

15 群组的管理和文件权限管理

更新时间：2019-06-28 11:46:49



“

今天应做的事没有做，明天再早也是耽误了。

——裴斯泰洛齐

”

内容简介

1. 前言
2. 群组管理的命令
3. 修改文件的所有者和群组
4. chmod 命令：修改访问权限
5. 总结

1. 前言

上一课用户和权限，有权就任性。用户管理中，我们开启了用户与权限的新篇章，学了不少用户管理的基础知识。

这一课我们接着来学习群组管理和文件的权限管理。配合表格和原理图，相信这一课有点复杂的知识也不在话下。

2. 群组管理的命令

上一课我们说过，Linux 中每一个用户都属于一个特定的群组。

那你要问了：“那么我们刚才创建的 `thomas` 是属于哪个群组呢？我们以前创建的个人用户 `oscar` 又属于哪个群组呢？我们之前都没配置呀。”

事实上，如果你不设置用户的群组，那么默认会创建一个和它的用户名一样的群组，并且把用户划归到这个群组。

因为上一课我们删除了 `thomas` 这个用户，连他的家目录也删了。所以我们再用 `adduser thomas` 命令把 `thomas` 用户加回来。

我们可以用 `ls -l` 命令来看一下 `/home` 目录下的内容：

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# ls -l
total 8
drwxr-xr-x 20 oscar oscar 4096 May  5 19:04 oscar
drwxr-xr-x  2 thomas thomas 4096 May  5 08:01 thomas
root@oscar-laptop:/home#
```

可以看到，我们的 `oscar` 用户和 `thomas` 用户的家目录分别是 `/home/oscar` 和 `/home/thomas`。

每一行的各列都有不同意义，我们之前的课程里有讲解过。

所以，第三列表示文件或目录的所有者，第四列表示文件或目录的所在群组。

可以看到：`oscar` 这个目录的所有者是 `oscar`，群组是 `oscar`；`thomas` 这个目录的所有者是 `thomas`，群组是 `thomas`。

但是，把用户分在不同的群组，到底有何意义呢？

在用户不多的时候，我们会觉得一个用户属于一个群组（比如默认是与用户名相同的群组名）是挺不错的。但是一旦用户一多，你可能就想要创建群组了。

我们来学习如何管理群组。当然，群组还有权限的考量因素，我们这课之后会讲权限。

当然了，群组管理的命令也需要 `root` 身份。

addgroup: 创建群组

`addgroup` 是 `add` 和 `group` 的缩写，`add` 是英语“添加”的意思，`group` 是英语“群组”的意思。所以 `addgroup` 命令用于添加一个新的群组。

用法也很简单，和 `adduser` 命令类似，后接需要创建的群组名。例如：

```
addgroup friends
```

以上命令用于创建一个名为 `friends` 的群组。`friends` 是英语“朋友”的意思，也是美剧《老友记》的名字。

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# addgroup friends
Adding group `friends' (GID 1002) ...
Done.
root@oscar-laptop:/home#
```

在上图中，我们看到用 `addgroup` 命令创建了一个新的群组，名叫 `friends`，而且成功了。`Done` 是英语“完成”的意思。

很不错，不过目前 `friends` 这个群组还是空的，因为还没有往里面添加用户呢。

usermod 命令：修改用户账户

usermod 是 user 和 modify 的缩写，user 是英语“用户”的意思，modify 是“修改”的意思。usermod 命令用于修改用户的账户。

usermod 命令有好多参数，可以实现不同的功能。不过我们暂时只需要记得它的两个参数：

- -l: 对用户重命名，但是 /home 目录中的用户家目录名不会改变，需要手动修改；
- -g: 修改用户所在群组。此用户的家目录里的所有文件的所在群组会相应改变。

用法很简单，假如我要将 thomas 这个用户放到我刚创建的 friends 这个群组里，可以这样写：

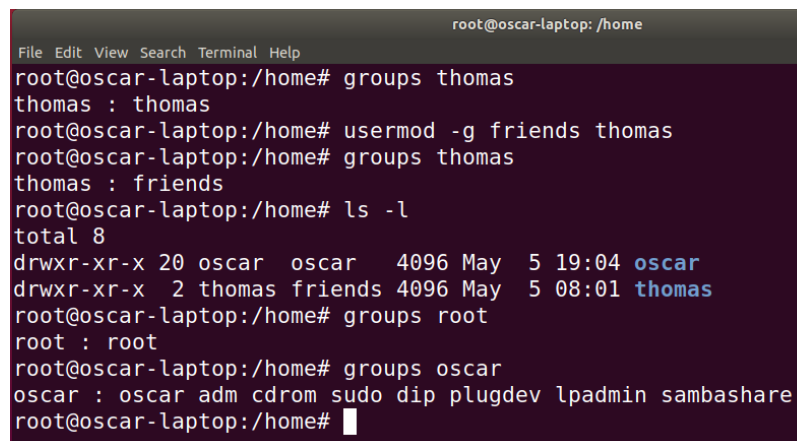
```
usermod -g friends thomas
```

我们知道，用户 thomas 之前的群组是 thomas，默认的。运行完 usermod -g friends thomas 之后，thomas 的群组就会变成 friends 了，/home/thomas 这个目录的群组也从 thomas 变成 friends。

我们怎么知道用户 thomas 的群组已经改变为 friends 了呢？

我们可以用 groups 命令，这个命令可以获知一个用户属于哪个（些）群组。

用法很简单，后接用户名就可以了，当然用户要存在才行：



```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# groups thomas
thomas : thomas
root@oscar-laptop:/home# usermod -g friends thomas
root@oscar-laptop:/home# groups thomas
thomas : friends
root@oscar-laptop:/home# ls -l
total 8
drwxr-xr-x 20 oscar oscar 4096 May  5 19:04 oscar
drwxr-xr-x  2 thomas friends 4096 May  5 08:01 thomas
root@oscar-laptop:/home# groups root
root : root
root@oscar-laptop:/home# groups oscar
oscar : oscar adm cdrom sudo dip plugdev lpadmin sambashare
root@oscar-laptop:/home#
```

可以看到：

- thomas 的群组从 thomas 变成了 friends，/home/thomas 这个目录的群组也从 thomas 变成 friends；
- root 的群组就是 root；
- oscar 的群组有好几个，说明我加入了很多“组织”，其中一个群组是 oscar。

当然我们也可以一次性将一个用户添加到多个群组，就用 -G 参数（大写的 G）。

首先我们用 addgroup 命令来新建两个群组：happy 和 funny。happy 是英语“开心的”的意思，funny 是英语“有趣的”的意思：

```
addgroup happy
```

然后：

```
addgroup funny
```

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# addgroup happy
Adding group `happy' (GID 1003) ...
Done.
root@oscar-laptop:/home# addgroup funny
Adding group `funny' (GID 1004) ...
Done.
root@oscar-laptop:/home#
```

接着，我们运行以下命令：

```
usermod -G friends,happy,funny thomas
```

以上命令把 thomas 添加到 friends、happy 和 funny 三个群组。记得群组名之间要用逗号分隔，而且没有空格：

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# addgroup happy
Adding group `happy' (GID 1003) ...
Done.
root@oscar-laptop:/home# addgroup funny
Adding group `funny' (GID 1004) ...
Done.
root@oscar-laptop:/home# usermod -G friends,happy,funny thomas
root@oscar-laptop:/home# groups thomas
thomas : friends happy funny
root@oscar-laptop:/home#
```

注意：使用 usermod 时要小心，因为配合 -g 或 -G 参数时，它会把用户从原先的群组里剔除，加入到新的群组。如果你不想离开原先的群组，又想加入新的群组，可以在 -G 参数的基础上加上 -a 参数，a 是英语 append 的缩写，表示“追加”。例如：

```
addgroup good
```

然后：

```
usermod -aG good thomas
```

以上命令就把 thomas 追加到群组 good 里了，这样 thomas 就属于四个群组：friends、happy、funny 和 good。good 是英语“好的”的意思。

可以用 groups 命令测试一下。

注意：groups 命令如果单独用，不加任何参数，会显示当前用户所在群组。

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# groups thomas
thomas : friends happy funny
root@oscar-laptop:/home# addgroup good
Adding group `good' (GID 1005) ...
Done.
root@oscar-laptop:/home# usermod -aG good thomas
root@oscar-laptop:/home# groups thomas
thomas : friends happy funny good
root@oscar-laptop:/home# groups
root
root@oscar-laptop:/home#
```

记得，追加群组的时候，一定要用大写的 G 这个参数，不能用小写的 g 这个参数，即使只追加一个群组。

delgroup 命令：删除群组

delgroup 是 delete 和 group 的缩写，delete 是英语“删除”的意思，group 是英语“群组”的意思。所以 delgroup 命令用于删除一个已存在的群组。

用法很简单，后接想要删除的群组名。例如：

```
delgroup happy
```

就删除了 happy 这个群组。

再用 groups 命令测试，发现 thomas 只属于 friends、funny 和 good 这三个群组了。因为 happy 这个群组被删除了嘛。

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# delgroup happy
Removing group `happy' ...
Done.
root@oscar-laptop:/home# groups thomas
thomas : friends funny good
root@oscar-laptop:/home#
```

注意：addgroup 和 delgroup 命令只是 Debian 一族（包括 Ubuntu）才有的命令。对于其它的 Linux 发行版，一般来说，添加用户和删除用户是用 groupadd 和 groupdel 命令。

3. 修改文件的所有者和群组

只有 root 用户可以修改一个文件的所有者和群组。

比如说，我自己的用户 oscar 的家目录有一个文件，file.txt，是我刚创建的。

我们用 ls -l 命令来看一下它的信息：

```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ touch file.txt  
oscar@oscar-laptop:~$ ls -l file.txt  
-rw-r--r-- 1 oscar oscar 0 May  5 21:40 file.txt  
oscar@oscar-laptop:~$
```

可以看到，file.txt 的所有者和群组都是 oscar。

现在我决定，把这个文件转让给 thomas，也就是让 file.txt 的所有者变为 thomas，怎么做呢？

chown 命令：改变文件的所有者

此命令也需要 root 身份才能运行。

chown 是 change 和 owner 的缩写，change 是英语“改变”的意思，owner 是英语“所有者”的意思。因此 chown 命令用于改变文件的所有者。

用法也很简单，后接新的所有者的用户名，再接文件名。例如：

```
chown thomas file.txt
```

```
root@oscar-laptop: /home/oscar  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ ls -l file.txt  
-rw-r--r-- 1 oscar oscar 0 May  5 21:40 file.txt  
oscar@oscar-laptop:~$ chown thomas file.txt  
chown: changing ownership of 'file.txt': Operation not permitted  
oscar@oscar-laptop:~$ sudo su  
[sudo] password for oscar:  
root@oscar-laptop:/home/oscar# chown thomas file.txt  
root@oscar-laptop:/home/oscar# ls -l file.txt  
-rw-r--r-- 1 thomas oscar 0 May  5 21:40 file.txt  
root@oscar-laptop:/home/oscar#
```

可以看到，用 chown 命令，把 file.txt 文件的所有者改为 thomas 之后，file.txt 的所在群组是不变的，还是 oscar。

正所谓“身在曹营心在汉”。那么如何使它“身在曹营心也在曹营”呢？

就要用到 chgrp 命令了。

chgrp 命令：改变文件的群组

chgrp 是 change 和 group 的缩写，change 是英语“改变”的意思，group 是英语“群组”的意思。chgrp 命令用于改变文件的群组。

用法也很简单，后接新的群组名，再接文件名。例如：

```
chgrp thomas file.txt
```

```
root@oscar-laptop: /home/oscar
File Edit View Search Terminal Help
root@oscar-laptop:/home/oscar# ls -l file.txt
-rw-r--r-- 1 thomas oscar 0 May  5 21:40 file.txt
root@oscar-laptop:/home/oscar# chgrp thomas file.txt
root@oscar-laptop:/home/oscar# ls -l file.txt
-rw-r--r-- 1 thomas thomas 0 May  5 21:40 file.txt
root@oscar-laptop:/home/oscar#
```

好了，这下 file.txt 的所有者和群组都是 thomas 了。

其实，chown 命令也可以改变文件的群组，用法如下：

```
chown thomas:friends file.txt
```

这句命令就把 file.txt 这个文件的所有者改为 thomas，群组改为 friends 了。用法也很简单，就是在所有者和群组之间用冒号隔开。

```
root@oscar-laptop: /home/oscar
File Edit View Search Terminal Help
root@oscar-laptop:/home/oscar# ls -l file.txt
-rw-r--r-- 1 thomas thomas 0 May  5 21:40 file.txt
root@oscar-laptop:/home/oscar# chown thomas:friends file.txt
root@oscar-laptop:/home/oscar# ls -l file.txt
-rw-r--r-- 1 thomas friends 0 May  5 21:40 file.txt
root@oscar-laptop:/home/oscar#
```

-R 参数：递归设置子目录和子文件

chown 命令的 -R 参数非常有用，还记得以前我们有些命令也会使用 -R 参数么？

是的，R 是 recursive 的缩写，表示“递归”。所以如果 chown 命令配上 -R 参数，就会使得被修改的目录的所有子目录和子文件都被改变所有者（或者连群组也改变，如果用上述冒号的方法来同时修改所有者和群组）。

例如，我突然变得“很坏”，想要把用户 thomas 的家目录的所有子目录和文件都占为己有。我可以这么做：

```
chown -R oscar:oscar /home/thomas
```

这样不但使 /home/thomas 这个目录的所有者和群组都变成 oscar，而且其子目录和子文件也都是如此：

```
root@oscar-laptop: /home
File Edit View Search Terminal Help
root@oscar-laptop:/home# ls -l
total 8
drwxr-xr-x 20 oscar oscar 4096 May  5 21:40 oscar
drwxr-xr-x  2 thomas friends 4096 May  5 08:01 thomas
root@oscar-laptop:/home# chown -R oscar:oscar thomas
root@oscar-laptop:/home# ls -l
total 8
drwxr-xr-x 20 oscar oscar 4096 May  5 21:40 oscar
drwxr-xr-x  2 oscar oscar 4096 May  5 08:01 thomas
root@oscar-laptop:/home#
```

可以看到，/home/thomas 都归我（oscar）所有了。可怜的 thomas，正所谓“三十年河东，三十年河西。昔日权倾天下有，今朝都归我所有”。

4. chmod 命令：修改访问权限

好了，这一节我们要攻坚这一课最难的部分了：访问权限。

权限的原理

在 Linux 系统里，每个文件和目录都有一列权限属性。这一列访问权限指明了谁有读的权利，谁有修改的权利，谁有运行的权利。

我们其实早就见过访问权限了。是的，就在我们运行 `ls -l` 命令的时候，显示的每个文件或目录的第一列信息就是访问权限。

比如我们在当前用户的家目录下运行 `ls -l` 命令试试：

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~$ ls -l
total 44
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Desktop
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Documents
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Downloads
-rw-r--r-- 1 thomas friends 0 May 5 21:40 file.txt
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Music
drwxr-xr-x 4 oscar oscar 4096 May 3 21:49 one
drwxr-xr-x 3 oscar oscar 4096 May 3 21:44 one_copy
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Pictures
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Public
-rw-r--r-- 1 oscar oscar 0 May 3 21:14 renamed_file
drwxr-xr-x 1 root root 640 Jul 12 2018 share
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Templates
drwxr-xr-x 2 oscar oscar 4096 May 3 22:46 test
drwxr-xr-x 2 oscar oscar 4096 May 1 10:48 Videos
oscar@oscar-laptop: ~$
```

上图中文件信息的第一列比较复杂，我们可以看到不少 `d`、`r`、`w`、`l`、`x` 等字母。如果不细分的话，这些我们可以通称为文件访问权限符。

以下列出我们看到的字母的含义：

- `d`：英语 `directory` 的缩写，表示“目录”。就是说这是一个目录；
- `l`：英语 `link` 的缩写，表示“链接”。就是说这是一个链接；
- `r`：英语 `read` 的缩写，表示“读”。就是说可以读这个文件；
- `w`：英语 `write` 的缩写，表示“写”。就是说可以写这个文件，也就是可以修改；
- `x`：英语 `execute` 的缩写，表示“执行，运行”。就是说可以运行这个文件。

如果 `x` 权限在一个目录上，那么表示的是这个目录可以被读，也就是可以打开此目录来看其子目录和子文件，如果它同时有 `r` 权限的话。

如果相应位置有字母，表示有相应权限；如果相应位置是一个短横 `-`，则表示没有相应权限。

为什么我们看到这一排有好多个重复出现的 `r`、`w` 和 `x` 呢？

那是因为访问权限是按照用户来划分的：



属性

所有者

群组用户

其他用户

如上图，除开第一个表示文件或目录属性的符号（此处是 `d`，表示目录；如果是 `l`，则是链接；还有其它字母，我们暂时不深究；如果是短横 `-`，那么是普通文件。），其它的 9 个符号被划分为三组，从左到右分别表示：

- 第一组 `rwx` 表示文件的所有者对于此文件的访问权限；
- 第二组 `rwx` 表示文件所属群组的其他用户对于此文件的访问权限；
- 第三组 `rwx` 表示除前两组之外的其他用户对于此文件的访问权限。

我们用一个具体的文件来作为例子分析一下：

```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ ls -l renamed_file  
-rw-r--r-- 1 oscar oscar 0 May  3 21:14 renamed_file  
oscar@oscar-laptop:~$
```

可以看到，`renamed_file` 这个文件的访问权限是：

```
-rw-r--r--
```

我们从左到右来分析这些符号都表示什么：

- `-` 第一个短横表示这是一个普通文件。如果此处是 `d`，那么表示目录；如果是 `l`，那么表示链接等等；
- `rw-` 表明文件的所有者（此处是 `oscar`）对文件有读、写的权限，但是没有运行的权限。也很好理解，因为这是一个普通文件，默认没有可执行的属性。记住：如果有 `w` 权限（写的权限），那么表明也有删除此文件的权限；
- `r--` 表明文件所在的群组（此处是 `oscar`）的其他用户（除了 `oscar` 之外）只可以读此文件，但不能写也不能执行，“可远观而不可亵玩焉”；
- `r--` 表示其他用户（除去 `oscar` 这个群组的用户）只可以读此文件，但不能写也不能执行。

综上所述，`renamed_file` 这个文件是一个普通文件，不是一个目录，也不是链接文件，它的所有者 `oscar` 可以读写它，但不能执行；其他的用户只能读。

那么 `root` 呢？对于此文件 `root` 用户的访问权限是什么呢？

记住：`root` 是超级管家，它有所有权限，“只有它想不到的，没有它做不到的”。它可以读、写、运行任意文件。

chmod 命令：修改文件的访问权限

既然我们已经学会了如何查看和理解文件的访问权限，我们就来学习如何修改文件的访问权限吧。

我们要用到 `chmod` 命令，这个命令也是 Linux 中常用的命令。

毕竟“一朝权倾天下有”，“争权夺利”谁不喜欢啊。开个玩笑……

开始讲解之前，要说明一点，`chmod` 命令不需要是 `root` 用户才能运行。只要你是此文件的所有者，你就可以用 `chmod` 来修改文件的访问权限。

`chmod` 是 `change` 和 `mode` 的缩写，`change` 是英语“改变”的意思，`mode` 是“模式”的意思。`chmod` 命令用于修改文件的各种访问权限。

`chmod` 这个命令充满魅力，因为它的用法不止一种，好像百变星君，令人捉摸不透。

最常见的用法应该是数字式的。

用数字来分配权限：`chmod` 的绝对用法

我们接下来要做一些加法，大家准备好了吗？不要让小学数学老师哭晕在体育办公室哦。什么？你的小学数学是地理老师教的，好，算你厉害……

不要怕，只是做一些极为简单的加法，我们只要心算就可以了。

事实上，Linux 系统为每种权限（`r`、`w` 和 `x`）分配了对应的数字：

权限	数字
<code>r</code>	4
<code>w</code>	2
<code>x</code>	1

所以，如果我们要合并这些权限，就需要做简单的加法了：将对应的数字相加。

假如我们要分配读、写权限，那么我们就要用 `4+2`，就等于 `6`。数字 `6` 表示具有读和写权限。

以下是可能的组合形式：

权限	数字	计算
—	0	$0 + 0 + 0$
<code>r—</code>	4	$4 + 0 + 0$
<code>-w—</code>	2	$0 + 2 + 0$
<code>-x—</code>	1	$0 + 0 + 1$
<code>rw—</code>	6	$4 + 2 + 0$
<code>-wx—</code>	3	$0 + 2 + 1$
<code>r-x—</code>	5	$4 + 0 + 1$
<code>rw-x</code>	7	$4 + 2 + 1$

不难吧？

所以，对于访问权限的三组（所有者的权限、群组用户的权限、其他用户的权限），我们只要分别做加法就可以了，然后把三个和连起来。

例如，`640` 分别表示：

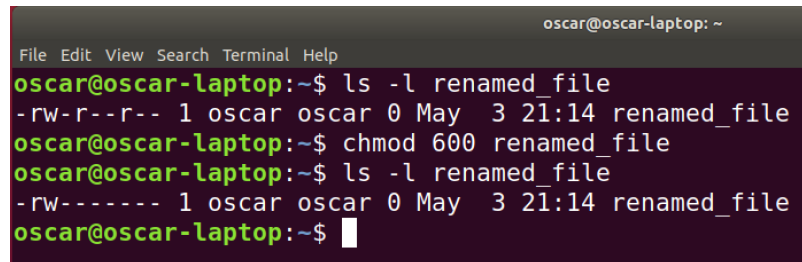
- 文件的所有者有读和写的权限；
- 文件所在群组的其他用户具有读的权限；
- 除此之外的其他用户没有任何权限。

因此，我们可以给的最宽泛的权限就是 777：所有者，群组用户，其他用户都有读、写和运行的权限。这样，所有人就都可以对此文件“为所欲为”了。

相反，如果权限是 000，那么没有人能对此文件做什么。当然，除了 root 之外，root 可以做任何事。

我们现在来修改 renamed_file 的权限试试：

```
chmod 600 renamed_file
```

A terminal window titled 'oscar@oscar-laptop: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The user runs 'ls -l renamed_file' showing '-rw-r--r-- 1 oscar oscar 0 May 3 21:14 renamed_file'. Then they run 'chmod 600 renamed_file'. Finally, they run 'ls -l renamed_file' again, showing '-rw----- 1 oscar oscar 0 May 3 21:14 renamed_file'.

```
oscar@oscar-laptop:~$ ls -l renamed_file
-rw-r--r-- 1 oscar oscar 0 May 3 21:14 renamed_file
oscar@oscar-laptop:~$ chmod 600 renamed_file
oscar@oscar-laptop:~$ ls -l renamed_file
-rw----- 1 oscar oscar 0 May 3 21:14 renamed_file
oscar@oscar-laptop:~$
```

可以看到，我们的 renamed_file 文件的访问权限被修改为了：

```
rw-----
```

正好是 600。

所以现在只有 oscar 可以读和写此文件，其他人都不能做什么。当然，除了 root 之外。

用字母来分配权限：chmod 的相对用法

除了用数字，我们也可以用另一种方式来分配文件的访问权限：用字母。

原理是类似的，但是有时用字母的方式更加精巧，因为不需要一次性把三组权限都写出来。

我们需要知道不同的字母代表什么：

- u: user 的缩写，是英语“用户”的意思。表示所有者；
- g: group 的缩写，是英语“群组”的意思。表示群组用户；
- o: other 的缩写，是英语“其他”的意思。表示其他用户；
- a: all 的缩写，是英语“所有”的意思。表示所有用户。

当然了，和这些字母配合的还有几个符号：

- +: 加号，表示添加权限；
- -: 减号，表示去除权限；
- =: 等号，表示分配权限。

接下来，我们举例说明如何使用：

```
#文件 file.txt 的所有者增加读和运行的权限。  
chmod u+rx file.txt
```

```
#文件 file.txt 的群组其他用户增加读的权限。  
chmod g+r file.txt
```

```
#文件 file.txt 的其他用户移除读的权限。  
chmod o-r file.txt
```

```
#文件 file.txt 的群组其他用户增加读的权限，其他用户移除读的权限。  
chmod g+r o-r file.txt
```

```
#文件 file.txt 的群组其他用户和其他用户均移除读的权限。  
chmod go-r file.txt
```

```
#文件 file.txt 的所有用户增加运行的权限。  
chmod +x file.txt
```

```
#文件 file.txt 的所有者分配读，写和执行的权限；  
#群组其他用户分配读的权限，不能写或执行；  
#其他用户没有任何权限。  
chmod u=rwx,g=r,o=- file.txt
```

-R 参数：递归地修改访问权限

-R 参数可是“死性不改”，上一课可以配合 cp 命令来递归拷贝文件，这一课又来“捣蛋”：

chmod 配合 -R 参数可以递归地修改文件访问权限。

假如我想要只允许 oscar 这个用户能读、写、运行 /home/oscar 这个目录的所有文件（当然，root 不算，root 可以做任何事），该怎么做呢：

```
chmod -R 700 /home/oscar
```

就是这么简单。

关于权限，大家可以多做练习，来巩固知识点。

小结

1. 有一些命令的运行须要先切换到 root 身份。比如：usermod（用于修改用户账户）、chgrp（用于修改群组）和 chown（用于修改所有者和群组）；
2. chmod 命令不需要 root 身份就可以运行。我们可以用 chmod 命令来修改文件的访问权限。有三种权限：r（读权限）、w（写权限）和 x（运行权限）。

今天的课就到这里，一起加油吧！