第八章：权限管理

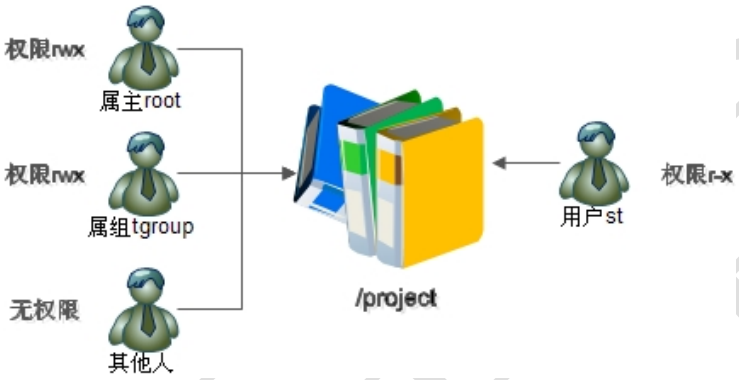
尚硅谷云计算 Linux 课程

版本：V1.0

讲师：沈超

一、ACL 权限

1、ACL 概述：　ACL 是用于解决用户对文件身份不足的问题的



2、开启 ACL

```
[root@localhost ~]# dumpe2fs -h /dev/sda3
#dumpe2fs 命令是查询指定分区详细文件系统信息的命令
选项：
    -h  仅显示超级块中信息，而不显示磁盘块组的详细信息。
...省略部分输出...
Default mount options:    user_xattr acl
...省略部分输出...
```

如果没有开启，手工开启分区的 ACL 权限：

```
[root@localhost ~]# mount -o remount,acl /
#重新挂载根分区，并挂载加入 acl 权限
```

也可以通过修改/etc/fstab 文件，永久开启 ACL 权限：

```
[root@localhost ~]# vi /etc/fstab
UUID=c2ca6f57-b15c-43ea-bca0-f239083d8bd2  /    ext4    defaults,acl        1  1
#加入 acl
[root@localhost ~]# mount -o remount /
#重新挂载文件系统或重启动系统，使修改生效
```

3、ACL 基本命令

getfacl　文件名　　查询文件的 ACL 权限

setfacl　选项　文件名　　设定 ACL 权限

-m 设定 ACL 权限
-b 删除 ACL 权限
-x:用户 删除单个用户的 ACL 权限

setfacl -m u:用户名:权限 文件名
setfacl -m g:组名：权限 文件名

setfacl -m u:aa:rwx /test 给 test 目录赋予 aa 是读写执行的 ACL 权限

setfacl -m u:cc:rx -R soft/ 赋予递归 ACL 权限，只能赋予目录
-R 递归

setfacl -m d:u:aa:rwx -R /test ACL 默认权限。 注意：默认权限只能赋予目录

注意：如果给目录赋予 acl 权限，两条命令都要输入
递归与默认的区别：
setfacl -m u:cc:rx -R soft/ 只对已经存在的文件生效
setfacl -m d:u:aa:rwx -R /test 只对以后新建的文件生效

4、 最大有效权限 mask

```
[root@localhost /]# setfacl -m m:rx project/
#设定 mask 权限为 r-x。使用“m:权限”格式
[root@localhost /]# getfacl project/
# file: project/
# owner: root
# group: tgroup
user::rwx
group::rwx                      #effective:r-x
mask::r-x
#mask 权限变为了 r-x
other::---
```

5、删除 ACL 权限

```
[root@localhost /]# setfacl -x u:st /project/
#删除指定用户和用户组的 ACL 权限
[root@localhost /]# setfacl -b project/
#会删除文件的所有的 ACL 权限
```

二、sudo 授权 给普通用户赋予部分管理员权限

/sbin/ 在此目录下命令只有超级用户才能使用

/usr/sbin/

1 root 身份：

visudo 赋予普通用户权限命令，命令执行后和 vi 一样使用

```
root    ALL=(ALL)       ALL
#用户名   被管理主机的地址=（可使用的身份）     授权命令（绝对路径）
# %wheel        ALL=(ALL)       ALL
#%组名   被管理主机的地址=（可使用的身份）     授权命令（绝对路径）
```

- ✧ 用户名/组名：代表 root 给哪个用户或用户组赋予命令，注意组名前加“%”
- ✧ 用户可以用指定的命令管理指定 IP 地址的服务器。如果写 ALL，代表可以管理任何主机，如果写固定 IP，代表用户可以管理指定的服务器。（这里真的很奇怪啊，超哥一直认为这里的 IP 地址管理的是登录者来源的 IP 地址，查了很多资料也都是这样的。直到有一天超哥查看“man 5 sudoers”帮助，才发现大家原来都理解错误了，这里的 IP 指定的是用户可以管理哪个 IP 地址的服务器。那么如果你是一台独立的服务器，这里写 ALL 和你服务器的 IP 地址，作用是一样的。而写入网段，只有对 NIS 服务这样用户和密码集中管理的服务器才有意义）。如果我们这里写本机的 IP 地址，不代表只允许本机的用户使用指定命令，而代表指定的用户可以从任何 IP 地址来管理当前服务器。
- ✧ 可使用身份：就是把来源用户切换成什么身份使用，（ALL）代表可以切换成任意身份。这个字段可以省略
- ✧ 授权命令：代表 root 把什么命令授权给普通用户。默认是 ALL，代表任何命令，这个当然不行。如果需要给那个命令授权，写入命令名即可，不过需要注意一定要命令写成绝对路径

2、举例

1）

举个例子，比如授权用户 user1 可以重启服务器，则由 root 用户添加如下行：

```
[root@localhost ~]# visudo
user1       ALL= /sbin/shutdown  -r now
[user1@localhost ~]$ sudo -l
#查看可用的授权
```

2)

再举个例子，授权一个用户管理你的 Web 服务器，不用自己插手是不是很爽，以后修改设置更新网页什么都不用管，一定 Happy 死了，Look：

首先要分析授权用户管理 Apache 至少要实现哪些基本授权：

1、可以使用 Apache 管理脚本

2、可以修改 Apache 配置文件

3、可以更新网页内容

假设 Aapche 管理脚本程序为/etc/rc.d/init.d/httpd 。

为满足条件一，用 visudo 进行授权：

```
[root@localhost ~]# visudo
user1       192.168.0.156=/etc/rc.d/init.d/httpd reload,\
/etc/rc.d/init.d/httpd configtest
```

授权用户 user1 可以连接 192.168.0.156 上的 Apache 服务器，通过 Apache 管理脚本重新读取配置文件让更改的设置生效（reload）和可以检测 Apache 配置文件语法错误（configtest），但不允许其执行关闭（stop）、重启（restart）等操作。（“\”的意思是一行没有完成，下面的内容和上一行是同一行。）

为满足条件二，同样使用 visudo 授权：

```
[root@localhost ~]# visudo
user1        192.168.0.156=/binvi /etc/httpd/conf/httpd.conf
```

授权用户 user1 可以用 root 身份使用 vi 编辑 Apache 配置文件。

以上两种 sudo 的设置，要特别注意，很多朋友使用 sudo 会犯两个错误：第一，授权命令没有细化到选项和参数；第二，认为只能授权管理员执行的命令。

条件三则比较简单，假设网页存放目录为/var/www/html ，则只需要授权 user1 对此目录具有写权限或者索性更改目录所有者为 user1 即可。如果需要，还可以设置 user1 可以通过 FTP 等文件共享服务更新网页。

3）授权 aa 用户可以添加其他普通用户

aa ALL=/usr/sbin/useradd 赋予 aa 添加用户权限.命令必须写入绝对路径

aa ALL=/usr/bin/passwd 赋予改密码权限，取消对 root 的密码修改

aa ALL=/usr/bin/passwd [A-Za-z]*, !/usr/bin/passwd ″″, !/usr/bin/passwd root

aa 身份

sudo /usr/sbin/useradd ee 普通用户使用 sudo 命令执行超级用户命令

三、文件特殊权限 SetUID、SetGID、Sticky BIT

1、SetUID

1）SetUID 是什么

SetUID 的功能可以这样理解：

- ✧ 只有可以执行的二进制程序才能设定 SUID 权限
- ✧ 命令执行者要对该程序拥有 x（执行）权限
- ✧ 命令执行者在执行该程序时获得该程序文件属主的身份（在执行程序的过程中灵魂附体为文件的属主）
- ✧ SetUID 权限只在该程序执行过程中有效，也就是说身份改变只在程序执行过程中有效

2）举例

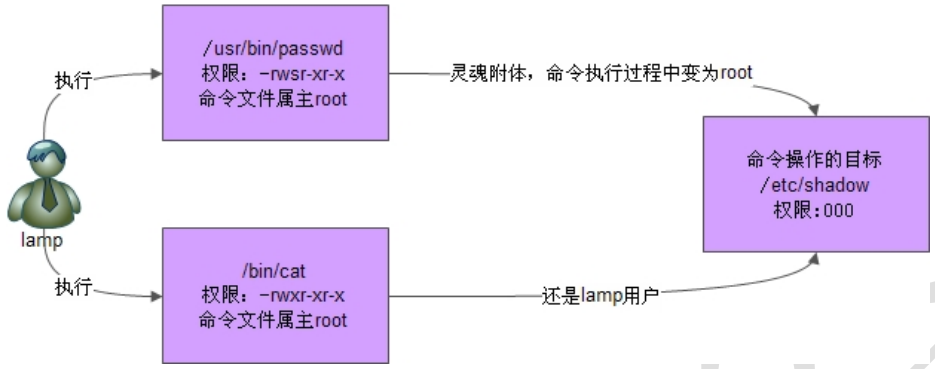
```
[root@localhost ~]# ll /etc/passwd
-rw-r--r-- 1 root root 1728 1 月  19 04:20 /etc/passwd
[root@localhost ~]# ll /etc/shadow
---------- 1 root root 1373 1 月  19 04:21 /etc/shadow
```

因为

```
[root@localhost ~]# ll /usr/bin/passwd
-rwsr-xr-x 1 root root 25980 2 月  22 2012 /usr/bin/passwd
```

/usr/bin/passwd 命令拥有特殊权限 SetUID ，也就是在属主的权限位的执行权限上是 s。可以这样来理解它：当一个具有执行权限的文件设置 SetUID 权限后，用户执行这个文件时将以文件所有者的

身份执行。/usr/bin/passwd 命令具有 SetUID 权限，所有者为 root（Linux 中的命令默认所有者都是 root），也就是说当普通用户使用 passwd 更改自己密码的时候，那一瞬间突然灵魂附体了，实际是在用 passwd 命令所有者 root 的身份在执行 passwd 命令，root 当然可以将密码写入/etc/shadow 文件（不要忘记 root 这个家伙是 superman 什么事都可以干），所以普通用户也可以修改/etc/shadow 文件，命令执行完成后该身份也随之消失



如果取消 SetUID 权限，则普通用户就不能修改自己的密码了

3）危险的 SetUID

```
[root@localhost ~]# chmod u+s /usr/bin/vim
[root@localhost ~]# ll /usr/bin/vim
-rwsr-xr-x 1 root root 1847752 4 月    5 2012 /usr/bin/vim
```

4）有几点建议：

- ✧ 关键目录应严格控制写权限。比如“/”、“/usr”等；
- ✧ 用户的密码设置要严格遵守密码三原则；
- ✧ 对系统中默认应该具有 SetUID 权限的文件作一列表，定时检查有没有这之外的文件被设置了 SetUID 权限。

5）检测 SetUID 的脚本

```
[root@localhost ~]# vi suidcheck.sh
#!/bin/bash
# Author: shenchao （E-mail: shenchao@atguigu.com）

find / -perm -4000 -o -perm -2000 > /tmp/setuid.check
#搜索系统中所有拥有 SUID 和 SGID 的文件，并保存到临时目录中。
for i in $(cat /tmp/setuid.check)
#做循环，每次循环取出临时文件中的文件名
do
        grep $i /root/suid.list > /dev/null
        #比对这个文件名是否在模板文件中
                if [ "$?" != "0" ]
                #如果在，不报错
                then
                        echo "$i isn't in listfile! " >> /root/suid_log_$(date +%F)
```

```
                    #如果文件名不再模板文件中，则报错。并把报错报错到日志中
                fi
done
rm -rf /tmp/setuid.check
#删除临时文件

[root@localhost ~]# chmod u+s /bin/vi
#手工给 vi 加入 SUID 权限
[root@localhost ~]# ./suidcheck.sh
#执行检测脚本
[root@localhost ~]# cat suid_log_2013-01-20
/bin/vi isn't in listfile!
#报错了，vi 不再模板文件中。代表 vi 被修改了 SUID 权限
```

2、SetGID

1）针对文件的作用

SGID 即可以针对文件生效，也可以针对目录生效，这和 SUID 明显不同。如果针对文件，SGID 的含义如下：

- ✧ 只有可执行的二进制程序才能设置 SGID 权限
- ✧ 命令执行者要对该程序拥有 x（执行）权限
- ✧ 命令执行在执行程序的时候，组身份升级为该程序文件的属组
- ✧ SetGID 权限同样只在该程序执行过程中有效，也就是说组身份改变只在程序执行过程中有效

```
[root@localhost ~]# ll /var/lib/mlocate/mlocate.db
-rw-r----- 1 root slocate 1838850 1 月  20 04:29 /var/lib/mlocate/mlocate.db
```

大家发现属主权限是 r、w，属组权限是 r，但是其他人权限是 0：

```
[root@localhost ~]# ll /usr/bin/locate
-rwx--s--x 1 root slocate 35612 8 月  24 2010 /usr/bin/locate
```

当普通用户 user1 执行 locate 命令时，会发生如下事情：

- ✧ /usr/bin/locate 是可执行二进制程序，可以赋予 SGID
- ✧ 执行用户 user1 对/usr/bin/locate 命令拥有执行权限
- ✧ 执行/usr/bin/locate 命令时，组身份会升级为 slocate 组，而 slocate 组对/var/lib/mlocate/mlocate.db 数据库拥有 r 权限，所以普通用户可以使用 locate 命令查询 mlocate.db 数据库
- ✧ 命令结束，user1 用户的组身份返回为 user1 组

2）针对目录的作用

如果 SGID 针对目录设置，含义如下：

- ✧ 普通用户必须对此目录拥有 r 和 x 权限，才能进入此目录
- ✧ 普通用户在此目录中的有效组会变成此目录的属组
- ✧ 若普通用户对此目录拥有 w 权限时，新建的文件的默认属组是这个目录的属组

这样写的实在太难看明白了，举个例子：

```
[root@localhost ~]# cd /tmp/
```

```
#进入临时目录做此实验。因为临时目录才允许普通用户修改
[root@localhost tmp]# mkdir dtest
#建立测试目录
[root@localhost tmp]# chmod g+s dtest
#给测试目录赋予 SGID
[root@localhost tmp]# ll -d dtest/
drwxr-sr-x 2 root root 4096 1 月  20 06:04 dtest/
#SGID 已经生效
[root@localhost tmp]# chmod 777 dtest/
#给测试目录权限，让普通用户可以写
[root@localhost tmp]# su - user1
#切换成普通用户 user1
[user1@localhost ~]$ cd /tmp/dtest/
#普通用户进入测试目录
[user1@localhost dtest]$ touch abc
#普通用户建立 abc 文件
[user1@localhost dtest]$ ll
总用量 0
-rw-rw-r-- 1 user1 root 0 1 月  20 06:07 abc
#abc 文件的默认属组不再是 user1 用户组，而变成了 dtest 组的属组 root
```

3、文件特殊权限之 Sticky BIT

Sticky BIT 粘着位，也简称为 SBIT。SBIT 目前仅针对目录有效，它的作用如下：

- ✧ 粘着位目前只对目录有效
- ✧ 普通用户对该目录拥有 w 和 x 权限，即普通用户可以在此目录拥有写入权限
- ✧ 如果没有粘着位，因为普通用户拥有 w 权限，所以可以删除此目录下所有文件，包括其他用户建立的文件。一但赋予了粘着位，除了 root 可以删除所有文件，普通用户就算拥有 w 权限，也只能删除自己建立的文件，但是不能删除其他用户建立的文件。

4、设定文件特殊权限

特殊权限这样来表示：

- ✧ 4 代表 SUID
- ✧ 2 代表 SGID
- ✧ 1 代表 SBIT

```
[root@localhost ~]# chmod 4755 ftest
#赋予 SUID 权限
[root@localhost ~]# chmod 2755 ftest
#赋予 SGID 权限
[root@localhost ~]# mkdir dtest
[root@localhost ~]# chmod 1755 dtest/
#SBIT 只对目录有效，所以建立测试目录，并赋予 SBIT
```

四、文件系统属性 chattr 权限

1、命令格式

```
[root@localhost ~]# chattr [+-=] [选项] 文件或目录名
选项：
    +:  增加权限
    -:  删除权限
    =:  等于某权限
    i:  如果对文件设置 i 属性，那么不允许对文件进行删除、改名，也不能添加和修改数
        据；如果对目录设置 i 属性，那么只能修改目录下文件的数据，但不允许建立和删
        除文件。
    a:  如果对文件设置 a 属性，那么只能在文件中增加数据，但是不能删除也不能修改数
    据；如果对目录设置 a 属性，那么只允许在目录中建立和修改文件，但是不允许删        除
    e:  Linux 中绝大多数的文件都默认拥有 e 属性。表示该文件是使用 ext 文件系统进行
        存储的，而且不能使用“chattr -e”命令取消 e 属性。
```

2、查看文件系统属性 lsattr

```
[root@localhost ~]# lsattr 选项 文件名
选项：
    -a  显示所有文件和目录
    -d  若目标是目录，仅列出目录本身的属性，而不是子文件的
```

3、举例

```
例 1：
#给文件赋予 i 属性
[root@localhost ~]# touch ftest
#建立测试文件
[root@localhost ~]# chattr +i ftest
[root@localhost ~]# rm -rf ftest
rm: 无法删除"ftest": 不允许的操作
#赋予 i 属性后，root 也不能删除
[root@localhost ~]# echo 111 >> ftest
-bash: ftest: 权限不够
#也不能修改文件的数据

#给目录赋予 i 属性
[root@localhost ~]# mkdir dtest
#建立测试目录
[root@localhost dtest]# touch dtest/abc
#再建立一个测试文件 abc
[root@localhost ~]# chattr +i dtest/
#给目录赋予 i 属性
[root@localhost ~]# cd dtest/
[root@localhost dtest]# touch bcd
```

```
touch: 无法创建"bcd": 权限不够
#dtest 目录不能新建文件
[root@localhost dtest]# echo 11 >> abc
[root@localhost dtest]# cat abc
11
#但是可以修改文件内容
[root@localhost dtest]# rm -rf abc
rm: 无法删除"abc": 权限不够
#不能删除
```

```
例 2：
[root@localhost ~]# mkdir -p /back/log
#建立备份目录
[root@localhost ~]# chattr +a /back/log/
#赋予 a 属性
[root@localhost ~]# cp /var/log/messages /back/log/
#可以复制文件和新建文件到指定目录
[root@localhost ~]# rm -rf /back/log/messages
rm: 无法删除"/back/log/messages": 不允许的操作
#但是不允许删除
```