Linux系统是一个多用户多任务的分时操作系统，任何一个要使用系统资源的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。

为了更加方便的管理多个用户，就出现了用户组的概念，关于用户和用户组：

- 系统上的每个进程(运行的程序)都是作为特定用户运行
- 每个文件是由一个特定的用户拥有
- 访问文件和目录受到用户的限制
- 与正在运行的进程相关联的用户确定该进程可访问的文件和目录

实现用户账号的管理，要完成的工作主要有如下几个方面：

- 用户账号的添加、删除与修改
- 用户口令的管理
- 用户组的管理

用户和用户组查看

id

用于显示用户的ID，以及所属群组的 ID

id会显示用户以及所属群组的实际与有效 ID。若两个 ID 相同，则仅显示实际 ID。若仅指定用户名称，则显示目前用户的 ID。

```
id [OPTION]... [USER]
```

选项	含义
-g	显示用户所属群组的 ID
-G	显示用户所属附加群组的 ID
-n	显示用户，所属群组或附加群组的名称
-r	显示用户真实 ID，用户真实的 uid
-u	显示用户有效 ID，可以都某些高权限用户，通过有效 id 限制权限

uid 约定

Linux 操作系统会依据用户的 uid 数值来判定这个用户的角色，分别如下

- **0**：超级管理员，也就是root，在linux系统中拥有所有权力
- **1~999**：系统用户，系统用户往往是用来约束系统中的服务的
- **1000+**：普通用户，可以用来登陆和使用Linux操作系统

关于root用户

- uid是0
- 拥有操作系统所有权力
- 该用户有权力覆盖文件系统上的普通权限
- 安装或删除软件并管理系统文件和目录
- 大多数设备只能由root控制

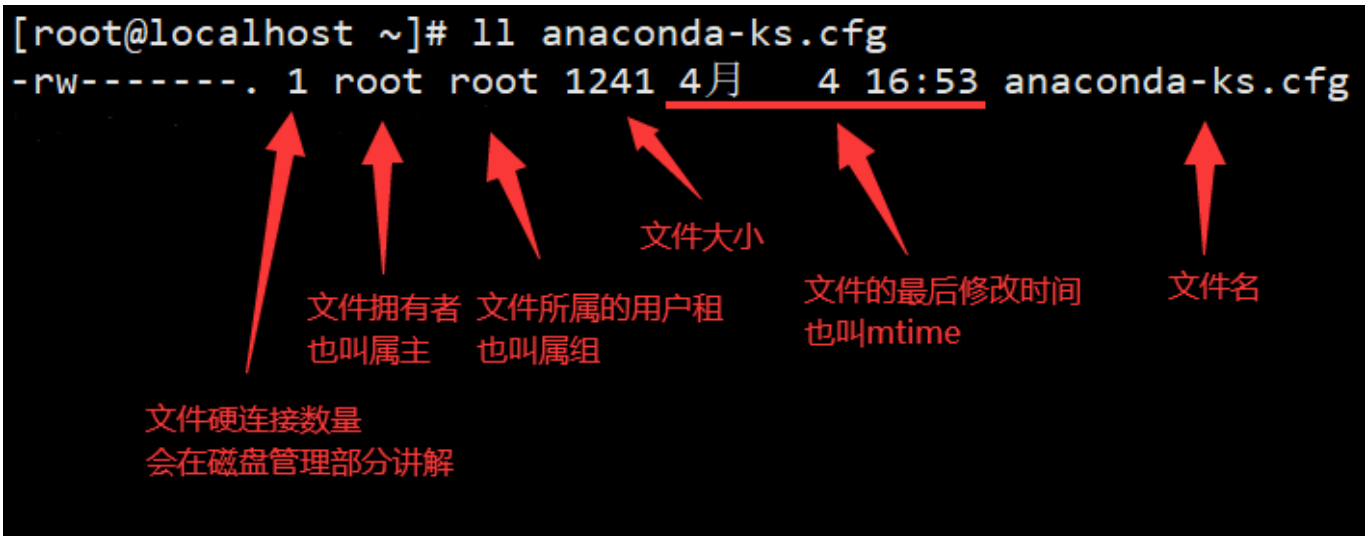
案例演示

查看当前登陆的用户信息

```
[root@localhost ~]# id
uid=0(root) gid=0(root) 组=0(root) 环境
=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

查看文件的拥有者

```
[root@localhost ~]# ll anaconda-ks.cfg
-rw-------. 1 root root 1241 4月   4 16:53 anaconda-ks.cfg
```



查看运行进程的用户名，ps命令会在后面进程管理部分讲解

```
[root@localhost ~]# ps aux
USER PID %CPU %MEM    VSZ   RSS TTY STAT START   TIME COMMAND
root 2  0.0  0.0      0     0 ? S    09:06   0:00 [kthreadd]
root 3  0.0  0.0      0     0 ? S    09:06   0:01 [ksoftirqd/0]
root 4  0.1  0.0      0     0 ? R    09:06   0:09 [kworker/0:0]
root 5  0.0  0.0      0     0 ? S<   09:06   0:00 [kworker/0:0H]
```

相关的文件

之前说过 Linux 一切皆文件，所以用户和用户组相关的信息也都是保存在文本文件中的，下面列举出相关的文件。

passwd 文件

用于保存用户的信息，一般第一行是 root 用户，下面都是其他用户

```
[root@localhost ~]# head -n 1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
# 这个格式为用户名:密码:uid:gid:描述:家目录:登陆后执行的命令
```

shadow 文件

格式中密码占位置太长了，所以使用x来替代，Linux系统会到shadow中查找x部分的的密码内容

```
[root@localhost ~]# head -n 1 /etc/shadow
root:$6$frokclXSnQa8EbKs$pWElbjPlmxjYh30tr8qLsTQVOhuPg7GmW9Sanm2yXAK8TNMgj
e1gyc/vwPgqvmSMf6VaoEvveM0gFvtETmXy/.::0:99999:7:::
# 这个格式为用户名：加密密码：最后一次修改时间：最小修改时间间隔：密码有效期：密码需要变
更前的警告天数：密码过期后的宽限时间：账号失效时间：保留字段
```

格式不需要大家记住，只需要知道关于这个用户的密码和有效期都在这个文件中即可。

密码在passwd文件中会使用加密算法加密，所以别想知道我的密码是什么，加密算法默认是$6，这个类型6的加密算法是sha-512。我们也可以在man手册中看到对shadow文件的详细解释。

```
[root@localhost ~]# man 5 shadow
# man手册一个有9个章节，其中第5个章节是对文件格式的说明
# 对man手册感兴趣的同学，也可以自己在网上查找学习man手册的更多内容
```

group文件

用户和组的对应关系，会保存在group文件中

```
[root@localhost ~]# head -n 1 /etc/group
root:x:0:
# 这个格式是组名:口令:组标识号:组内用户列表
```

用户组管理

添加用户组：groupadd

groupadd 命令用于创建一个新的工作组，新工作组的信息将被添加到系统文件中

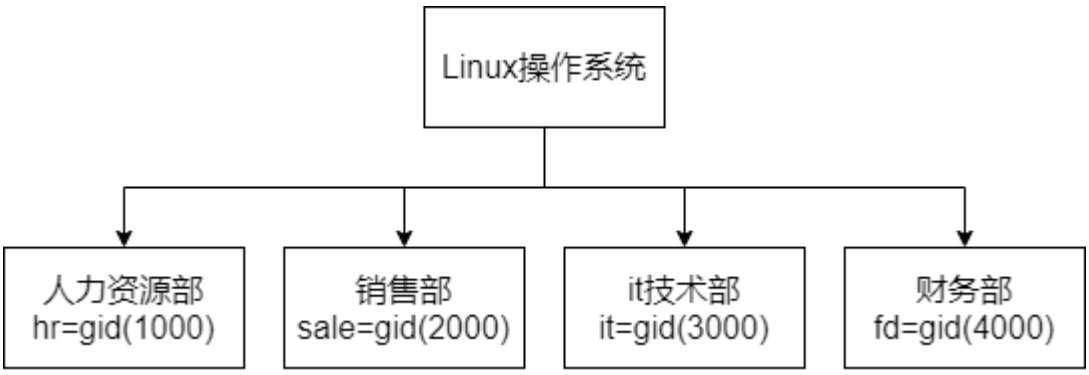
```
groupadd [选项] 组
```

常用选项

选项	含义
-g	指定新建工作组的 id
-r	创建系统工作组，系统工作组的组ID小于 500
-K	覆盖配置文件/etc/login.defs
-o	允许添加组 ID 号不唯一的工作组
-f	如果指定的组已经存在，此选项将失明了仅以成功状态退出。当与 -g 一起使用，并且指定的GID_MIN已经存在时，选择另一个唯一的GID（即-g关闭）

案例演示

按照下图创建组，并且指定gid，并且检查是否成功




```
[root@localhost ~]# groupadd hr -g 1000
[root@localhost ~]# groupadd sale -g 2000
[root@localhost ~]# groupadd it -g 3000
[root@localhost ~]# groupadd fd -g 4000
[root@localhost ~]# tail -n 4 /etc/group
hr:x:1000:
sale:x:2000:
it:x:3000:
fd:x:4000:
```

修改用户组：groupmod

groupmod命令用于更改群组识别码或名称

```
groupmod [选项] 组
```

常用选项

- **-g**：将组 ID 改为 GID
- **-n**：改名为 NEW_GROUP
- **-o**：允许使用重复的 GID

案例演示

- 修改fd组的名字为finance

```
[root@localhost ~]# groupmod -n finance fd
[root@localhost ~]# tail -n 1 /etc/group
finance:x:4000:
```

删除用户组：groupdel

groupdel命令用于删除群组

需要从系统上删除群组时，可用groupdel(group delete)指令来完成这项工作。倘若该群组中仍包括某些用户，则必须先删除这些用户后，方能删除群组。

```
groupdel [组名]
```

案例演示

- 删除一个用户组

```
[root@localhost ~]# groupadd test
[root@localhost ~]# groupdel test
```

用户组成员管理：gpasswd

gpasswd 是 Linux 下工作组文件 /etc/group 和 /etc/gshadow 管理工具，用于将一个用户添加到组或者从组中删除

```
gpasswd [选项] 组
```

常用选项

- **-a**：添加用户到组；
- **-d**：从组删除用户；
- **-A**：指定管理员，可以执行添加或者删除组员；
- **-M**：替换组中的全部用户列表，不包含在内的用户将会从组中删除；
- **-R**：限制用户登入组，只有组中的成员才可以用newgrp加入该组。

案例演示

创建用户 itadmin，并且将其加入 it 组

```
[root@localhost ~]# useradd itadmin
[root@localhost ~]# gpasswd -a itadmin it
正在将用户“itadmin”加入到“it”组中
[root@localhost ~]# cat /etc/group |grep it:
it:x:3000:itadmin
# 在组文件中，可以看到这个组的成员
[root@localhost ~]# id itadmin
uid=6667(itadmin) gid=6667(itadmin) 组=6667(itadmin),3000(it)
# 在用户的信息中，可以看到这个用户的所属组
```

用户管理

添加用户：useradd

useradd 可以用来添加新的用户账号

```
useradd [选项] 用户名
```

常用选项

- **-c comment**：指定一段注释性描述。
- **-d 目录**：指定用户主目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。
- **-m**：创建用户的主目录
- **-g 用户组**：指定用户所属的用户组，默认会创建一个和用户名同名的用户组。
- **-G 用户组**：用户组 指定用户所属的附加组，一个用户可以属于多个附加组。
- **-s Shell文件**：指定用户的登录Shell。
- **-u 用户号**：指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号。

案例演示

- 添加一般用户

```
[root@localhost ~]# useradd user01
```

- 为添加的用户指定相应的用户组

```
[root@localhost ~]# useradd -g root user02
```

- 为新添加的用户指定home目录

```
[root@localhost ~]# useradd -d /home/test user03
```

- 建立一个不给登录的用户

```
[root@localhost ~]# useradd -s /sbin/nologin user04
```

- 查看用户描述信息

```
[root@localhost ~]# useradd -c "this is python_developmer user" python-developmer
[root@localhost ~]# cat /etc/passwd |grep python-developmer
python-developmer:x:1005:1005:this is python_developmer user:/home/python-developmer:/bin/bash
```

切换用户： su

su命令用户Linux系统中的用户切换

```
su [选项] 用户名
```

常用选项

- `-`：以目标用户的环境变量启动新会话。这将模拟用户完全登录到新用户账户，包括其家目录、环境变量等。
- `-c 命令`：执行指定的命令，并在执行完毕后返回原始用户。
- `-s shell`：使用指定的 shell 替代目标用户的默认 shell。

修改用户： usermod

usermod命令用于修改用户帐号

usermod可用来修改用户帐号的各项设定

```
usermod [选项] 登录
```

常用选项

- **-c<备注>**：修改用户帐号的备注文字。
- **-a**：追加，默认的修改是覆盖
- **-d登入目录>**：修改用户登入时的目录。
- **-e<有效期限>**：修改帐号的有效期限。
- **-f<缓冲天数>**：修改在密码过期后多少天即关闭该帐号。
- **-g<群组>**：修改用户所属的群组。
- **-G<群组>**：修改用户所属的附加群组。
- **-l<帐号名称>**：修改用户帐号名称。
- **-L**：锁定用户密码，使密码无效。
- **-s**：修改用户登入后所使用的shell。
- **-u**：修改用户ID。
- **-U**：解除密码锁定。

案例演示

- 更改登录的目录

```
[root@localhost ~]# usermod -d /home user01
[root@localhost ~]# su - user01
[user01@localhost ~]$ pwd
/home
```

- 改变用户的uid

```
[root@localhost ~]# usermod -u 6666 user02
```

删除用户：userdel

userdel命令用于删除用户帐号

userdel可删除用户帐号与相关的文件。若不加参数，则仅删除用户帐号，而不删除相关文件

```
userdel [-r][用户帐号]
```

常用选项

- **-r**：删除用户登入目录以及目录中所有文件

案例演示

删除用户账号

```
[root@localhost ~]# userdel user04
```

删除用户账户和家目录

```
[root@localhost ~]# userdel -r user03
```

passwd 文件中的 shell

查看/etc/passwd文件会发现在每行的最后是登录成功之后执行的命令，有两种是使用最为频繁的：

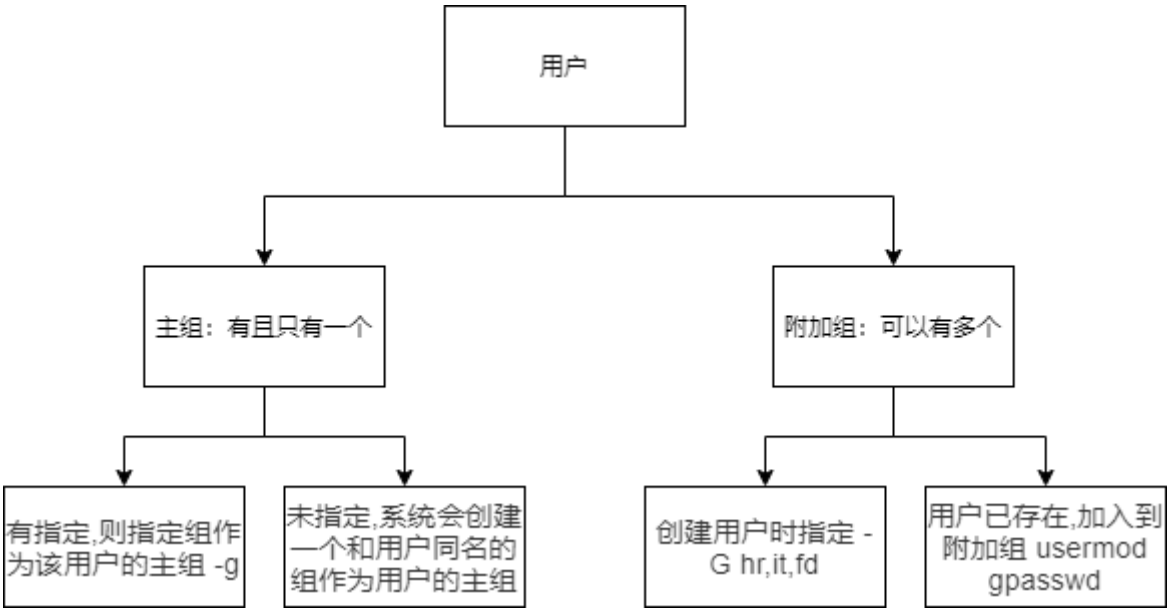
- /bin/bash：这个是 Linux 的命令行工具，我们正常登陆之后默认就是进入命令行
- /sbin/nologin：如果写成 nologin，那么用户将无法登录，有些用户是作为进程权限管理而存在的，不需要登录。如果提供登录的功能反而不安全，所以写成 nologin

```
[root@localhost ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

我们可以新建一个用户，然后尝试自定义登录成功之后执行的命令，用来加深印象。

```
[root@localhost ~]# useradd test01
[root@localhost ~]# tail -n 1 /etc/passwd
test01:x:1000:1000::/home/test01:/bin/vi
```

切换到test01用户，会发现自动进入vi的界面，说明最后的这个段内容就是用户登录之后会运行的程序



用户密码管理

root用户可以直接设置普通用户密码，普通用户必须要提供原密码，才可以修改自己密码。

passwd

passwd命令用来更改使用者的密码

```
passwd [选项...] <帐号名称>
```

常用选项

- **-k**：保持身份验证令牌不过期
- **-d**：删除已命名帐号的密码(只有根用户才能进行此操作)
- **-l**：锁定指名帐户的密码(仅限 root 用户)
- **-u**：解锁指名账户的密码(仅限 root 用户)
- **-x**：密码的最长有效时限(只有根用户才能进行此操作)
- **-n**：密码的最短有效时限(只有根用户才能进行此操作)
- **-w**：在密码过期前多少天开始提醒用户(只有根用户才能进行此操作)
- **-i**：当密码过期后经过多少天该帐号会被禁用(只有根用户才能进行此操作)
- **-S**：报告已命名帐号的密码状态(只有根用户才能进行此操作)
- **--stdin**：从标准输入读取令牌(只有根用户才能进行此操作)

案例演示

修改test01用户密码

```
[root@localhost ~]# passwd test01
更改用户 test01 的密码 。
新的 密码:
```

```
重新输入新的 密码:
passwd: 所有的身份验证令牌已经成功更新。
```

使用管道符设置用户密码

```
[root@localhost ~]# echo 123456 | passwd --stdin test01
更改用户 test01 的密码 。
passwd: 所有的身份验证令牌已经成功更新。
```

login.defs文件

/etc/login.defs文件是用来创建用户时进行一定的限制，但是优先级低于/etc/passwd和/etc/shadow,如果有冲突的地方,系统会以/etc/passwd和/etc/shadow为准

下面是这个文件的内容，egrep命令我们后续会讲到，这边可以理解为不看文件的注释和空行。

```
[root@localhost ~]# egrep -v '^[ ]*$|^#' /etc/login.defs
MAIL_DIR    /var/spool/mail  # 系统消息(邮件)文件夹
PASS_MAX_DAYS   99999        # 密码有效最大天数
PASS_MIN_DAYS   0            # 密码有效最小天数
PASS_MIN_LEN    5            # 密码长度
PASS_WARN_AGE   7            # 密码失效警告倒计时
UID_MIN                  1000    # 用户UID最小1000
UID_MAX                 60000    # 用户UID最大60000
SYS_UID_MIN               201    # 系统用户UID最小201
SYS_UID_MAX               999    # 系统用户UID最大999
GID_MIN                  1000    # 用户组GID最小1000
GID_MAX                 60000    # 用户组GID最大60000
SYS_GID_MIN               201
SYS_GID_MAX               999
CREATE_HOME yes              # 创建家目录
UMASK           077          # 创建文件/目录的权限掩码
USERGROUPS_ENAB yes          # 创建用户时同时生成组是  如果此处是no 创建的用户 会
是gid=100(users)groups=100(users)
ENCRYPT_METHOD SHA512        # 加密  方法  sha 512 这个方法生成的密码
在/etc/shadow里面的第二列会以$6$开头
```

chage

chage是用于更改用户密码过期信息

```
chage [选项] 登录
```

常用选项

- **-d**：设置密码的最后更改日期
- **-E 过期日期**：设置账号的过期日期
- **-I INACITVE**：设置密码过期后若未更改，多少天后用户账号被禁用。
- **-l**：显示用户账号的密码过期信息。
- **-m 最小天数**：设置两次改变密码之间相距的最小天数
- **-M 最大天数**：设置将两次改变密码之间相距的最大天数
- **-W 警告天数**：设置密码过期前的警告天数

案例演示

强制用户在下次登录的时候换密码

```
[root@localhost ~]# chage -d 0 test01
[root@localhost ~]# logout
You must change your password now and login again!
更改用户 test01 的密码 。
为 test01 更改 STRESS 密码。
（当前）UNIX 密码:

[root@localhost ~]# chage -l root
最近一次密码修改时间                                        : 从不
密码过期时间                                          : 从不
密码失效时间                                          : 从不
帐户过期时间                                                : 从不
两次改变密码之间相距的最小天数          : 0
两次改变密码之间相距的最大天数          : 99999
在密码过期之前警告的天数        : 7

# 设置用户 user01 的密码过期前7天提醒用户修改密码
[root@localhost ~]# chage -W 7 user01

# 设置用户 user01 的密码最短使用天数为7天，即密码修改后7天内不能再次修改
[root@localhost ~]# chage -m 7 user01
[root@localhost ~]# chage -l user01
最近一次密码修改时间                                        : 11月 08, 2024
密码过期时间                                          : 从不
密码失效时间                                          : 从不
帐户过期时间                                                : 从不
两次改变密码之间相距的最小天数          : 7
两次改变密码之间相距的最大天数          : 99999
在密码过期之前警告的天数        : 7
```

小知识： 当你新建用户的时候，用户的home目录下会有一些默认的隐藏文件，这些隐藏文件是在创建用户的时候从`/etc/skel/`中复制过去的。

sudoers

Linux是多用户多任务的操作系统, 共享该系统的用户往往不只一个。出于安全性考虑, 有必要通过useradd创建一些非root用户, 只让它们拥有不完全的权限; 如有必要，再来提升权限执行。

sudo就是来解决这个需求的: 这些非root用户不需要知道root的密码，就可以提权到root，执行一些root才能执行的命令。

```
sudo [选项] [用户名] [命令]
```

sudo 命令执行过程

1. 当用户执行sudo时，系统会主动寻找/etc/sudoers文件，判断该用户是否有执行sudo的权限
2. 确认用户具有可执行sudo的权限后，让用户输入用户自己的密码确认
3. 若密码输入成功，则开始执行sudo后续的命令

赋予用户 sudo 操作的权限

通过useradd添加的用户，并不具备sudo权限。在ubuntu/centos/RockyLinux等系统下, 需要将用户加入admin组或者wheel组或者sudo组。以root用户身份执行如下命令, 将用户加入wheel/admin/sudo组。

```
usermod -a -G wheel <用户名>
```

如果提示wheel组不存在, 则还需要先创建该组

```
groupadd wheel
```

配置文件

sudo 的权限控制可以在 /etc/sudoers 文件中查看到。一般来说，通过 cat /etc/sudoers 指令来查看该文件, 会看到如下几行代码。

```
[root@localhost ~]# egrep -v '^[ ]*$|^#' /etc/sudoers
=====省略=====
root    ALL=(ALL)   ALL
%wheel  ALL=(ALL)   ALL
```

对/etc/sudoers文件进行编辑的代码公式可以概括为

```
授权用户/组 主机=[(切换到哪些用户或组)] [是否需要输入密码验证] 命令1,命令2,...
字段1 字段2 =[(字段3)] [字段4] 字段5
```

凡是[ ]中的内容, 都能省略; 命令和命令之间用,号分隔，字段3、字段4，是可以省略的。

- "字段1"不以%号开头的表示"将要授权的用户"，以%号开头的表示"将要授权的组"。

- "字段2"表示允许登录的主机, ALL表示所有，;如果该字段不为ALL,表示授权用户只能在某些机器上登录本服务器来执行sudo命令
- "字段3"如果省略, 相当于(root:root)，表示可以通过sudo提权到root，如果为(ALL)或者(ALL:ALL), 表示能够提权到(任意用户:任意用户组)。
- "字段4"的可能取值是NOPASSWD:。请注意NOPASSWD后面带有冒号:。表示执行sudo时可以不需要输入密码。
    - 比如:`lucy ALL=(ALL) NOPASSWD: /bin/useradd`表示: 普通用户lucy可以在任何主机上, 通过sudo执行/bin/useradd命令, 并且不需要输入密码
    - 比如:`peter ALL=(ALL) NOPASSWD: ALL`,表示: 普通用户peter可以在任何主机上, 通过sudo执行任何命令, 并且不需要输入密码。
- "字段5"是使用逗号分开一系列命令,这些命令就是授权给用户的操作; ALL表示允许所有操作。命令都是使用绝对路径, 这是为了避免目录下有同名命令被执行，从而造成安全隐患。
    - 如果你将授权写成如下安全性欠妥的格式:`lucy ALL=(ALL) chown,chmod,useradd`那么用户就有可能创建一个他自己的程序, 也命名为userad, 然后放在它的本地路径中, 如此一来他就能够使用root来执行这个"名为useradd的程序"。这是相当危险的!

编辑配置文件

在实践中,去编辑`/etc/sudoers`文件，系统提示我没权限，这是因为`/etc/sudoers`的内容如此敏感，以至于该文件是只读的。所以，编辑该文件前，请确认清楚你知道自己正在做什么。

强烈建议通过`visudo`命令来修改该文件，通过`visudo`修改，如果配置出错，会有提示。

官方文档推荐的做法，不是直接修改`/etc/sudoers`文件，而是将修改写在`/etc/sudoers.d/`目录下的文件中。如果使用这种方式修改sudoers，需要在`/etc/sudoers`文件的最后行，加上`#includedir /etc/sudoers.d`一行(默认已有)。需要注意，这个`#includedir /etc/sudoers.d`中的`#`并不是注释，请勿修改。

sudo 选项

- **-u**：以指定用户或 ID 运行命令(或编辑文件)
- **-l**：显示出自己（执行 sudo 的使用者）的权限
- **-b**：将要执行的指令放在后台执行
- **-i**： 以目标用户身份运行一个登录 shell；可同时指定一条命令。相当于切换到root，不过只需要用户自己的密码即可。

案例演示

- 以管理员身份查看shadow文件

```
[root@localhost ~]# useradd user
[root@localhost ~]# echo 123456 | passwd --stdin user
[root@localhost ~]# usermod -a -G wheel user
[root@localhost ~]# su - user
[user@localhost ~]$ cat /etc/shadow
cat: /etc/shadow: 权限不够
[user@localhost ~]$ sudo -u root cat /etc/shadow
[sudo] user 的密码:
```

```
[user@localhost ~]$ sudo cat /etc/shadow
# sudo -u root用的比较多，可以被精简为sudo
```

- 查看下列示例

```
papi ALL=(root) NOPASSWD: /bin/chown,/usr/sbin/useradd
```

```
* 表示: 用户papi能在所有可能出现的主机上，提权到root下执行`/bin/chown`，不必输入密码；但运行`/usr/sbin/useradd`命令时需要密码
* 在具有sudo操作的用户下，执行`sudo -l`可以查看到该用户被允许和被禁止运行的命令
```

- 查看下列示例

```
papi ALL=/usr/sbin/,/sbin/,!/usr/sbin/fdisk
```

```
* 命令前面加上!号表示取消该命令
* 用户papi在所有可能出现的主机上，能够运行目录/usr/sbin和/sbin下所有的程序，但fdisk除外。
```

- 默认情况下输入一次sudo可以保持15分钟不再要求输入密码，如果想要延长这个时间，可以修改配置文件

```
[root@localhost ~]# visudo
Defaults env_reset,pwfeedback,timestamp_timeout=60
# 这个是改成60分钟才会需要再次输入密码，并且输入密码的时候会显示*号
```