrwxrwx　1 root root　　3 9 月　19 2017 /dev/cdrom -> sr0

brw-rw----　1 root disk 8, 0 9 月　19 2017 /dev/sda

crw--w----　1 root tty　4, 0 9 月　19 2017 /dev/tty0

-rw-r--r--. 1 root root 2053 9 月　19 2017 /etc/passwd　　如下效果图：

```
[root@xuegod63 ~]# ll /dev/sda /dev/cdrom  /etc/passwd /dev/tty0
lrwxrwxrwx  1 root root    3 9月   19 2017 /dev/cdrom -> sr0
brw-rw----  1 root disk 8,  0 9月   19 2017 /dev/sda
crw--w----  1 root tty  4,  0 9月   19 2017 /dev/tty0
-rw-r--r--. 1 root root 2053 9月   19 2017 /etc/passwd
```

7.1.3 权限说明

对于文件来说：

r：读　　cat

w：写　　vim echo

x：执行　　运行命名或者脚本

对于目录来说：

r：读（看到目录里面有什么）　ls

w：在目录里面建文件，删除，移动　touch mkdir　rm　mv cp

x：目录是不能直接运行的，对目录赋予 x 权限，代表用户可以进入目录，也就是说，赋予 x 权限的用户或群组可以使用 cd 命令。

7.1.4 文件拥有者

UGO：所有者--用户组--其它用户

User-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的文件。

Group-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。

Other-其它用户：系统内的其他所有者用户就是 other 用户类

7.1.5 常见几种文件权限组成

- rwx --- ---：文件所有者对文件具有读取、写入和执行的权限。

- rwx r-- r--：　文件所有者具有读、写与执行的权限，用户组里用户及其他用户则具有读取的权限

- rw- rw- r-x：文件所有者与同组用户对文件具有读写的权限，而其他用户仅具有读取和执行的权限。

drwx--x--x：　目录所有者具有读写与进入目录的权限,其他用户近能进入该目录，却无法读取任何数据。

drwx------：　除了目录所有者具有完整的权限之外，其他用户对该目录完全没有任何权限。

举例如下：

每个用户都拥有自己的专属目录，通常放置/home 下

```
[root@xuegod63 home]# ll /home/
总用量 0
drwx------. 3 user1 user1 78 9 月　19 2017 user1
```

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。

例 2: 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

```
[root@xuegod63 ~]# useradd user1
[root@xuegod63 ~]# su - user1
[user1@xuegod63 ~]$ touch a.txt
[user1@xuegod63 ~]$ ll a.txt
-rw-rw-r-- 1 user1 user1 0 5 月　 8 20:58 a.txt
```

互动： su - user1 和 su user1 的区别?

```
[root@xuegod63 ~]# su - user1    #从 root 切换到普通用户 user1
user1@xuegod63 ~]$ pwd
/home/user1
[user1@xuegod63 ~]$ exit
[root@xuegod63 ~]# su user1
[user1@xuegod63 root]$ pwd    #查看切完后的路径
/root
[user1@xuegod63 root]$ touch a.txt
touch: 无法创建 'a.txt': 权限不够
[user1@xuegod63 root]$ cd /home/user1/
[user1@xuegod63 ~]$ touch a.txt
[user1@xuegod63 ~]$ exit
```

注： su - 用户，加上-减号，切换用户时，会把用户家目录和环境变量都彻底切成用 user1 的。 不加-减号，保留切换前的路径和环境变量。 一般切换用户都加减号- 。

7.1.6 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（不常用）

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 把文件的属主改为 hr 属组改为 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 组

:也可以用.代替

-R ： 递归（目录下的所有内容都更改，否则只修改目录）

例：

```
[root@xuegod63 ~]# touch {a,b,c}.txt
[root@xuegod63 ~]# ll *.txt
-rw-r--r-- 1 root root 0 5 月  8 21:03 a.txt
-rw-r--r-- 1 root root 0 5 月  8 21:03 b.txt
-rw-r--r-- 1 root root 0 5 月  8 21:03 c.txt

[root@xuegod63 ~]# chown user1 a.txt
[root@xuegod63 ~]# ll a.txt
-rw-r--r-- 1 user1 root 0 5 月  8 21:03 a.txt
[root@xuegod63 ~]# chown user1:user1 a.txt
[root@xuegod63 ~]# ll a.txt
-rw-r--r-- 1 user1 user1 0 5 月  8 21:03 a.txt
[root@xuegod63 ~]# chown :root  a.txt
[root@xuegod63 ~]# ll a.txt
```

-rw-r--r-- 1 user1 root 0 5 月 8 21:03 a.txt
[root@xuegod63 ~]# chown .bin a.txt
[root@xuegod63 ~]# ll a.txt
-rw-r--r-- 1 user1 bin 0 5 月 8 21:03 a.txt

互动：一个文件只有读的权限，拥有者是否可以写这个文件?
实验：
[root@xuegod63 ~]# su - user1
[user1@xuegod63 ~]$ touch a.txt
[user1@xuegod63 ~]$ ll a.txt
-rw-rw-r-- 1 user1 user1 0 5 月 8 21:07 a.txt
[user1@xuegod63 ~]$ chmod 000 a.txt #修改成 000 权限
[user1@xuegod63 ~]$ ll a.txt
---------- 1 user1 user1 4 3 月 19 10:48 a.txt
[user1@xuegod63 ~]$ vim a.txt # 写入 aaa ， :wq! 保存
在另一个终端上，以 root 身份登录：
[root@xuegod63 ~]# cat /home/user1/a.txt
aaaaa
注：使用 user1 身份，无法查看 cat /home/user1/a.txt
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
扩展：
[root@xuegod63 ~]# ll /etc/shadow* /etc/passwd*
-rw-r--r-- 1 root root 2819 7 月 2 21:06 /etc/passwd
-rw-r--r--. 1 root root 2778 7 月 2 21:03 /etc/passwd- #安装完系统时，会生成 passwd 备份文件。系统安装好后，新增加的用户，不会写到/etc/passwd-
---------- 1 root root 1544 7 月 2 21:06 /etc/shadow
---------- 1 root root 1412 7 月 2 21:05 /etc/shadow- #安装完系统时，会生成 shadow 备份文件。系统安装好后，新增加用户的密码信息，不会写到/etc/shadow-

7.1.7 修改权限

方法 1：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

例如下面的组合：

u-w	user	拥有者
g+x	group	组
o=r	other	其他人
a+x	all	所有人

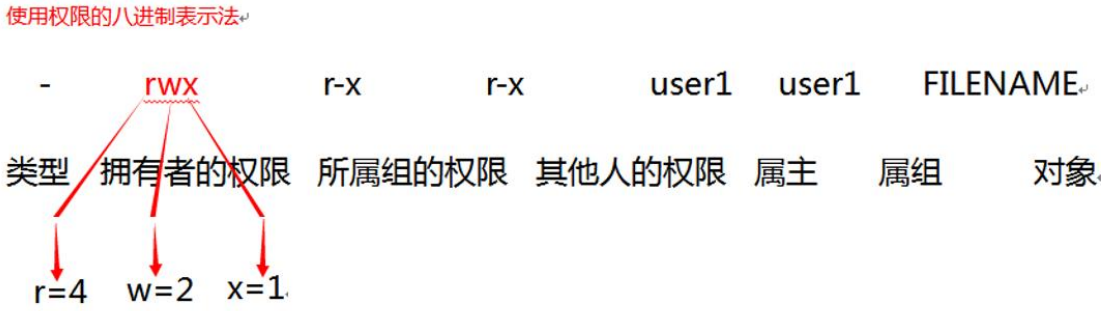
例：chmod 修改权限

```
[root@xuegod63 ~]# touch 1.txt
[root@xuegod63 ~]# ll 1.txt
-rw-r--r-- 1 root root 0 5 月     8 21:20 1.txt
```

7.1.8 使用八进制（0-7）数字表示权限法

权限	二进制值	八进制值	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	有读取和执行权限
rw-	110	6	有读取和写入权限
rwx	111	7	有全部权限

例 1：



例 1：

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=755

语法：

chmod 755 文件或文件夹名字

chmod a=rwx b.txt 等于 chmod 777 b.txt

例：

```
[root@xuegod63 ~]# touch dd.txt
[root@xuegod63 ~]# ll dd.txt
-rw-r--r-- 1 root root 0 5 月     8 21:40 dd.txt
[root@xuegod63 ~]# chmod 755 dd.txt
[root@xuegod63 ~]# ll dd.txt
```

-rwxr-xr-x 1 root root 0 5 月 8 21:40 dd.txt

[root@xuegod63 ~]# chmod 700 dd.txt

[root@xuegod63 ~]# ll dd.txt

-rwx------ 1 root root 0 5 月 8 21:40 dd.txt

7.1.9 权限对文件和目录的影响

有三种权限可以应用：读取，写入与执行，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	可以列出目录的内容（文件名）
w(写入)	可以更改文件的内容	可以创建或删除目录中的任意文件
x(执行)	可以作为命令执行文件	可以访问目录的内容（取决于目录中文件的权限）

扩展：补码

为什么我们创建的文件的权限默认是 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@xuegod63 ~]# vim /etc/profile

```
59 if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
60     umask 002
61 else
62     umask 022
63 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 -gn 显示组名，id -un 显示用户名

id -g 显示用户组 ID ，id -u 显示用户 id

[root@xuegod63 ~]# touch aaa

[root@xuegod63 ~]# ll aaa

-rw-r--r-- 1 root root 0 3 月 19 11:15 aaa 666-022=644

6 4 4

[root@xuegod63 ~]# su - user1

[user1@xuegod63 ~]$ touch aaa

[user1@xuegod63 ~]$ ll aaa

-rw-rw-r-- 1 user1 user1 0 3 月 19 11:17 aaa 666-002= 664

6 6 4

[user1@xuegod63 ~]$ mkdir dir2

[user1@xuegod63 ~]$ ll -d dir2

drwxrwxr-x 2 user1 user1 6 3 月 19 11:22 dir2 777-002=775

7 7 5

临时生效： umask 权限补码

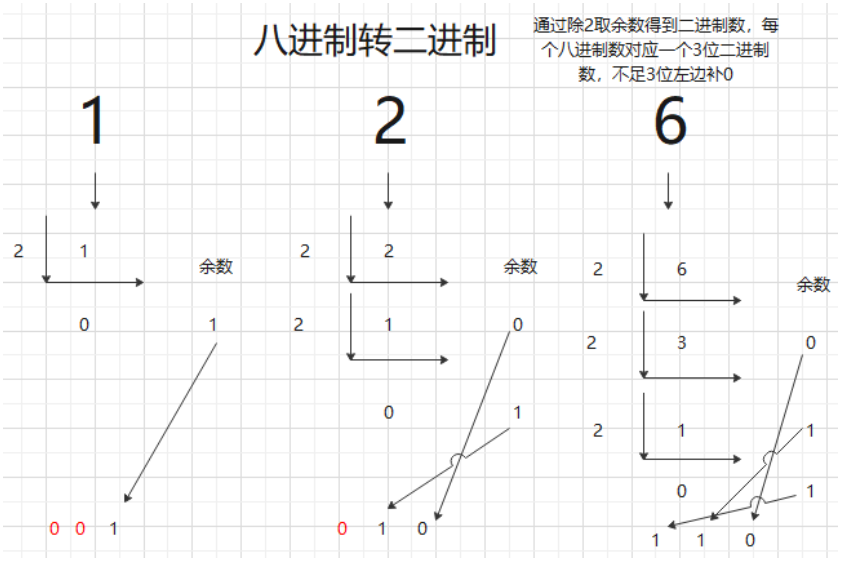
[user1@xuegod63 ~]$ umask 044

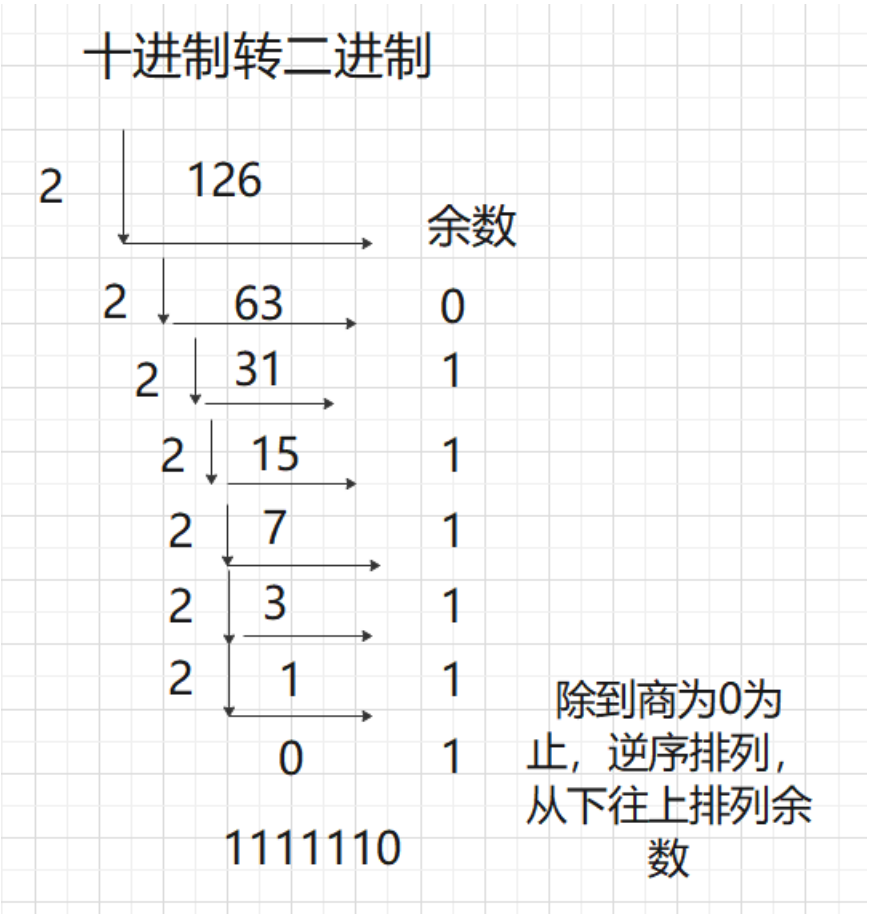
[user1@xuegod63 ~]$ touch ss.txt
[user1@xuegod63 ~]$ ll ss.txt
-rw--w--w- 1 user1 user1 0 5 月 8 21:47 ss.txt 666-044=622
6 2 2

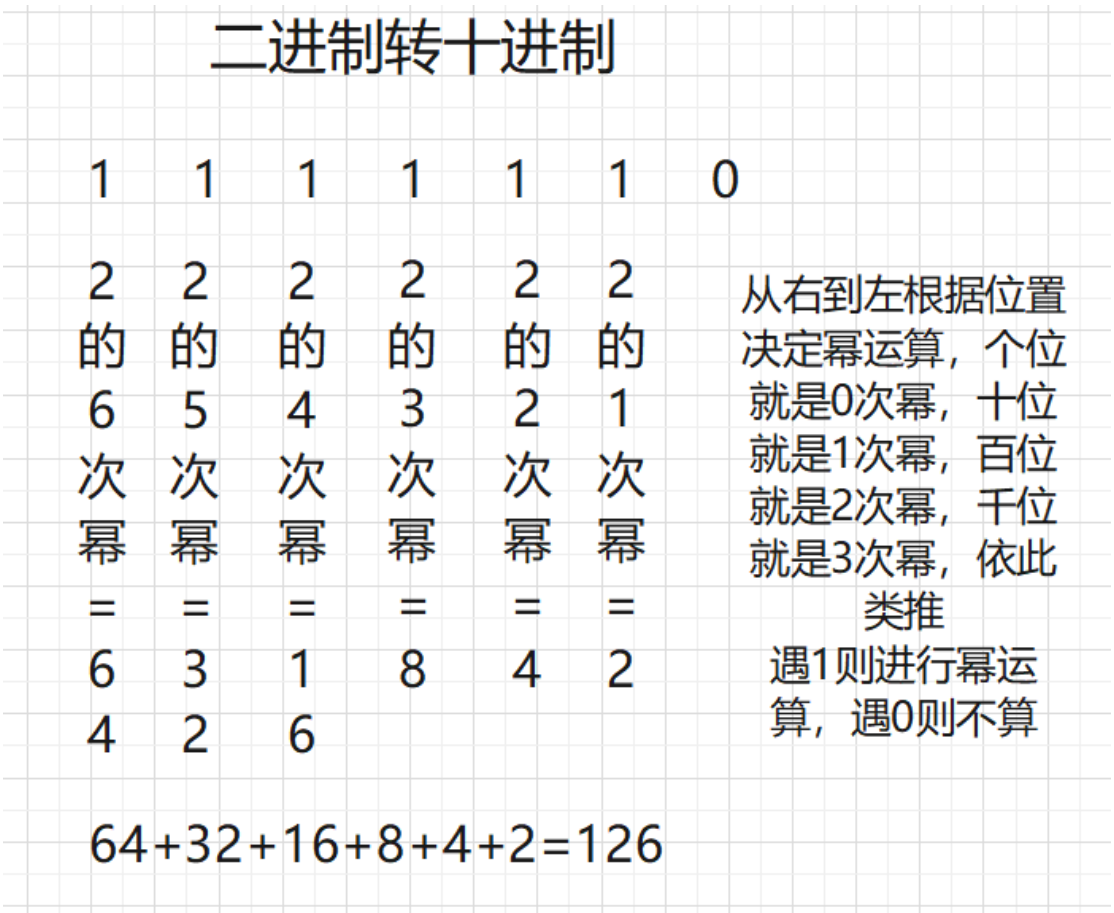
权限的算法：一般情况是：目录默认权限-umask 值
666-022=644
777-022=755
#这是一个好的记忆方法，但不严谨。

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）?
例：[root@xuegod63 ~]# umask 033
[root@xuegod63 ~]# touch k.txt
[root@xuegod63 ~]# ll k.txt
-rw-r--r-- 1 root root 0 5 月 8 22:00 k.txt
6 4 4
答：结果为： 644

权限科学的计算方法步骤：
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 # 默认权限和 umask 都转成二进制

111 101 101 # umask 取反的值

110 110 110 与 #第二步，默认权限二进制值和 umask 二进制值取反后做与运算

111 101 101 # umask 取反的值

110 100 100 #与运算，都为 1 的为 1，否则为 0

6 4 4 #转成 8 进制

例 2： umask 为 033 结果为： 644

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，默认权限二进制值和 umask 二进制值取反后做与运算

111 100 100 # umask 取反的值

110 100 100 #与运算，都为 1 的为 1，否则为 0

6 4 4 #转成 8 进制

https://tool.lu/hexconvert/ 在线进制转换器

7.2 文件的特殊权限：suid sgid sticky 和文件扩展权限 ACL

其实文件与目录设置不止这些，还有所谓的特殊权限。由于特殊权限会拥有一些“特权”.

7.2.1 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
4	2	1
chmod u+s	chmod g+s	chmod o+t
chmod 4664	chmod 2664	chmod 1664

suid 的代表数字是 4，比如 4755 的结果是 rwsr-xr-x

sgid 的代表数字是 2，比如 2755 的结果是 rwxr-sr-x

sticky 位代表数字是 1，比如 1755 的结果是 rwxr-xr-t

suid+sgid+sticky=7 ，比如 7755 的结果是 rwsr-sr-t

例：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。

```
[root@xuegod63 ~]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 6月  10 2014 /usr/bin/passwd
```

[root@xuegod63 ~]# ll /usr/bin/passwd #passwd 命令有 suid 权限

-rwsr-xr-x. 1 root root 27832 6 月 10 2014 /usr/bin/passwd

互动： 普通用户 user1，没有对 shadow 文件写入的权限， 但是 user1 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@xuegod63 ~]# ll /etc/shadow

----------. 1 root root 1179 9 月 19 2017 /etc/shadow

[root@xuegod63 ~]# passwd user1 #改 user1 密码为 123456

[root@xuegod63 ~]# tail -5 /etc/shadow |grep user1 #查看下密码

[root@xuegod63 ~]# su - user1

上一次登录：二 5 月 8 21:07:24 CST 2018pts/0 上

[user1@xuegod63 ~]$ passwd

更改用户 user1 的密码 。

为 user1 更改 STRESS 密码。

（当前）UNIX 密码：123456

新的 密码：xuegod123

重新输入新的 密码：xuegod123

passwd：所有的身份验证令牌已经成功更新。

[user1@xuegod63 ~]$ exit

[root@xuegod63 ~]# tail -5 /etc/shadow |grep user1 #查看 shadow 文件已经被 user1 用户修改成功。

因为 user1 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。

例 2：

[root@xuegod63 ~]# useradd user3

[root@xuegod63 ~]# passwd user3

[root@xuegod63 ~]# su - user3

[user1@xuegod63 ~]$ less /etc/shadow

less: 打不开 /etc/shadow: 权限不够

[root@xuegod63 ~]# su - root

[root@xuegod63 ~]# chmod u+s /usr/bin/less #切换到 root，给一个 suid 权限

[root@xuegod63 ~]# su - user3

[user3@xuegod63 ~]$ less /etc/shadow #看到了 shadow 内容，不要按 q 退出进程

新开 shell 窗口查看 u+s 后的效果：

[root@xuegod63 ~]# ps aux | egrep 'less|USER'| grep -v grep

```
[root@xuegod63 ~]# ps aux | egrep  'less|USER'| grep -v grep
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      63656  0.0  0.0 110308  1016 pts/1    S+   11:53   0:00 less /etc/shadow
```

[root@xuegod63 ~]# ll /usr/bin/less

-rwsr-xr-x 1 root root 154536 Sep 26 2011 /usr/bin/less

另外：

[root@xuegod63 ~]# chmod 4755 /usr/bin/less # 等同于 chmod u+s /usr/bin/less

[root@xuegod63 ~]# chmod u-s /usr/bin/less #去掉 less 命令的 suid 权限

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。

给二进制可执行程序设置，则运行程序时，临时获得该程序所属组权限

[root@xuegod63 ~]# echo aaaaa > file2.txt

[root@xuegod63 ~]# ll file2.txt #查看到 file2.txt 文件所属组为 root，且所属组有读权限

-rw-r--r-- 1 root root 6 3 月 19 12:46 file2.txt

[root@xuegod63 ~]# su - user1

[user1@xuegod63 ~]$ less /root/file2.txt #查看 file2.txt ，权限不足，已去掉了 suid 权限

[user1@xuegod63 ~]$ exit

[root@xuegod63 ~]# chmod u-s,g+s /usr/bin/less #去掉 less 的 suid 权限，增加 sgid 权限

[root@xuegod63 ~]# ll /usr/bin/less #查看 less 所属组为 root

-rwxr-sr-x. 1 root root 158240 7 月 31 2015 /usr/bin/less

[root@xuegod63 ~]# su - user1

[user1@xuegod63 ~]$ less /root/file2.txt #已经有权限了，不要按 q 退出进程，新开 shell 查看

[root@xuegod63 ~]# ps aux | egrep 'USER|less' | grep -v grep

```
[root@xuegod63 ~]# ps aux | egrep 'USER|less' | grep -v grep
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user1     64729  0.0  0.0 110308  1020 pts/2    S+   12:54   0:00 less /root/file2.txt
```

可以看到运行 less 的用户身份为 user1，可以读取 file2.txt ，是因为临时获取 less 程序的所属组身份，而 file2.txt 所属组与 less 程序所属组相同，且有读权限。

```
[root@xuegod63 ~]# ll /root/file2.txt
-rw-r--r-- 1 root root 6 3月  19 12:46 /root/file2.txt
```

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@xuegod63 ~]# mkdir test
[root@xuegod63 ~]# ll -d test
drwxr-xr-x 2 root root 4096 Jan 24 20:14 test
[root@xuegod63 ~]# chmod g+s test
[root@xuegod63 ~]# ll -d test
drwxr-sr-x 2 root root 4096 Jan 24 20:14 test
```

测试：sgid 效果

```
[root@xuegod63 ~]# chown :bin test/
[root@xuegod63 ~]# touch test/a.txt
[root@xuegod63 ~]# ll !$          #!$代表得是上一条命令中最后一个参数，快捷键是 esc 加.
ll test/a.txt
-rw-r--r-- 1 root bin 0 Jan 24 20:15 test/a.txt
```

Stickybit

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。

例： 系统中的 tmp 目录就是这样

```
[root@xuegod63 ~]# ll -d /tmp/
drwxrwxrwt. 11 root root 4096 Jan 24 19:41 /tmp/
```

用法：

```
chmod o+t /tmp
```

```
[root@xuegod63 ~]# mkdir /test
[root@xuegod63 ~]# ll -d /test
[root@xuegod63 ~]# chmod 1777 /test
[root@xuegod63 ~]# ll -d /test
```

```
[root@xuegod63 ~]# ll -d /test
drwxrwxrwt 2 root root 6 3月  19 13:38 /test
```

```
[root@xuegod63 ~]# su - user1
[user1@xuegod63 ~]$ echo user1 > /test/user1.txt
[user1@xuegod63 ~]$ chmod o+rw /test/user1.txt
[user1@xuegod63 ~]$ ll /test/user1.txt
[user1@xuegod63 ~]$ exit
```

```
[user1@xuegod63 ~]$ ll /test/user1.txt
-rw-rw-rw- 1 user1 user1 6 3月  19 13:48 /test/user1.txt
```

```
[root@xuegod63 ~]# su - user2
[user2@xuegod63 ~]$ echo user2 > /test/user1.txt
```

[user2@xuegod63 ~]$ cat /test/user1.txt　　#可以修改文件内容
[user2@xuegod63 ~]$ rm -f /test/user1.txt

```
[user2@xuegod63 ~]$ rm -f /test/user1.txt
rm: 无法删除"/test/user1.txt": 不允许的操作
```

[root@xuegod63 ~]# chmod o-t /test
[user2@xuegod63 ~]$ rm -f /test/user1.txt

其他用户无法删除，只有文件创建者、目录所有者才能删除

7.2.1　文件扩展权限 ACL

扩展 ACL　：access control list
[root@xuegod63 ~]# touch　/opt/a.txt
[root@xuegod63 ~]# ll /opt/a.txt
-rw-r--r-- 1 root root 0 7 月　 2 22:12 /opt/a.txt
设置只让用户 user1 对文件 a.txt 拥有的 rwx 权限 ，user1 不属于 a.txt 的所属主，和组，other 的权限还是 r--。怎么做?
[root@xuegod63 ~]# id user1
[root@xuegod63 ~]# getfacl　/opt/a.txt
getfacl: Removing leading '/' from absolute path names
file: opt/a.txt
owner: root
group: root
user::rw-
group::r--
other::r--
[root@xuegod63 ~]# setfacl -m u:user1:rwx /opt/a.txt　 # u ：　设置某个用户拥有的权限
[root@xuegod63 ~]# setfacl -m u:user2:rwx /opt/a.txt　 # u ：　设置某个用户拥有的权限
-m, --modify=acl 更改文件的访问控制列表

[root@xuegod63 ~]# getfacl　/opt/a.txt
getfacl: Removing leading '/' from absolute path names
...
user::rw-
user:user1:rwx

[root@xuegod63 ~]# su - user1
[user1@xuegod63 ~]$ vim /opt/a.txt　#发现 user1 虽然是 other，但是可以写 a.txt
[user1@xuegod63 ~]$ ll /opt/a.txt
-rw-rwxr--+ 1 root root 8 5 月　 8 22:42 /opt/a.txt　　#setfacl 权限会同步到组权限中
注：只要增加了扩展权限，这个地址就会有一个 + 加号

例 2：　给目录加扩展权限
[root@xuegod63 ~]# mkdir /opt/test

[root@xuegod63 ~]# setfacl -m d:u:user1:rwx　　/opt/test　　#d　默认在就是有这个权限

例 3：给目录下所有文件都加扩展权限

[root@xuegod63 ~]# setfacl -R -m u:user1:rw- /opt/test　　#-R 一定要在-m 前面，表示目录下所有文件

-R, --recursive 递归操作子目录

[root@xuegod63 ~]# setfacl -x u:user1 /opt/a.txt　　# 去掉单个权限

[root@xuegod63 ~]# getfacl　/opt/a.txt

[root@xuegod63 ~]# setfacl -b　/opt/a.txt　　# 去掉所有 acl 权限

7.3　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@xuegod63 ~]# touch hack.sh aa.sh

[root@xuegod63 ~]# 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@xuegod63 ~]# rm -rf aa.sh

黑客使用 xshell 悄悄执行在后台添加 attr 扩展属性：(这个别让学员看到^_^)

[root@xuegod63 ~]# chattr　+i hack.sh

删除文件:

[root@xuegod63 ~]# rm -rf　hack.sh　#发现删除不了

为什么删除不了?

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

解决:

[root@xuegod63 ~]# lsattr hack.sh

----i----------- hack.sh

[root@xuegod63 ~]# chattr -i hack.sh

[root@xuegod63 ~]# echo aa >> hack.sh

[root@xuegod63 ~]# lsattr hack.sh　#查看扩展属性

---------------- hack.sh

[root@xuegod63 ~]# chattr +a hack.sh

```
[root@xuegod63 ~]# rm -rf hack.sh
rm: 无法删除"hack.sh"：不允许的操作
[root@xuegod63 ~]# echo aaa >> hack.sh
```

总结：

7.1 文件的基本权限：r w x （UGO）

7.2 文件的特殊权限：suid sgid sticky 和文件扩展权限 ACL

7.3 实战：创建一个让 root 都无法删除的文件