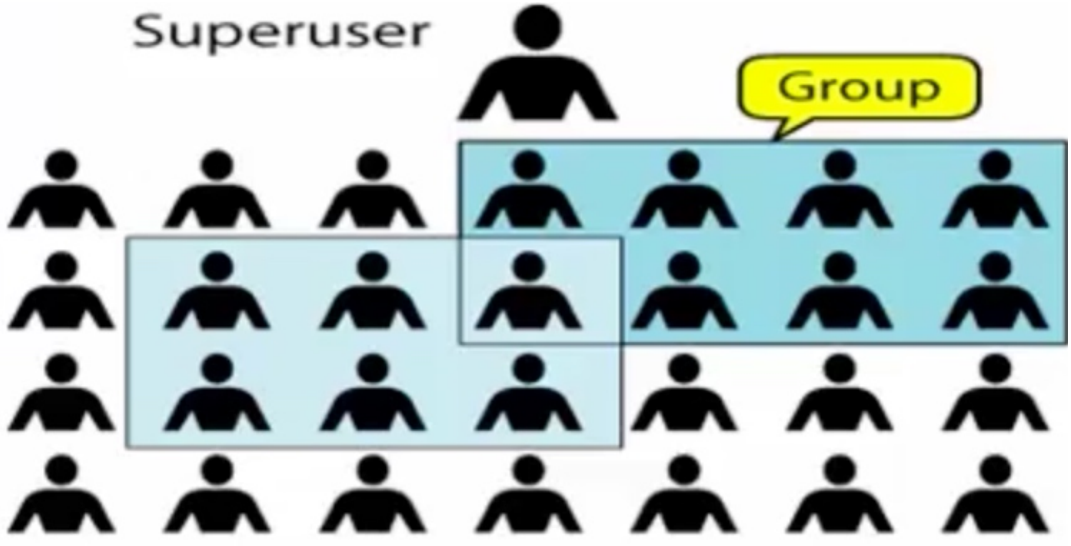
day02-用户-提权-权限管理

2.1 linux用户管理

2.1.1 用户基本概述

2.1.1.1 什么是用户

用户指的是能够正常登录 `Linux` 或 `Windows` 系统，比如：登陆QQ的用户、登陆荣耀的用户、等等



2.1.1.2 为什么需要用户

- 1.系统上的每一个进程(运行的程序)，都需要一个特定的用户运行；
- 2.通常在公司是使用普通用户管理服务器，因为root权限过大，容易造成故障；

2.1.1.3 用户有哪些分类

- 系统对用户有一个约定？(约定娶你，就真的会娶嘛？tz)

用户UID	系统中约定的含义
0	超级管理员，最高权限，有着极强的破坏能力
1~200	系统用户，用来运行系统自带的进程，默认已创建
201~999	系统用户，用来运行用户安装的程序，所以此类用户无需登录系统
1000+	普通用户，正常可以登陆系统的用户，权限比较小，能执行的任务有限

2.1.1.4 查询用户ID信息

- 使用 `id` 命令查询当前登录用户的信息

```
[root@web ~]# id    #查看当前所登陆的用户信息
uid=0(root) gid=0(root) groups=0(root)
[root@web ~]# id xu #查看其它用户的信息
uid=1000(xu) gid=1000(xu) groups=1000(xu)
```

2.1.2 用户相关配置文件

- 当我们创建一个新的用户，系统会将用户的信息存放在 `/etc/passwd` 中，而密码单独存储在 `/etc/shadow` 中也就是说这两个文件非常的重要，不要轻易删除与修改。

2.1.2.1 passwd文件

- `/etc/passwd` 配置文件解释如下图，或者使用命令 `man 5 passwd` 获取帮助

[root@bgx ~]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash 以:作为分隔符，总共七列

root 用户名称	x 密码占位符	0 用户UID	0 组GID	root 注释信息	/root 用户家目录	/bin/bash 登陆shell
第一列	第二列	第三列	第四列	第五列	第六列	第七列

2.1.2.2 shadow文件

- `/etc/shadow` 配置文件解释如下图，或者使用命令 `man 5 shadow` 获取帮助
- PS: 使用change修改密码过期时间示例

[root@bgx ~]# head -1 /etc/shadow
bgx:!!:16312:0:99999:7:2:66275: 以:作为分隔符，总共九列

第一列:bgx， 用户名称
第二列:!!， 密码为一长串字符，!!则表示无密码
第三列:16312，最近一次变更密码，从1970年到现在，过了多少天
第四列:0， 密码最少使用天数，0无限制
第五列:99999，密码最长使用天数，默认999999不过期
第六列:7， 密码到期前，系统会在密码到期前7天提醒变更密码
第七列:2， 密码到期后，密码过期后2天强制提示变更用户密码
第八列:66275，账户失效时间，从1970年起，账户在这个日期前可使用，到期后失效

2.1.3 用户相关命令

2.1.3.1 添加用户useradd

若想要添加 `Linux` 系统普通用户，可以使用 `useradd` 命令，使用 `root` 账号登录 `Linux` 系统之后就可以添加系统普通用户了。

选项	功能描述

-u	指定要创建用户的UID,不允许冲突
-g	指定要创建用户基本组
-G	指定要创建用户附加组,逗号隔开可添加多个附加组
-d	指定要创建用户家目录
-s	指定要创建用户的bash shell
-c	指定要创建用户注释信息
-M	给创建的用户不创建家目录
-r	创建系统账户，默认无家目录

2.1.1 添加用户示例1

- 创建 `oldxu` 用户
  - 用户ID为 `6969`
  - 基本组为 `ops`，附加组 `dev`
  - 注释信息 `2000 new student`，登陆 `shell:/bin/bash`

```
[root@web ~]# groupadd ops
[root@web ~]# groupadd dev
[root@web ~]# useradd -u 5001 -g ops -G dev -c "2000 student" -s /bin/bash oldxu
```

2.1.2 添加用户示例2

- 创建一个 `mysql` 系统用户
  - 该用户不需要家目录
  - 该用户不需要登陆系统

```
[root@web ~]# useradd -r dba -M -s /sbin/nologin
```

2.2 修改用户usermod

若想要修改 `Linux` 系统普通用户，可以使用 `usermod` 命令，使用 `root` 账号登录 `Linux` 系统之后就可以修改系统普通用户了。

选项	功能描述

-u	指定修改用户的UID
-g	指定要修改用户基本组
-G	指定要修改用户附加组，使用逗号隔开多个附加组, 覆盖原有的附加组
-d	指定要修改用户家目录
-s	指定要修改用户的bash shell
-c	指定要修改用户注释信息
-l	指定要修改用户的登陆名
-L	指定要锁定的用户
-U	指定要解锁的用户

2.2.1 修改用户示例1

- 修改 `oldxu` 用户
  - `uid` 为 `5008`,
  - 基本组为 `network`，附加组为 `ops,dev,sa`
  - 注释信息为 `student`，登陆名称为 `new_oldxu`

```
[root@web ~]# groupadd network
[root@web ~]# usermod oldxu -c "student" -g network -aG sa -l new_oldxu
```

2.2.1 修改用户示例2

- 修改 `new_oldxu` 用户
  - 为 `new_oldxu` 配置密码
  - 锁定该用户，然后测试远程连接登陆
  - 解锁该用户然后再次测试远程连接登陆

```
# 锁定用户
[root@web ~]# echo "123" |passwd --stdin new_oldxu
[root@web ~]# usermod -L new_oldxu

# 解锁用户
[root@web ~]# usermod -U new_oldxu
```

2.3 删除用户userdel

若想要删除 `Linux` 系统普通用户，可以使用 `userdel` 命令，使用 `root` 账号登录 `Linux` 系统之后就可以删除系统普通用户了

2.2.1 删除用户示例1

- 删除 `new_oldxu` 用户
  - 连同家目录一起删除

```
[root@web ~]# userdel -r new_oldxu
```

2.2.1 删除用户示例2

- 批量系统中此前创建过的所有无用的用户
  - 使用 `awk` 提取无用的用户名称
  - 使用 `sed` 拼接删除用户的命令
  - 调用 `userdel` 命令，连同家目录一起全部删除

```
[root@web ~]# awk -F ':' '$3>1000{print $1}' /etc/passwd |sed -r 's#(.*)#userdel -r
 \1#g'|bash
```

2.4 设定密码passwd

- 创建用户后，如需要使用该用户进行远程登陆系统则需要为用户设定密码，设定密码使用 `passwd`
  - 1.普通用户只允许变更自己的密码，无法修改其他人密码，并且密码长度必须8位字符
  - 2.管理员用户允许修改任何人的密码，无论密码长度多长或多短。
- 推荐密码保存套件工具，支持 `windows、MacOS、Iphone` 以及浏览器插件 Lastpass官方网站

2.4.1 交互设定密码

- 通过交互方式为用户设定密码

```
[root@web ~]# passwd         #给当前用户修改密码
[root@web ~]# passwd root    #给root用户修改密码
[root@web ~]# passwd oldxu #给oldboy用户修改密码，普通用户只能自己修改自己
```

2.4.2 非交互设定密码

- 非交互式设定简单密码

```
[root@web ~]# echo "123" | passwd --stdin xuliangwei
```

- 非交互式设定随机密码

```
[root@web ~]# yum install -y expect
[root@web ~]# echo $(mkpasswd -l 10 -d 2 -c 2 -C 2 -s 4) |tee pass.txt| passwd --st
din oldxu
```

2.2 linux用户组管理

2.2.1 什么是用户组

- 组是一种逻辑层面的定义
- 逻辑上将多个用户归纳至一个组，当我们对组操作，其实就相当于对组中的所有用户进行操作。

2.2.2 组有几种类别

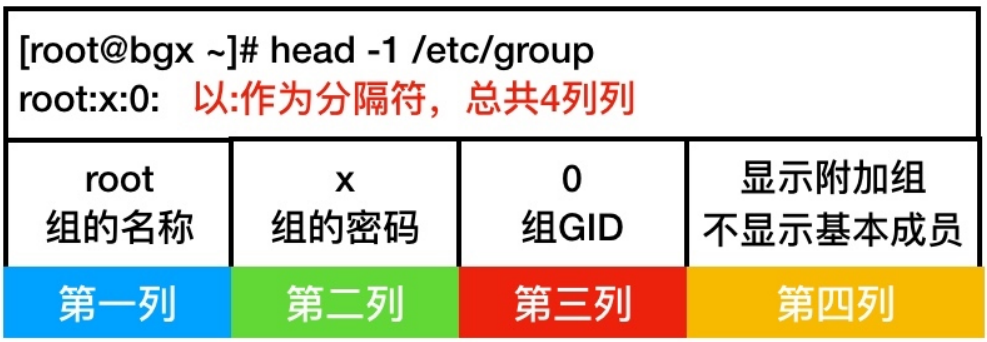
- 对于用户来说，组分为如下几类
  - 默认组：创建用户时不指定组，则默认创建与用户同名的组；
  - 基本组：用户有且只能有一个基本组，创建时可通过-g指定（亲爹）；
  - 附加组：用户可以有多个附加组，创建时通过-G指定（干爹）；

2.2.3 组相关配置文件

- 组账户信息保存在 `/etc/group` 和 `/etc/gshadow` 两个文件中，重点关注 `group`

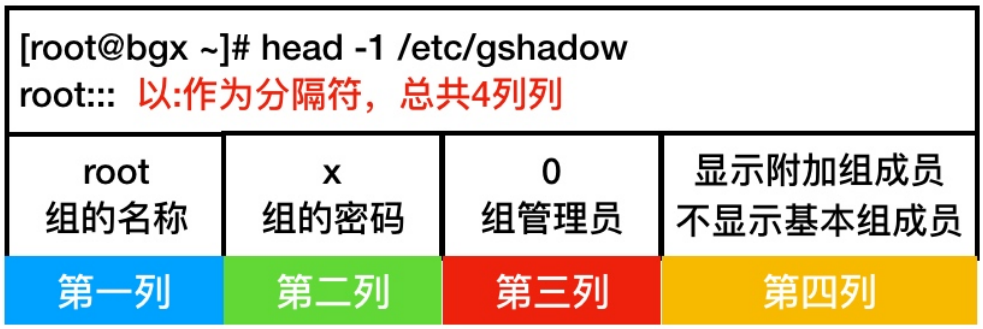
2.2.3.1 group文件

- `/etc/group` 配置文件解释如下图



2.2.3.2 gshadow文件

- `/etc/gshadow` 配置文件解释如下图



2.2.4 用户组相关命令

2.2.4.1 添加组groupadd

若想要添加 `Linux` 用户组，可以使用 `groupadd` 命令，使用 `root` 账号登录 Linux 系统之后就可以添加用户组了。

选项	功能描述
-f	如果组已经存在，会提示成功创建的状态
-g	为新组设置 GID，若 GID 已经存在会提示 GID 已经存在
-r	创建一个系统组

2.2.4.1.1 添加组示例1

- 添加一个 `salary` 的组
  - 为组设定 `gid` 为 `10000`

```
[root@web ~]# groupadd salary -g 10000
[root@web ~]# tail  -1 /etc/group
salary:x:10000:
```

2.2.4.1.2 添加组示例2

- 添加一个 `salary_2` 的组
  - 添加为系统组

```
[root@web ~]# groupadd -r salary_2
[root@web ~]# tail -1 /etc/group
salary_2:x:988:
```

2.2.4.2 修改组groupmod

若想要修改 `Linux` 用户组，可以使用 `groupmod` 命令，使用 `root` 账号登录 Linux 系统之后就可以修改用户组了。

选项	功能描述
-f	如果组已经存在，会提示成功创建的状态
-g	为新组设置 GID，若 GID 已经存在会提示 GID 已经存在
-r	创建一个系统组
-n	改名为新的组

2.2.4.2.1 修改组示例1

- 修改 `salary` 用户组组名为 `system`

```
[root@web ~]# groupmod -n system salary
[root@web ~]# tail -1 /etc/group
system:x:10000:
```

2.2.4.2.2 修改组示例2

- 修改 `system` 用户组 `GID` 为 `5000`

```
[root@web ~]# groupmod system -g 5000
[root@web ~]# tail -1 /etc/group
system:x:5000:
```

2.2.4.3 删除组groupadd

若想要修改 `Linux` 用户组，可以使用 `groupdel` 命令，使用 `root` 账号登录 Linux 系统之后就可以修改用户组了。

2.2.4.3.1 删除组示例1

- 删除 `salary_2` 系统用户组

```
[root@web ~]# groupdel salary_2
```

2.2.4.3.2 删除组示例2

- 创建 `tom` 用户，设置主组为 `system`，然后测试删除 `system` 组

```
[root@web ~]# useradd tom -g system
[root@web ~]# groupdel system
groupdel: cannot remove the primary group of user 'tom'

# 如果组中存在用户是无法删除该组，必先删除用户后在删除组
[root@web ~]# userdel -r tom
[root@web ~]# groupdel system
```

2.2.4.4 用户与用户组场景

- 1.创建 `dev` 与 `ops` 两个组；
- 2.创建 `bob` 用户，设定基本为 `dev`，密码为 `123`；
- 3.创建 `alice` 用户，设定基本为 `ops`，密码为 `123`；
- 4.创建 `/opt/reosurce` 文件，然后修改属组为 `ops`、权限为 `664`
- 5.测试发现 `alice` 用户可以读写，而 `bob` 用户仅可以查看；
- 6.现在希望 `bob` 也能够对文件进行读写，如何快速实现（为 `bob` 添加 `ops` 附加组）；

1.创建组与用户；

```
[root@web ~]# groupadd dev
[root@web ~]# groupadd ops
[root@web ~]# useradd bob -g dev
[root@web ~]# useradd alice -g ops
```

2.为用户设定登陆密码；

```
[root@web ~]# echo "123" | passwd --stdin bob
[root@web ~]# echo "123" | passwd --stdin alice
```

3.建立文件，然后分配好权限（可先不理解，照着敲）；

```
[root@web ~]# echo "data" > /opt/resource
```

```
[root@web ~]# chgrp ops /opt/resource
[root@web ~]# chmod 664 /opt/resource
```

4.使用 `ops` 组的 `alice` 用户测试读和写权限，没有任何问题；

```
[alice@web~]$ echo "alice-data" >> /opt/resource
[alice@web ~]$ cat /opt/resource
data
alice-data
```

5.使用 `dev` 组的 `bob` 用户测试读和写权限，发现只有读权限，没有写权限；

```
[bob@web ~]$ echo "bob-data" >> /opt/resource
-bash: /opt/resource: 权限不够
[bob@web ~]$ cat /opt/resource
data
alice-data
```

6.为 `bob` 用户添加 `ops` 附加组，这样 `bob` 用户就能借助 `ops` 组权限，实现读写操作；

```
[root@web ~]# usermod bob -G ops
[root@web ~]# id bob
uid=1002(bob) gid=2020(dev) groups=2020(dev),2021(ops)
```

7.再次测试，发现 `bob` 用户能借助 `ops` 组实现读和写操作；

```
[bob@web ~]$ echo "bob-data" >> /opt/resource
[bob@web ~]$ cat /opt/resource
data
alice-data
bob-data
```

2.3 普通用户如何提权

往往公司的服务器对外都是禁止 `root` 用户直接登录，所以我们通常使用的都是普通用户，那么问题来了？当我们使用普通用户执行 `/sbin` 目录下部门命令时，会发现没有权限，这种情况会造成无法正常管理服务器，那如何才能不使用 `root` 用户直接登录系统，同时又保证普通用户能完成日常工作呢?

- 我们可以使用如下两种方式：`su、sudo`
  - 1. `su switch user`身份切换，使用普通用户登录，然后使用 `su` 命令切换到 `root`
    - 优点：简单
    - 缺点：需要知道 `root` 密码
  - 2. `sudo`提权，当需要使用 `root` 权限时进行提权，而无需切换至 `root`用户
    - 优点：安全、方便、
    - 缺点：需要预先定义规则、较为复杂

2.3.1 su命令身份切换

- 在使用 `su` 切换身份前，我们需要 `shell` 登陆分类、环境变量配置文件加载顺序；

2.3.1.1 Shell登陆分类

- 登陆`shell` 需要输入用户名和密码才能进入 `Shell` 日常接触的最多的一种
- 非登陆`shell` 不需要输入用户和密码就能进入 `Shell` ，比如运行 `bash` 会开启一个新的会话窗口

2.3.1.2 环境变量配置文件

- `profile`类文件：设定环境变量，登陆前运行的脚本和命令
- `bashrc` 类文件：设定本地变量，定义命令别名
- 用户配置文件：
  - `~/.bash_profile`
  - `~/.bashrc`
- 全局环境变量：
  - `/etc/profile`
  - `/etc/profile.d/*.sh`
  - `/etc/bashrc`
- 登录式 `shell` 配置文件加载顺序：`/etc/profile->/etc/profile.d/*.sh->~/.bash_profile->~/.bashrc->/etc/bashrc`
- 非登录式 `shell` 配置文件加载顺序：`/.bashrc->/etc/bashrc->/etc/profile.d/*.sh`

2.3.1.3 su与环境变量的关系

- `su - username` 属于登陆式 `Shell`
- `su username` 属于非登陆式 `Shell`
- 它们最大的区别就在于加载的环境变量不一样；

1.普通用使用 `su` 切换到 `root` 用户，需要输入 `root` 超级管理员密码；

```
[oldxu@web ~]$ su - root
密码:
[root@node1 ~]# pwd
/root
```

2.以某个用户的身份执行某个服务，使用命令 `su -c username`

```
[root@web ~]# su - oldxu -c 'ifconfig'
[root@web ~]# su - oldxu -c 'ls ~'
```

2.3.2 sudo命令提权

2.3.2.1 sudo的由来

`su` 命令在用户身份切换时，需要拿到 `root` 管理员密码；在多人协作时，如果当中某个用户不小心泄露了 `root` 密码；那系统会变得非常不安全，为了改进这个问题，从而就有了 `sudo`；

其实 `sudo` 就是给某个普通用户埋下了 `浩克hulk` 的种子，当需要执行一些特权操作时，进行发怒，获取最高权限，但正常情况下还是普通用户，任然会受到系统的约束以及限制；

2.3.2.2 sudo快速起步

- 快速配置 `sudo` 方式[先睹为快]

1.将用户加入 `wheel` 组，默认 `wheel` 组有 `sudo` 权限；

```
[root@node1 ~]# usermod oldxu -G wheel
```

2.切换到普通用户身份；

```
[root@web ~]# su - oldxu
```

3.普通用户正常情况下无法删除 `/opt` 目录；

```
[oldxu@web ~]$ rm -rf /opt/
rm: cannot remove `/opt: Permission denied
```

4.使用 `sudo` 提权，然后输入普通用户密码，会发现能正常删除无权限的 `/opt` 目录；

```
[oldxu@web ~]$ sudo rm -rf /opt
```

5.后期可以通过审计日志查看普通用户提权都执行了什么操作；

```
[root@web ~]# tail -f /var/log/secure
```

2.4 linux基础权限

2.4.1 linux权限介绍

2.4.1.1 什么是权限

- 权限是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。

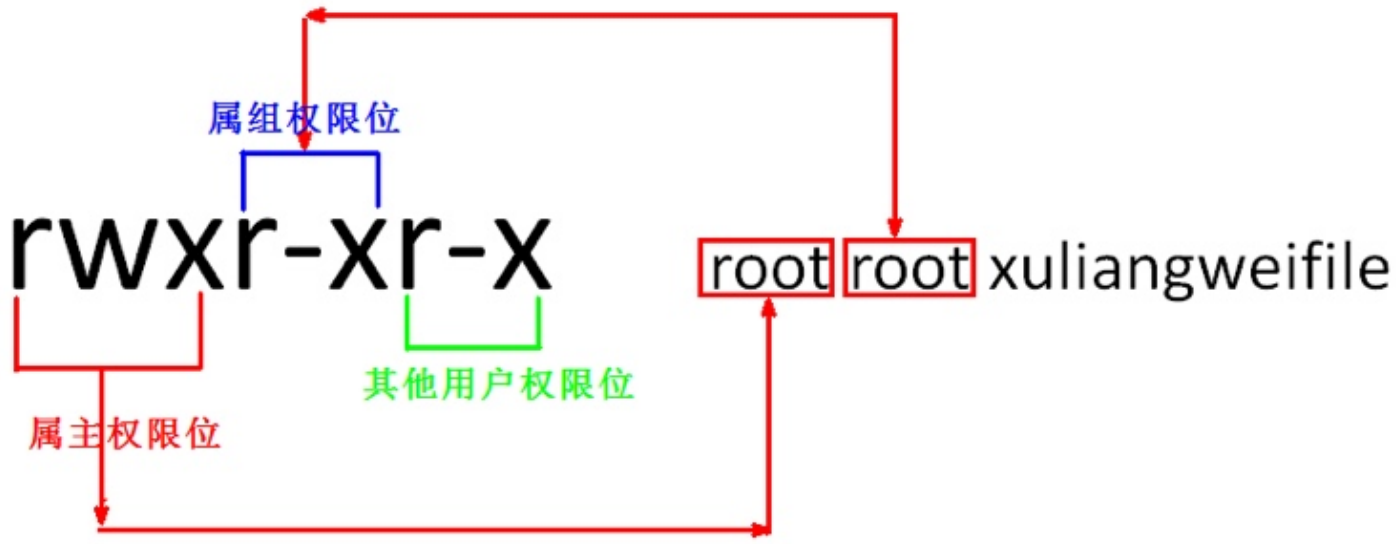
2.4.1.2 为什么需要权限

- `Linux` 是一个多用户系统，对于每一个用户来说，个人隐私的保护十分重要，所以需要进行权限划分；
  - 1.安全性：降低误删除风险、减少人为造成故障以及数据泄露等风险；
  - 2.数据隔离：不同的权限能看到、以及操作不同的数据（比如员工薪资表）；
  - 3.职责明确：电商场景客服只能查看投诉、无法查阅店铺收益，运营则能看到投诉以及店铺收益；

2.4.1.3 权限与用户的关系

- 在 `Linux` 系统中，权限是用来定义用户能做什么，不能做什么。
  - 1.针对文件定义了三种身份，分别是属主 `owner` 、属组 `group` 、其他人 `others`

- 2.每种身份又对应三种权限，分别是读`read`、写`write`、执行`execute`



- 当一个用户访问文件流程如下
  - 1) 判断用户是否为文件属主，如果是则按属主权限进行访问
  - 2) 判断用户是否为文件属组，如果是则按属组权限进行访问
  - 3) 如果不是文件属主、也不是该文件属组，则按其他人权限进行访问

2.4.1.4 权限中rwx的含义

`linux`中权限是由，`rwxr-xr-x`这9位字符来表示，主要控制文件的属主`User`、属组`Group`、其他用户`Other`

字母	含义	二进制	八进制权限表示法
r--	读取权限	100	4
-w-	写入权限	010	2
--x	执行权限	001	1
---	没有权限	000	0

- 文件权限示例1：`rwxrw-r-- alice hr file1.txt`
  - `Q1`：`alice`对`file1.txt`文件拥有什么权限?
  - `Q2`：`jack`对`file1.txt`文件有什么权限? 前提：`jack`附加组为`hr`组
  - `Q3`：`tom`对`file1.txt`文件有什么权限?
- 文件权限示例2：`rw-r----- root dev file2.txt`
  - `Q1`：`root`对`file2.txt`文件拥有什么权限?
  - `Q2`：`jack`对`file2.txt`文件有什么权限? 前提：`jack`附加组为`dev`组
  - `Q3`：`alice`对`file2.txt`文件有什么权限?

- 文件权限示例3：`rwxr--rwx jack ops file3.txt`
  - `Q1`：`jack` 对 `file3.txt` 文件拥有什么权限?
  - `Q2`：`tom` 对 `file3.txt` 文件有什么权限? 前提：`ops` 附加组为 `dev` 组
  - `Q3`：`alice` 对 `file2.txt` 文件有什么权限?

2.4.2 修改文件权限

2.4.2.1 修改权限的意义

- 简单来说就是：`赋于某个用户或组` --> `能够以何种方式（读写执行）` --> `访问文件`

2.4.2.2 如何修改权限

- 修改权限使用 `chmod（change mode）` 命令来实现
  - 对于 `root` 用户而言，可以修改任何人的文件权限；
  - 对于普通用户仅仅只能变更属于自己的文件权限；

2.4.2.2.1 UGO方式

- 给文件所有人添加读写执行权限

```
[root@web ~]# chmod a=rwx file
```

- 取消文件的所有权限

```
[root@web ~]# chmod a=-rwx file
```

- 属主读写执行，属组读写，其他人无权限

```
[root@web ~]# chmod u=rwx,g=rw,o=- file
```

- 属主属组读写执行，其他人读权限

```
[root@web ~]# chmod ug=rwx,o=r file
```

2.4.2.2.1 NUM方式

- 设定文件权限 `644`， `rw-r--r--`

```
[root@web ~]# chmod 644 file
```

- 设定文件权限 `600`， `rw-------`

```
[root@web ~]# chmod 600 file
```

- 设定目录权限为755，递归授权 `rwx-r-xr-x`

```
[root@web ~]# chmod -R 755 dir
```

2.4.2.3 权限设定案例

- 场景1：针对 `hr` 部门的访问目录 `/data/hr` 设置权限，要求如下：
  - 1.超级管理员 `root` 用户和 `hr` 组的员工可以读、写、执行。
  - 2.其他用户或者组没有任何权限。

```
[root@web ~]# groupadd hr
[root@web ~]# useradd hr01 -G hr
[root@web ~]# useradd hr02 -G hr
[root@web ~]# mkdir /home/hr
[root@web ~]# chgrp hr /home/hr
[root@web ~]# chmod 770 /home/hr
```

在 `Linux` 中权限设定对文件和对目录的影响是有区别的。

权限	对文件的影响	对目录的影响
读取权限（r）	具有读取\阅读文件内容权限	具有浏览目录及子目录
写入权限（w）	具有新增、修改文件内容的权限	具有增加和删除目录内的文件
执行权限（x）	具有执行文件的权限	具有访问目录的内容(取决于目录中文件权限)

2.4.3 权限对文件的影响

2.4.3.1 验证r权限

- 使用 `root` 身份，新建文件
  - 切换普通用户
  - 测试普通用户对该文件是否拥有可读权限
  - 测试普通用户对该文件是否拥有执行和删除权限

```
[root@web ~]# echo "date" > /opt/file
[root@web ~]# ll filename
-rw-r--r-- 1 root root 5 Jan 24 08:24 filename

# 切换普通身份
[root@web ~]# su - oldxu

# 查看
[oldxu@web ~]$ cat /opt/filename
date

# 删除
[oldxu@ansible-hostname ~]$ rm -f /opt/file
rm: cannot remove '/opt/file': Permission denied
```

2.4.3.2 验证w权限

- 修改权限只有 `w`
  - 测试能否查看文件
  - 测试能否写入数据至文件
  - 测试能否删除文件

```
# 修订权限
[root@web ~]# chmod 642 /opt/file

# 查看
[oldxu@ansible-hostname ~]$ cat /opt/file
cat: /opt/file: Permission denied

# 写入
[oldxu@ansible-hostname ~]$ vim /opt/file   # 权限不足
[oldxu@ansible-hostname ~]$ echo "date" >> /opt/file

# 删除
[oldxu@ansible-hostname ~]$ rm -f /opt/file
rm: cannot remove '/opt/file': Permission denied
```

2.4.3.3 验证x权限

- 修改权限只有 `x`
  - 测试能否查看文件
  - 测试能否写入数据至文件
  - 测试能否删除文件
  - 测试能否读取文件

```
# 修订权限
[root@web ~]# chmod 641 /opt/file

# 查看
[oldxu@ansible-hostname ~]$ cat /opt/file
cat: /opt/file: Permission denied

# 写入
[oldxu@ansible-hostname ~]$ vim /opt/file   # 权限不足
[oldxu@ansible-hostname ~]$ echo "date" >> /opt/file

# 删除
[oldxu@ansible-hostname ~]$ rm -f /opt/file
rm: cannot remove '/opt/file': Permission denied

# 执行（因为没有读，所以无法执行）
[oldxu@ansible-hostname ~]$ /opt/file
-bash: /opt/file: 权限不够
```

2.4.3.4 文件权限总结

- 1.读取权限 `r` ： 具有读取、阅读文件内容权限
  - 只能使用查看类命令 `cat、head、tail、less、more`
- 2.写入权限 `w` ： 具有新增、修改文件内容的权限
  - 2.1）使用 `vim` 会提示权限拒绝，但可强制保存，会覆盖文件的所有内容；
  - 2.2）使用 `echo` 命令重定向的方式可以往文件内写入数据， `>>` 可以追加内容
  - 2.3）使用 `rm` 无法删除文件，因为删除文件需要看上级目录是否有 `w` 的权限
- 3.执行权限 `x`： 具有执行文件的权限
  - 3.1）执行权限什么用都没有
  - 3.2）如果普通用户需要执行文件，需要配合 `r` 权限

2.4.4 权限对目录的影响

2.4.4.1 验证r权限

- 使用 `root` 身份，新建目录，修订权限为 `774`

    - 在目录中创建一个普通文件
    - 测试是否能查看目录中内容
    - 测试能否能进入该目录

```
[root@web ~]# mkdir /data
[root@web ~]# echo "123" > /data/file
[root@web ~]# chmod 774 /data/

# 测试查看目录内容
[oldxu@ansible-hostname ~]$ ls /data/
ls: cannot access /data/file: Permission denied
file

# 测试进入目录
[oldxu@ansible-hostname ~]$ cd /data/
-bash: cd: /data/: 权限不够
```

2.4.4.2 验证w权限

- 使用 `root` 身份，修订权限为 `772`
    - 测试是否能查看目录中内容
    - 测试是否能删除目录中文件

```
# 修订权限
[root@web ~]# chmod 772 /data/

# 测试查看
[oldxu@ansible-hostname ~]$ ls /data/
ls: cannot open directory /data/: Permission denied

# 测试进入
[oldxu@ansible-hostname ~]$ cd /data/
-bash: cd: /data/: 权限不够

# 测试删除
[oldxu@ansible-hostname ~]$ rm -f /data/file
rm: cannot remove '/data/file': Permission denied
```

2.4.4.3 验证x权限

- 使用 `root` 身份，修订权限为 `771`
    - 测试是否能查看目录中内容
    - 测试能否进入目录中

```
# 修订权限
[root@web ~]# chmod 771 /data/

# 测试查看
[oldxu@ansible-hostname ~]$ ls /data/
ls: cannot open directory /data/: Permission denied

# 测试进入
[oldxu@ansible-hostname ~]$ cd /data/
[oldxu@ansible-hostname data]$
```

2.4.4.4 目录权限小结

- 1.读取权限 `r` ： 具有浏览目录及子目录权限
  - 1.1）使用 `ls` 命令浏览目录及子目录， 但同时也会提示权限拒绝
  - 1.2）使用 `ls -l` 命令浏览目录及子目录，文件属性会带问号，并且只能看到文件名
- 2.写入权限 `w` ： 具有增加、删除或修改目录内文件名权限，需要 `x` 权限配合
  - 2.1）可以在目录内创建文件, 删除文件(跟文件本身权限无关)
  - 2.2）不能进入目录、不能复制目录、不能删除目录、不能移动目录
- 3.执行权限 `x` ： 具有执行文件的权限
  - 3.1）只能进入目录
  - 3.2）不能浏览、复制、移动、删除

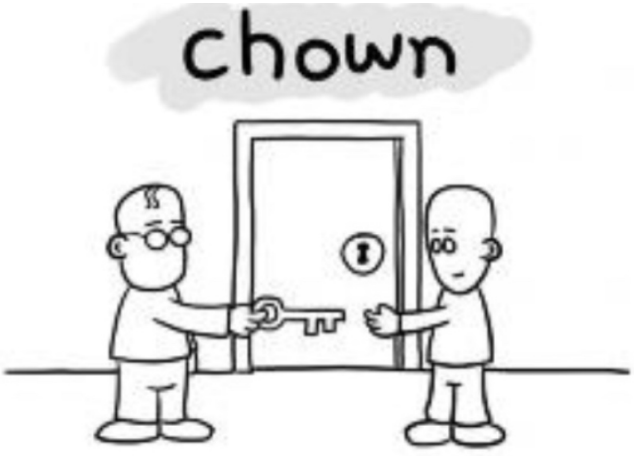
2.4.5 文件与目录权限总结

- 文件权限设定小结：
  - 文件 `r` 权限，只给用户查看,无其他操作；
  - 文件 `rw` 权限，可以查看和编辑文件内容；
  - 文件 `rx` 权限，允许查看和执行文件、但不能修改文件；
  - 文件 `rwx` 权限，能读、能写、能执行、不能删除；
- 目录权限设定小结：
  - 目录 `rx` 权限，允许浏览目录内文件以及子目录, 不允许在该目录下创建文件、删除文件
  - 目录 `rw` 权限，能查看目录，能往目录写入文件，但无法进入目录-->（使用的情况太少）
- 默认系统设定的安全权限：
  - 文件权限 `644`
  - 目录权限 `755`

2.4.6 修改文件所属关系

2.4.6.1 修改文件所属关系的意义

- 修改文件所属关系的意义是什么?
  - 假设：`alice` 用户现在有很多房产，希望将其中某一套出售给 `jack` 用户变现：
    - 1.通过 `root` 用户变更属主关系，将房产默认属主身份 `alice` 修改为 `jack`；
    - 2.修改完成后该房产拥有人则为 `jack` 用户，而不在是 `alice` 用户；
    - 3.此时房产属主为 `jack` 用户，而不在是 `alice` 用户；



2.4.6.2 如何修改文件的所属关系

- 可以使用 `chown（change owner）、chgrp（change group）` 命令实现。
  - `chown` 能变更文件的属主和属组；
  - `chgrp` 仅能变更文件的属组；

2.4.6.1.1 chown（change owner）

1.准备环境，创建文件和目录

```
[root@web ~]# mkdir /data
```

2.修改所属主为 `bin`

```
[root@web ~]# chown bin /data
```

3.修改所属组为 `adm`

```
[root@web ~]# chown .adm /data
```

4.修改目录所属主为 `root`，所属组为 `root`，并进行递归授权

```
[root@web ~]# chown -R root.root dir
```

2.4.6.1.2 chgrp（change group）

1.准备环境，创建文件和目录

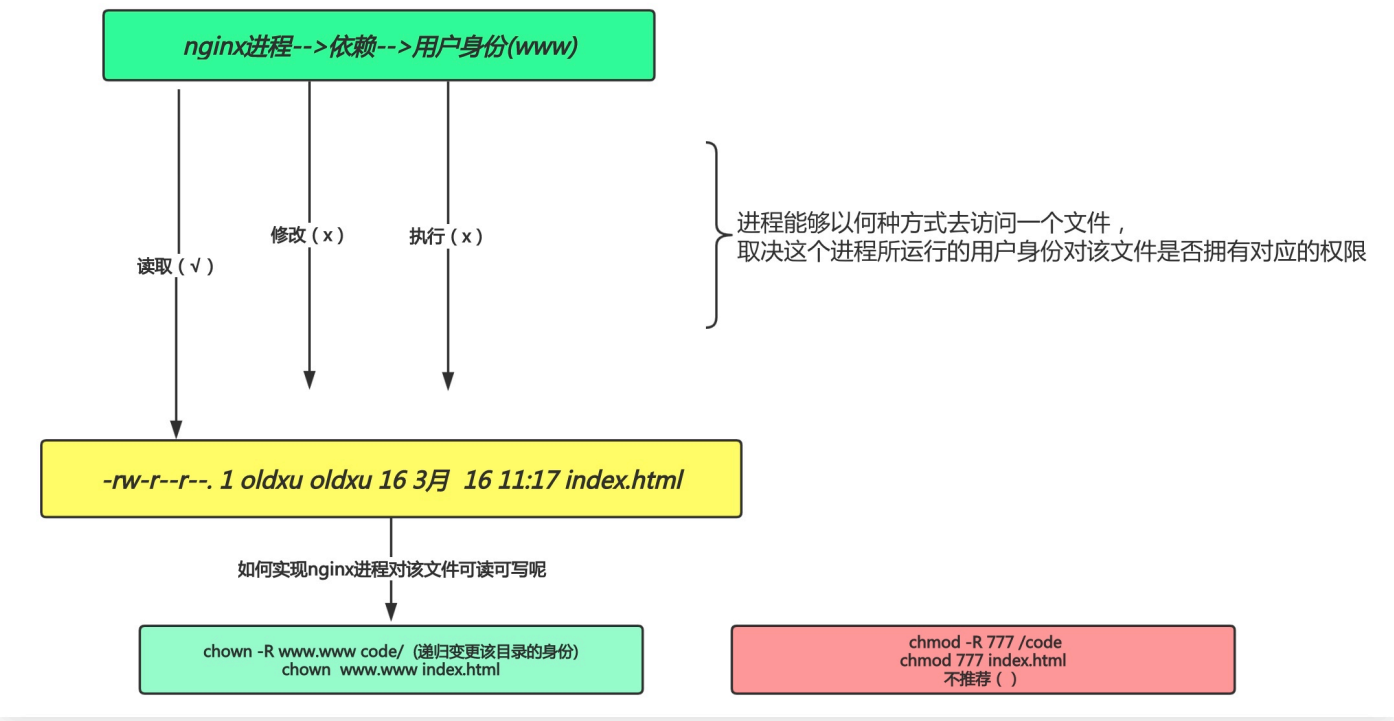
```
[root@web ~]# mkdir /data2
```

2.修改所属组为 `adm`

```
[root@web ~]# chgrp adm /data
```

2.4.7 修改文件所属关系场景

2.4.7.1 基于Httpd场景说明



2.4.7.2 基于Httpd场景实践

```
# 步骤如下
```

2.5 文件特殊权限

- 此前我们已经学习过 `r、w、 x` 这三种权限，但在查询系统文件时会发现有一些其他权限的字母；
  - 比如：`/usr/bin/passwd` 文件，属主应该是 `x` 的权限位出现了 `s`，
  - 比如：`/usr/bin/locate` 文件，属组应该是 `x` 的权限位出现了 `s`；
  - 比如：`/tmp` 目录，其他人应该是 `x` 的权限位出现了 `t`；
- 我们把这种称为特殊权限，那么特殊权限有什么作用呢？或者说能干啥？

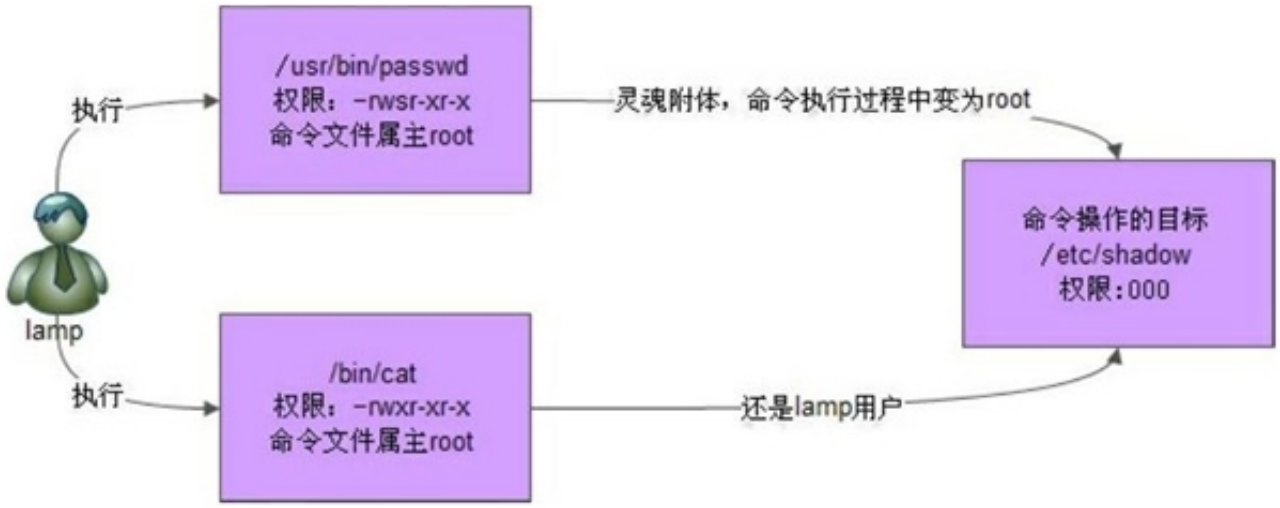
2.5.1 特殊权限SUID

2.5.1.1 SUID产生背景

在 `Linux` 系统中，每个普通用户都可以更改自己的密码，这是合理的设置；但是用户的密码信息存储在 `/etc/shadow` 文件中，也就是说，普通用户在更改自己密码时会更新 `/etc/shadow` 文件的内容。
但 `/etc/shadow` 文件不允许任何人修改？那为什么普通用户可以修改自己的权限呢？

```
[root@web ~]# ll /etc/shadow
---------- 1 root root 11409 Apr 13 03:26 /etc/shadow
```

其实，普通用户可以修改自己的密码在于 `passwd` 命令本身，该命令拥有特殊权限 `SetUID` 也就是在属主的权限位的执行权限上是 `s` 那如何理解特殊权限 `SetUID`：当一个执行文件设置 `SetUID` 后，用户在执行这个文件时将以文件所有者的身份来执行。



- 当我们使用普通用户 `oldxu` 执行 `passwd` 命令会发生什么变化呢？
  - 1.由于 `passwd` 命令拥有 `suid` 特殊权限；(在命令属主权限位有一个s)
  - 2.所以 `passwd` 命令在执行的过程中，会以命令的属主身份运行该命令；（也是root身

份）
    - 3.总结：`oldxu` --> `passwd` --> 转换为命令属主身份 `root` 执行 --> 操作 `/etc/shadow` 信息变更；

2.5.1.2 SUID配置语法

```
[root@web ~]# chmod u+s /usr/bin/cat
[root@web ~]# chmod 4755 /usr/bin/cat
```

2.5.1.3 SUID作用总结

- 1.让普通用户对可执行的二进制文件，临时拥有二进制文件的所属主权限；
- 2.如果设置的二进制文件没有执行权限，那么 `suid` 的权限显示就是大 `S`；
- 3.特殊权限 `suid` 仅对二进制可执行程序有效，其他文件或目录则无效；
- 注意: `suid` 相对危险，不建议对 `vim` 或 `rm` 进行 `suid` 设定操作；

2.5.2 特殊权限SGID

2.5.2.1 什么是SGID

- `SGID`：
    - 设置二进制可执行文件，命令在执行的过程中，会以命令的属组身份运行该命令
    - 设置在目录上，这时候在该目录下新建的文件/目录自动继承父目录的属组

2.5.2.2 SGID配置语法

```
[root@web ~]# chmod g+s /dir
[root@web ~]# chmod 2755 /dir
```

2.5.2.3 SGID场景说明

- 需求描述
    - 系统有两个用户，分别为 `ex1` 与 `ex2`，这两个用户都拥有 `example` 附加组；
    - 1.这两个用户需要共同拥有 `/data/code` 目录的开发权；
    - 2.互相之间能修改彼此的文件，且该目录不允许其他人进入查阅；

```
[root@web ~]# groupadd example
[root@web ~]# useradd ex1 -G example
[root@web ~]# useradd ex2 -G example
```

```
[root@web ~]# mkdir /data/code
[root@web ~]# chgrp project /data/code/
[root@web ~]# chmod 770 /data/code/
[root@web ~]# chmod 2770 /data/code/
```

2.5.3 特殊权限SBIT

2.5.3.1 什么是SBIT

一旦目录被赋予了粘滞位 `Sticky(SI TI KI)` 除了 `root` 可以删除目录中的所有文件，普通用户对该目录就算拥有 `w` 权限，也只能删除自己建立的文件，而不能删除其他用户建立的文件。

2.5.3.2 SBIT配置示例

需求：默认情况下 `/mnt` 不是粘滞位，如何将此目录设置为粘滞位；

```
[root@web ~]# chmod 1755  /tmp
[root@web ~]# chmod o+t /tmp
```

2.5.3.3 SBIT使用场景

后期当我们要初始化 `MySQL` 服务时，服务会创建一些临时文件存储至 `/tmp` 目录下，当初始化完毕后，自己会清理掉里面的数据，别人无法清理。（如果这个目录不是粘滞位，那么初始化MySQL就会报错）

- 编写Shell脚本模拟此场景
  - 1.模拟 `MySQL` 初始化创建文件至 `/tmp` 目录；
  - 2.然后登陆普通用户删除 `MySQL` 的初始化文件；
  - 3.如果普通用户删除成功，则初始化失败 （因为 `MySQL` 服务创建的文件，需要自行销毁）
  - 4.如果普通用户删除失败，则 `MySQL` 服务尝试删除，删除成功，则初始化成功；
- `Shell` 脚本如下

```
# 脚本如下
[root@web ~]# cat myql_init.sh
#!/urs/bin/bash

mysql_tmp_file=/tmp/mysql.init
User=oldxu
```

```
# 1.初始化MySQL服务
touch ${mysql_tmp_file}

# 2.模拟用户删除文件
useradd oldxu
su - ${User} -c "rm -f ${mysql_tmp_file} &>/dev/null"

# 3.检查是否删除成功
if [ $? -eq 0 ];then
    echo "${mysql_tmp_file} 文件被 ${User} 用户删除成功，该目录不是sbit，mysql初始化失败"
else
    echo "${mysql_tmp_file} 文件被 ${User} 用户删除失败，该目录是sbit，mysql初始化成功"
fi
```

结果测试与验证

```
# 默认粘滞位测试
[root@web ~]# sh myql_init.sh
/tmp/mysql.init 文件被 oldxu 用户删除失败，该目录是sbit，mysql初始化成功

# 修改为普通目录测试
[root@web ~]# chmod 777 /tmp/
[root@web ~]# sh myql_init.sh
/tmp/mysql.init 文件被 oldxu 用户删除成功，该目录不是sbit，mysql初始化失败
```

2.5.3.4 SBIT作用总结

- 1.让所有普通用户对该目录具有写入权限，并且能实现每个用户只能删自己的文件；
- 2.粘滞位目录表现在 `others` 的 `x` 位，用 `t` 表示，如果没有执行权限则显示为 `T`；
- 3.粘滞位目录的属主以及 `root` 用户有权限删除目录中的内容，其他用户无权限删除；

2.6 文件特殊属性

2.6.1 什么是特殊属性

这类文件属性凌驾于 `rwx` 基础权限之上，是一种高级属性。

2.6.2 特殊属性的作用

- 1）创建一个文件，不允许被修改、移动、删除，包括 `root` 也不行-->适合 `/etc/passwd`；

- 2）创建一个文件，仅允许往文件里面追加数据，不允许修改、移动、删除。-->适合 `sudo` 审计日志；

2.6.3 特殊属性如何配置

- `Linux` 系统通过 `chattr` 来实现特殊属性的配置
- 命令格式：`chattr [+-=] [选项] 文件或目录名`
  - `a`：可对文件进行追加内容；
  - `i`：锁定文件，不允许其他操作；

1.配置 `/etc/passwd` 文件，不能改，不能追加，不能删除。

```
#1.赋予i权限
[root@web ~]# chattr  +i /etc/passwd

#2.验证权限
[root@web ~]# rm -f /etc/passwd
rm: cannot remove '/etc/passwd': Operation not permitted
```

2.配置 `/var/log/secure` 文件，只能追加写入日志，不允许手动修改，也不允许删除。

```
#1.赋予 a 权限
[root@web ~]# chattr +a /var/log/secure
[root@web ~]# lsattr /var/log/secure
-----a---------- /var/log/secure

#2.测试追加数据
[root@web ~]# echo "test" >> /var/log/secure
[root@web ~]# echo "test" >> /var/log/secure

#3.不能删除，不能修改
[root@oldboy tmp]# rm -f /var/log/secure
rm: cannot remove '/var/log/secure': Operation not permitted
```

3.如果想取消特殊属性，需要使用 `root` 身份。

```
[root@web ~]# chattr -i /etc/passwd
[root@web ~]# chattr -a /var/log/secure
```

2.6.4 特殊属性场景示例

- 模拟病毒串改站点，然后使用 `chattr` 锁住文件，让病毒程序无法串改，然后追踪并杀死病毒程序；
    - 1.安装并启动 `http` 服务；
    - 2.模拟病毒脚本篡改网页内容；
    - 3.锁定篡改文件，然后找出病毒，将其杀死；

1.安装 `http` 服务，然后启动对外

```
[root@web ~]# setenforce 0
[root@web ~]# systemctl stop firewalld
[root@web ~]# systemctl disable firewalld
[root@web ~]# yum install httpd -y
[root@web ~]# systemctl start httpd
```

2.编写病毒脚本，尝试篡改网页内容

```
[root@web ~]# cat virus.sh
#!/usr/bin/bash
web_site=/var/www/html/index.html

while true
do
    echo "我是病毒Code" > ${web_site}
    sleep 20
done
```

3.锁定篡改文件，然后杀死病毒

```
[root@web ~]# chatttr +i /var/www/html/index.html
[root@web ~]# kill $(ps -ef | grep virus | grep -v grep | awk '{print $2}')
```