# 用户管理、连接和权限

毛迪林
dlmao@fudan.edu.cn

# 用户和用户组

- 用户通过一个唯一的非负的UID（User ID）标识
  - UID=0的用户为超级用户，一般名字为root，拥有系统的所有权限
  - 许多Linux发行版会限制root用户的登录
  - 系统管理员一般使用自己的普通用户身份登录进行常规的工作，只有在进行需要特殊权限的工作时才切换到超级用户，完成工作后切换回普通用户身份
- 文件系统的文件和目录可以供哪些用户访问？
  - 其属性部分包括拥有者ID以及用户组ID
- 一个用户可以属于多个用户组，用户组通过一个GID标识
- 纯粹的数字记忆起来比较麻烦，用户和组除了对应的ID，同样也包括了一个用户名和组名
  - 用户名和组名也应该保证在系统中唯一
  - 用户名可能按照某些规则进行命名，比方说为你的真实姓名中名字的首字母再加上姓（比如我经常用的用户名dlmao）
  - Linux区分大小写，在实践中一般用户名都是采用小写字母

# 密码

- 不要使用用户名，或者用户名反写
- 不要使用自己的姓名，不要使用爱人或者家人朋友的姓名
- 不要选择一个键盘序列，比如123456，qwerty或1q2w3e4r等
- 不要选择一串对自己有特殊意义的数字，比如电话号码、门牌号、（生日等）重要日期
- 不要选择字典中的单词
- 不要选择Harry Potter、Star Wars等与流行文化相关的密码
- 不要将口令写在一张纸上（但是很可能会忘记）
- 定期改变口令（这点可能很难做到☺）
- 一个好的密码：
  - 你不用写下来就可以记住但其他人不易猜测
  - 比如一个对您有意义的短语或者句子为基础进行适当的转换
    - dontBL8 (Don't be late)
    - 2b||~2b (To be or not to be)

# 用户管理相关文件

```
$ ls -l /etc/{passwd,shadow,group}
-rw-r--r-- 1 root root   1074 Sep 28 09:22 /etc/group
-rw-r--r-- 1 root root   2568 Sep 28 09:22 /etc/passwd
-rw-r----- 1 root shadow 1514 Sep 28 09:22 /etc/shadow
```

- **/etc/passwd里面保存有关用户的相关信息**
- **/etc/group保存了用户组的相关信息**
- **/etc/shadow保存用户密码，该文件只有超级用户可以读写，其他用户无法访问**
- /etc/gshadow保存用户组与安全相关的信息，包括用户组密码等
- /etc/login.defs和/etc/default/useradd： useradd增加用户时的缺省值
- /etc/shells: 合法的shell值
- /etc/skel: 新增用户拷贝到用户主目录的文件
- /etc/adduser.conf:采用adduser来增加用户时的缺省值

getent database [key…]

- 查看系统数据库中对应数据库(文件)中的内容，如果没有key则列出对应文件的所有内容，否则列出其中匹配的项。database可取值passwd/group/shadow/gshadow/hosts等

getent passwd 查看/etc/passwd文件

getent passwd sshd 查看passwd文件中sshd的行

# passwd和shadow文件

7个字段，用:分隔

| 用户名 | 密码 | 用户ID | 组ID | 描述 | 主目录 | 命令行解释器 |
|---|---|---|---|---|---|---|

- 命令行解释器
  - 交互式shell：/etc/shells里面的shell
  - 不允许交互式登录：命令执行的返回值为非0（可以通过echo $?查看），如/usr/sbin/nologin /bin/false等
- 密码：
  - x: 表示密码在shadow文件中
  - 空表示无密码

```
demo@mars:/etc$ less passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd time,,,:/run/systemd:/bin/false
demo:x:1000:1000:demo user,,,:/home/demo:/bin/bash
```

shadow文件: 9个字段，之间以:分隔

| 用户名 | 密码 | 上次密码更改时间(天) | 最小密码age | 最大密码age | 密码将过期警告间隔 | 密码不活跃(仍可用)间隔 | 账号过期时间(天) | 保留 |
|---|---|---|---|---|---|---|---|---|

- 密码字段:第一个字符为!或者*时 不允许通过用户名和密码方式登录
  - 超级用户缺省不允许通过密码登录
- 密码并不是明文保存，而是**用户输入的密码进行变换（加密）后的密码（散列值）**，格式为$id$salt$encrypted。id表示所采用的密码（散列）算法，salt为某随机字符串

encrypted=hash(salt+passwd)

时间单位：距1970年1月1日的天数

```
demo@mars:/etc$ sudo less /etc/shadow
[sudo] password for demo:
root:!:16985:0:99999:7:::
daemon:*:16911:0:99999:7:::
sys:*:16911:0:99999:7:::
demo:$6$/ayKvXjw$wsP/Vaby1qT981/QOJD/v/J2nWLzagV
4isOwJsGSnIeo1JDIfKuWb4m.oroSIgUFsFC7Puw5Bmyh1nP
H.7UDd/:16985:0:99999:7:::
```

普通用户没有权限查看

# group文件/etc/group

| 组名 | 密码 | 组ID | 用户列表，以逗号隔开 |
|---|---|---|---|

group文件: 4个字段，以:分隔

如果使用密码，密码保存在/etc/gshadow中。一般情况下不大用到

- 每个用户有一个**主用户组**，即/etc/passwd中包含的GID给出的组，同时用户也可以属于多个**从(辅助)用户组**，即/etc/group文件中该用户出现在某些用户组对应的用户列表中

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,demo
sudo:x:27:demo
demo:x:1000:
sambashare:x:128:demo
```

- id命令: id [options] [USER] 查看当前用户或者其他用户的用户和组信息
  - -u 显示用户ID
  - -g 显示用户组ID
  - -G 显示所有用户组ID
  - -n 显示名字而不是数字
  - id –un 等价于whoami
- groups命令: groups [USER] 查看当前用户或其他用户所在的组。显示的组中，第一个为主用户组，后面为从组

```
demo@mars:~$ id
uid=1000(demo) gid=1000(demo) groups=1000(demo), 4(adm),24(cdrom), 27(sudo), 30(dip), 44(video),46(plugdev),113(lpadmin),128(sambashare)
 demo@mars:~$ id root
uid=0(root) gid=0(root) groups=0(root)
demo@mars:~$ groups dlmao
dlmao : dlmao adm cdrom sudo dip plugdev
```

# 用户管理命令: useradd要求超级用户权限

- 用户主要的信息为：用户名、主目录、Shell和用户组，设置登录密码
  - 添加新用户：**useradd [options] LOGIN**
    - 如果没有指定，根据缺省配置为用户分配一个唯一的UID、主目录和Shell。注意缺省并不会创建用户主目录
    - 用户的缺省配置文件为/etc/login.defs和/etc/default/useradd，且login.def优先
    - 许多Linux发行版缺省情况下在创建用户的同时也会创建一个相同名字的group，并且为其分配一个组ID
    - 建议采用选项-m(创建用户主目录，并拷贝/etc/skel目录中的文件如.bashrc、.profile等到该目录)和-s SHELL(设置用户所使用的shell)

**useradd -m -s /bin/bash test1** 添加用户test1，创建用户主目录并设置采用bash

```
demo@mars:/etc$ sudo useradd -m -s /bin/bash test1
demo@mars:/etc$ sudo grep test1 /etc/{passwd,shadow,group}
/etc/passwd:test1:x:1002:1002::/home/test1:/bin/bash
/etc/shadow:test1:!:17032:0:99999:7:::
/etc/group:test1:x:1002:
```

- **passwd test1** 为该用户设置一个新的密码

```
demo@mars:/etc$ sudo passwd test1
[sudo] password for demo:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

# 用户管理命令: useradd userdel

- 用户属于哪些用户组的信息也至关重要
  - 比如sudo组中的用户相当于管理员用户，可以通过sudo获得超级用户权限
  - -g 选项可指定用户的主组(如果不指定，缺省创建一个新的与用户名一样的主组），而-G选项指定用户的从组

**useradd [options] LOGIN**
**-u UID 必须唯一，不指定时系统自动指定**
**-d HOME_DIR 缺省从/etc/default/userdd 给出的HOME变量产生，即$HOME/LOGIN，/etc/skel下的内容会复制到用户主目录**
**-c COMMENT 用户描述**
**-e YYYY-MM-DD 账号过期时间**
**-g GROUP 用户的组ID或者组名**
**-G GROUP1[,GROUP2,…[,GROUPN[] 该用户还属于那些组**

```
demo@mars:/etc$ sudo useradd –m –G sudo  -s /bin/bash admin
# 添加一个管理员用户admin，该用户属于sudo组中
```

- 删除用户，缺省情况下不删除用户主目录

userdel [options] test1
-r 删除用户主目录下的所有文件以及主目录本身

# 用户管理命令： usermod

- 已有用户的信息也可以修改

**usermod [options] username**

**-a 将用户加入到其他组(通过-G选项指定)**

**-G GROUP1[,GROUP2,…[,GROUPN[]** 该用户属于那些组，如果没有-a选项，则从那些不在指定的组里面的组中移走

-g group 修改用户组
-l NEW_LOGIN 修改用户名
-d home 修改用户主目录
-L 锁住该用户的口令
-U 解锁用户的口令

```
$ grep test1 /etc/group
adm:x:4:syslog,demo,test1
sudo:x:27:demo,test1
test1:x:1002:
$ sudo usermod -G video test1  # 只在video组中
$ grep test1 /etc/group
video:x:44:demo,test1
test1:x:1002:
$ sudo usermod -G adm,sudo test1 # 只在adm和sudo组中
$ groupadd students
# 添加一个新的group students
$ sudo usermod  -a -G sambashare, students  test1
$ grep test1 /etc/group
adm:x:4:syslog,demo,test1
sudo:x:27:demo,test1
sambashare:x:128:demo,test1
test1:x:1002:
students:x:1003:test1
```

# 用户管理命令: passwd/chsh/chfn

- passwd [USER] 改变当前用户或其他用户的密码
- chsh [USER]改变当前用户或其他用户的登录shell
- chfn [USER]改变当前用户或其他用户的全名
- 改变自己的密码/shell/全名并不需要超级用户权限

demo@mars: passwd [options] [LOGIN] 后面不提供用户名时表示对当前用户操作
-d 密码清空，不需要密码可登录
-e 密码过期，用户下次登陆后必须修改密码
**-l 锁住密码，无法通过密码登录　　-u 解锁密码**

**demo@mars:/etc$ chsh　　#改变shell**
Password:
Changing the login shell for demo
Enter the new value, or press ENTER for the default
　　Login Shell [/bin/bash]:

# adduser, deluser, addgroup,delgroup等

- 更加高端方便的命令，采用perl脚本语言编写，交互式

adduser [options] user 增加新用户
**adduser [options] user group 将用户加入到组**
deluser [options] user 删除用户
deluser [options] user group 将用户user从组中移走
addgroup [options] group 增加组
delgroup [options] group 删除组

```
demo@mars:/etc$ sudo adduser demo2
Adding user `demo2' ...
Adding new group `demo2' (1003) ...
Adding new user `demo2' (1004) with group `demo2' ...
Creating home directory `/home/demo2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for demo2
Enter the new value, or press ENTER for the default
        Full Name []:…
Is the information correct? [Y/n] y
```

```
$ sudo adduser demo2 students
Adding user `demo2' to group `students' ...
Adding user demo2 to group students
Done.

$ sudo usermod –a –G students demo2
```

# su（substitute userid）

- 变成另外一个用户（不指定用户名时变为超级用户）
  - 需要**输入要变成的用户的密码**，用另一个用户身份开启新会话
  - 退出时返回原会话
  - 当前用户已经是超级用户时，在切换用户时不需要输入密码

**su [options] [username] 如果没有用户名，则表示变成超级用户root**

-c command 给出在切换用户成功后传递给shell的参数(-c command)，这样以另一个身份执行指定的command，执行完后结束

-m,-p 缺省选项，表示保留当前用户的环境，除了HOME，SHELL，USER，LOGNAME，PATH和IFS。IFS重置为缺省值，而PATH根据切换后的用户身份（普通用户和超级用户）不同而设置一个缺省的值

-l, --login, - 切换成功后的shell为登录shell，当前用户的环境变量在切换后除了TERM、DISPLAY等环境变量外都被重置，如果采用-选项，应该为最后一个选项

```
demo@mars:~$ export MYENV=/home/demo
demo@mars:~$ echo $MYENV
/home/demo
demo@mars:~$ su test2   # 缺省传递当前用户的环境变量
Password:
test2@mars:/home/demo$ echo $MYENV
/home/demo
test2@mars:/home/demo$ cd
test2@mars:~$ exit      # 退出当前shell返回到su之前的shell
exit
```

**su - test2或su –l test2 全新的环境变量**

# sudo 以另一个用户身份执行程序

- sudo允许以另外一个用户（包括超级用户）身份执行一个命令（甚至包括su)
  - 要求用户输入的是**当前用户的密码**，而不是超级用户的密码
  - sudo执行一个命令后，在接下来的一段时间（缺省15分钟）下次sudo时不用再输入密码
  - 配置文件/etc/sudoers（该文件仅允许root访问）决定了允许使用sudo的用户以及用户所拥有的权限，建议采用visudo命令来修改，会加锁/etc/sudoers文件，同时可以检查语法错误等

**sudo [options] [command]**
-e 以另一个用户身份编辑一个或者多个文件，等价于调用sudoedit
**-u user 以user身份执行，缺省为root**
-g group 以主用户组为group(名字或者#gid) 的身份执行
**-s 以另一个用户身份执行shell(当前环境变量SHELL指定的shell)**
**-i 与-s类似，只是为login shell**
-l 列出用户（当前用户或者-U user指出的用户）可以执行的命令
-K 将用户信任缓存清除，下次调用sudo将要求输入当前用户的密码
-- 表示选项部分结束，后面为command或文件

考虑到安全因素，许多发行版不允许root身份密码登录
- 可以通过sudo –s 或者sudo -i 以root身份运行shell
- 或者sudo su以root身份执行su命令来切换成超级用户

# sudo: 配置文件

**demo@mars:~$ cat /etc/sudoers**
cat: /etc/sudoers: Permission denied

- /etc/sudoers给出了相关配置，建议 sudo visudo来编辑

```
demo@mars:~$ sudo cat /etc/sudoers    # man sudoers查看帮助
# This file MUST be edited with the 'visudo' command as root.
# 授权规则： 允许来自于哪些主机的哪些用户切换到哪些用户或用户组来使用哪些命令
# 格式：  授权用户  主机=[(切换到哪些用户或用户组)]  命令1,命令2,…命令N
# 授权用户： user 或者#uid表示用户名为user或者用户ID为uid的用户
#          %grp或%#gid 表示组名为grp或者组ID为gid的组的成员

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
#includedir /etc/sudoers.d
```

- 不在sudo或admin组中的用户无法sudo

```
$ id test2
uid=1003(test2)gid=1003(test2) groups=1003(test2)
$ sudo -l -U test2
#查看test2使用sudo允许执行的命令
User test2 is not allowed to run sudo on mars.
```

# sudo 例子

- 不在sudo或admin组中的用户无法sudo

```
$ sudo adduser test2 sudo
# sudo usermod –a –G sudo test2
#将test2添加到sudo组后就可以使用sudo来执行命令了
Adding user `test2' to group `sudo' ...
Adding user test2 to group sudo
Done.
demo@mars:~$ id test2
uid=1003(test2) gid=1003(test2) groups=1003(test2),27(sudo)
```

```
$ sudo shutdown -r +15 "quick reboot"
# 超级用户身份15分钟内重启
Shutdown scheduled for Sun 2016-08-21 18:18:07 CST, use 'shutdown -c' to cancel.
$ sudo shutdown –c
# 超级用户身份取消刚才关机或重启命令
$ sudo sh -c "cd /home; du -s * | sort -rn > USAGE; cat /home/USAGE"
# 超级用户身份，通过标准shell(sh)执行命令
95660   demo
28      test1
28      demo2
```

# 文件系统基础

- Linux的文件可以分为四种类型
  - 普通文件：文本文件和二进制文件
    - 二进制文件可通过od、xxd(建议), hexdump命令查看
  - 目录:维护了如何组织和访问其他文件以及子目录的信息
  - 符号连接：连接到其他文件或者目录
  - 伪文件(pseudo file): 设备文件、Unix Socket文件、命名管道和proc文件等
    - 没有使用数据块来存储数据，文件本身不占用数据空间
    - 仍然按照目录进行组织，通过目录项中的名字来保存设备文件名，通过inode节点来保存伪文件的属性
    - 设备文件是物理设备的内部表示，设备文件的inode保存了其设备属性（设备号），内核根据设备号调用相应的模块或函数进行设备的实际IO操作
    - Unix Socket文件类似于网络通信时的Socket，采用C/S模式，提供同一台主机上的客户和服务方进程之间的通信。
    - 命名管道提供了进程间通信的机制，能够将一个程序的输出连接到另一个程序的输入上
    - proc文件提供了对于Linux内核中的相关信息的访问

# xxd

将二进制形式的文件以十六进制方式显示(hexdump)。 infile和outfile都是可选，如果没有，则表示标准输入或者标准输出

xxd [options] [infile [outfile]]

xxd -r[evert] [options] [infile [outfile]]

-c cols : 每行输出多少个字节(octet)，一般缺省为16

-l len: 输出多少个字节

-o offset: 输出时对应地址再加上这个偏移量

-s [+][-]seek 表示不是从头开始而是seek个字节之后开始转换输出，+表示相对位置，即当前标准输入位置之后多少个字节，-表示文件结尾之前多少个字节

-r 表示将hexdump文件转换回原来的二进制形式

**vim –b file**
**:%!xxd 替换为hexdump格式**

**修改后转换为二进制格式**
**:%!xxd –r**

```
[15:24]demo@mars:/usr/bin$ xxd -l 512  -c 16 /usr/bin/clear
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0200 3e00 0100 0000 4007 4000 0000 0000  ..>.....@.@.....
00000020: 4000 0000 0000 0000 5011 0000 0000 0000  @.......P.......
00000030: 0000 0000 4000 3800 0900 4000 1c00 1b00  ....@.8...@.....
00000040: 0600 0000 0500 0000 4000 0000 0000 0000  ........@.......
00000050: 4000 4000 0000 0000 4000 4000 0000 0000  @.@.....@.@.....
00000060: f801 0000 0000 0000 f801 0000 0000 0000  ................
00000070: 0800 0000 0000 0000 0300 0000 0400 0000  ................
00000080: 3802 0000 0000 0000 3802 4000 0000 0000  8.......8.@.....
00000090: 3802 4000 0000 0000 1c00 0000 0000 0000  8.@.............
000000a0: 1c00 0000 0000 0000 0100 0000 0000 0000  ................
000000b0: 0100 0000 0500 0000 0000 0000 0000 0000  ................
000000c0: 0000 4000 0000 0000 0000 4000 0000 0000  ..@.......@.....
```

地址　　十六进制表示的字节内容

对应的ASCII字符，不可打印字符以点(.)表示

```
xxd –s 0x30 /usr/bin/clear
# 前面3*16个字节（相当于前面3行)跳过
xxd –s -0x30 /usr/bin/clear
# 最后3*16个字节(后面3行)
xxd –l 512 –c 16 /usr/bin/clear
# 每行16字节，输出512字节(32行)
echo "0000037: 3574 68" | xxd -r -  file
修改file中对应地址(0x0000037)开始的3个字节的内容
```

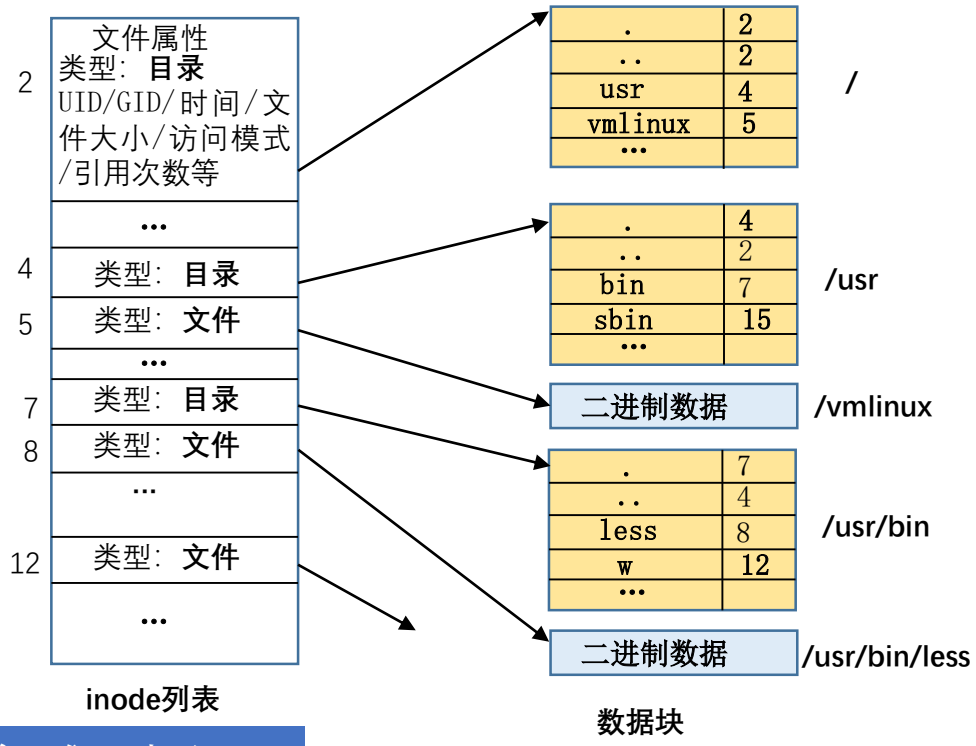
# 文件系统

- 注意：**文件的文件名并不在inode节点中，而是保存在包含该文件的目录文件的数据块中**
- 存储设备上的文件系统的根目录文件的inode编号缺省为2

- 文件系统的组织：
  - 保存文件相关的元信息的**索引节点inode**(index node)
    - 拥有者UID和GID
    - 文件长度
    - 类型和权限
    - 访问、修改和状态改变时间
    - 引用次数
    - **数据块编号列表**
  - 实际存储数据的数据块
    - 对于普通文件，数据块保存实际的内容
    - 对于目录文件，数据块保存的是该目录包含的子目录和文件所对应的**目录项**

许多文件系统可能会在inode也存放少部分数据

- 每个目录文件包含了多个目录项
- **每个目录项保存了其对应的文件（包括目录文件在内）的名字以及该文件对应的inode编号**
- ls –i 选项可以查看inode编号

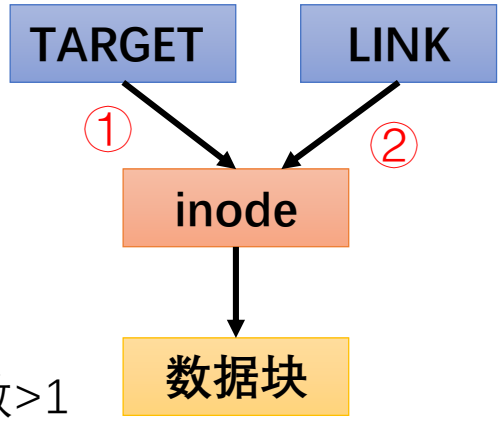



文件和目录等通过inode编号(而不是文件名)唯一标识

# 硬连接(Hard Link)

- 为什么文件名不保存在inode里面，而是保存在**包含该文件的目录文件的数据块**(以目录项的形式组织）里面呢?
  - 文件名的长度可变，最长255个字符。inode要分配多少空间就成为问题
  - **文件的唯一标识符是其inode编号**，而不是文件的名字
- 同一个文件系统中允许多个**目录项指向同一个inode，**即建立了一个到inode节点所标识的文件的**硬连接（Hard Link）**
  - **每个目录项保存了其对应的文件（包括目录文件在内）的名字以及该文件对应的inode编号**
  - 一个文件可以有多个名字(pathname)，从根目录开始到指向该文件的目录项中的名字，如/usr/bin/ls
  - inode通过**引用计数**记录指向它的硬连接的个数，引用计数为0时，inode被回收
  - 硬连接不允许跨越文件系统：由于通过目录项指向同一个inode来实现，两个不同文件系统的inode编号属于不同的地址空间
  - 每个目录至少存在两个硬连接，即表示当前目录和父目录的“.”和“..”
  - 硬连接缺省不能对目录进行创建，只可对文件创建
    - 如果允许对目录创建硬连接，可能导致目录回路

# 为什么要引入Hard Link




**ln TARGET LINK**
创建一个到TARGET的硬连接，该硬连接的文件名为LINK
两者必须在同一个文件系统，且TARGET一般不能是目录

**find -type f -links +1 2>/dev/null**
查找当前目录的分支中的硬链接： 类型为普通文件，且连接数>1

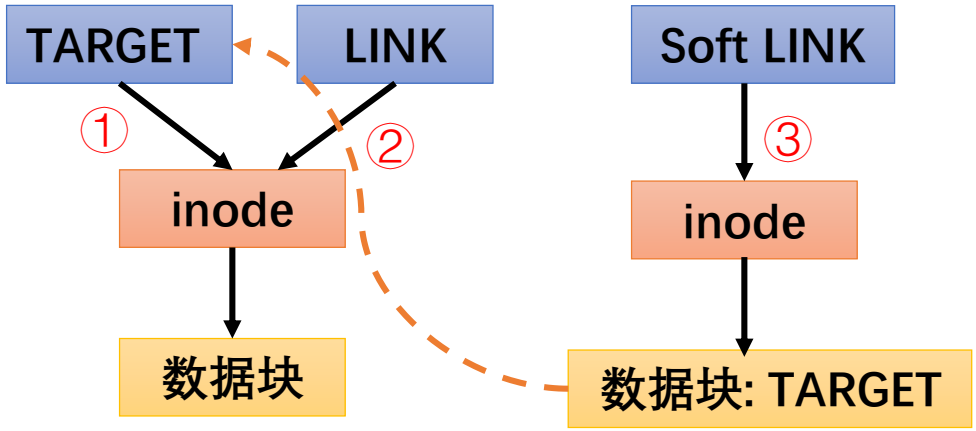
- 同一个文件可以有多个名字，根据名字的不同可能有不同的含义，
  - 比如同一个执行程序有多个名字，当以其中一个名字运行时，完成某项工作，而以另外一个名字运行时，完成另一项工作
- 用户可能觉察不到硬连接的存在，不过在通过其中一个名字来修改文件内容或者改变元属性时，另一个名字访问时可以看到变动

```
$ touch hosts
$ ln hosts hosts.lnk
$ ls -li hosts*
20322720 -rw-rw-r-- 2 dlmao dlmao 438 May 14 14:12 hosts
20322720 -rw-rw-r-- 2 dlmao dlmao 438 May 14 14:12 hosts.lnk
$ mkdir dir-a
$ ln dir-a d
ln: dir-a: hard link not allowed for directory
```

# 符号连接（symbolic link），也称为软连接

- 硬连接通过目录项来实现，而符号连接通过inode来实现,类似于windows的快捷方式
- 分配一个新的inode，其文件类型为**符号连接(l)**，同时其所指向的数据块存放的内容是要指向的另一个文件的路径名，即符号链接指向另外一个文件

创建符号连接 **ln –s TARGET LINK_NAME**




- 软连接允许跨越文件系统，可对文件或目录创建软连接，可以对不存在的文件或目录创建软连接，删除软链接并不影响被指向的文件
- 软连接可以有自己的文件属性及权限控制，但注意软连接在访问时并不检查权限等属性

```
$ ln -s hosts hosts.slnk
$ ls -li hosts*
20322720 -rw-rw-r-- 2 dlmao dlmao 438 May 14 14:12 hosts
20322720 -rw-rw-r-- 2 dlmao dlmao 438 May 14 14:12 hosts.lnk
20319422 lrwxrwxrwx 1 dlmao dlmao   5 May 14 14:55 hosts.slnk -> hosts
$ ln -s /usr/lib/python3.6 python3.6
$ ls -li python3.6
20319426 lrwxrwxrwx 1 dlmao dlmao 18 May 14 14:56 python3.6 -> /usr/lib/python3.6
```

# 创建连接命令ln

- ln [OPTION]... [-T] TARGET LINK_NAME　创建连接LINK_NAME，它指向TARGET
- ln [OPTION]... TARGET　等价于 ln -s TARGET ./$(basename TARGET)，在当前目录创建一个到TARGET的连接，该连接的名字为TARGET的最后文件名，比如ln –s /etc/hosts
- ln [OPTION]... TARGET... DIRECTORY
- ln [OPTION]... -t DIRECTORY TARGET...　在目录DIRECTORY中为每个TARGET创建一个连接，连接名为TARGET的最后文件名
- -s, --symbolic 创建符号连接，缺省为硬连接
- -r, --relative 符号连接中保存相对路径，缺省情况下符号连接中的内容为前面给出的TARGET，-r选项表示将TARGET转变为相对路径名

创建符号连接时，TARGET可以采用绝对路径或带路径的相对路径名。采用绝对路径名，符号链接可移动位置。采用相对路径名，TARGET和符号链接可整体搬移

```
$ ln -sr  /etc/passwd passwd
$ ls -l
lrwxrwxrwx 1 dlmao dlmao   25 Nov 13 20:54 passwd -> ../../../../../etc/passwd
$ mkdir dir-a && cd dir-a
$ ln -s ../hosts ../passwd .
$ ls –l
lrwxrwxrwx 1 dlmao dlmao 8 Nov 13 21:50 hosts -> ../hosts
lrwxrwxrwx 1 dlmao dlmao 9 Nov 13 21:50 passwd -> ../passwd
```

# 符号连接展开: readlink/realpath

man path_resolution

- 大多数命令(比如cp)在对符号连接进行操作时(如打开读写等)，实际上会自动展开(dereference)连接，操作实际的TARGET，如果第一层展开仍是符号连接，继续展开直到非符号连接为止。这些命令常有如下选项:
  - -L, --dereference 展开符号连接
  - -P, --no-dereference 不展开符号连接
- 删除(rm)、移动(mv)等命令不展开，而是对于符号连接进行操作
- ls/file/stat命令主要检查保存在inode节点的元信息，缺省不展开符号连接，但可通过-L选项来展开符号连接
- 在GNU系统中，符号连接的权限无法修改，也不起作用 (chmod会展开符号连接)，用户身份也只有在其所在目录具有sticky权限才有意义。参见后面的权限部分

```
$ ls -l hos*
lrwxrwxrwx 1 dlmao dlmao 10 May 14 16:04 hosts -> /etc/hosts
lrwxrwxrwx 1 dlmao dlmao  5 May 14 16:04 hosts.slnk -> hosts
```

- readlink [OPTION]... FILE 读取符号连接FILE中的内容，-f选项会一直展开符号连接直到最终的TARGET为止
- realpath [OPTION]... FILE 将路径名转换为绝对路径名，会展开符号连接

```
$ readlink hosts.slnk
hosts
$ readlink $(readlink hosts)
/etc/hosts
$ readlink -f hosts
/etc/hosts
$ realpath hosts
/etc/hosts
```

# 查看文件状态命令stat

- stat [OPTION]... FILE... 显示文件或所在文件系统的状态信息
  - 标识该文件的拥有者身份的UID(%u)和GID(%g)
  - 文件的长度、文件的类型和访问模式、引用次数等
  - 访问时间(读取文件内容时更新)、修改时间（修改文件内容时更新）、状态改变时间（文件内容改变或者chmod、chown等改变文件属性时更新）。如果为设备文件，还包括设备号等
- -f, --file-system 选项，显示文件所在的文件系统的状态信息，包括文件系统类型，数据块和inode节点的使用情况等
- -L, --dereference: 展开符号连接，查看连接到的文件
- -c FORMAT 按照格式中指出的输出相应的状态信息 --printf=FORMAT 与c类似，但支持转义(比如包括\n)
- -t, --terse 简洁方式，方便进一步分析

| %格式 | 含义 |
| --- | --- |
| n | 文件名(name) |
| u/U | 拥有者的UID和用户ID |
| g/G | 文件的用户组ID和用户组名 |
| s | 文件长度（size) |
| f/F | 文件类型, f采用十六进制数字描述 |

| | |
| --- | --- |
| a/A | 访问模式的数字和符号形式 |
| h | 引用(hard links)个数 |
| i | inode编号 |
| xX/yY/zZ | 文件的访问时间atime、修改时间mtime和状态改变时间ctime的友好输出或距epoch的秒数 |
| m | 加载点(mount) |
| t | 设备号 |

分配8个block(每个512字节)

```
demo@mars:~$ ls > contents
demo@mars:~$ stat contents
  File: 'contents'
  Size: 145             Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 293447      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    demo)   Gid: ( 1000/    demo)
Access: 2016-08-31 18:34:54.610030001 +0800
Modify: 2016-08-31 18:34:54.610030001 +0800
Change: 2016-08-31 18:34:54.610030001 +0800
 Birth: -
demo@mars:~$ stat /
  File: '/'
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 801h/2049d      Inode: 2           Links: 24
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-08-30 20:26:54.409161000 +0800
Modify: 2016-08-30 20:26:54.317161000 +0800
Change: 2016-08-30 20:26:54.317161000 +0800
 Birth: -

demo@mars:~$ stat /dev/sda2
  File: '/dev/sda2'
  Size: 0               Blocks: 0          IO Block: 4096   block special file
Device: 6h/6d   Inode: 366         Links: 1     Device type: 8,2
Access: (0660/brw-rw----)  Uid: (    0/    root)   Gid: (    6/    disk)
Access: 2016-08-30 20:27:08.369161000 +0800
Modify: 2016-08-28 06:40:21.528000000 +0800
Change: 2016-08-28 06:40:21.528000000 +0800
 Birth: -
```

```
demo@mars:~$ stat /dev/tty
  File: '/dev/tty'
  Size: 0               Blocks: 0          IO Block: 4096   character special file
Device: 6h/6d   Inode: 13          Links: 1     Device type: 5,0
Access: (0666/crw-rw-rw-)  Uid: (    0/    root)   Gid: (    5/     tty)
Access: 2016-08-31 18:39:07.588000000 +0800
Modify: 2016-08-30 16:21:05.588000000 +0800
Change: 2016-08-28 06:40:21.588000000 +0800
 Birth: -
demo@mars:~/linux_course$ stat tmpfile3.txt
  File: 'tmpfile3.txt' -> 'tmpfile.txt'
  Size: 11              Blocks: 0          IO Block: 4096   symbolic link
Device: 801h/2049d      Inode: 398277      Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/    demo)   Gid: ( 1000/    demo)
Access: 2016-10-10 13:37:33.367430657 +0800
Modify: 2016-10-10 13:37:28.841168654 +0800
Change: 2016-10-10 13:37:28.841168654 +0800
 Birth: -
demo@mars:~$ stat -f ~
  File: "/home/demo"
    ID: c9ccd6b97e761189 Namelen: 255     Type: ext2/ext3
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 4611519    Free: 3135511    Available: 2895499
Inodes: Total: 1179648    Free: 834123

demo@mars:~$ stat -c '%i %n %a' /usr/bin/* | less
917744 /usr/bin/[ 755
917735 /usr/bin/2to3 777
917736 /usr/bin/2to3-2.7 755
...
```

HOME目录所在文件系统信息

inode 文件名 访问模式(数字)

# 文件和目录权限

- i-node节点中包含了访问控制相关信息，这些信息是文件创建者的UID和GID，另外还包括访问模式（mode）
- Linux系统将要访问该文件的用户分成3种类型，分别是用户(user)、组用户(group)和其他用户(other)
- 访问模式包括了对于每种类型的用户所使用的权限，分别是读(read)、写(write)和执行(execute)权限。
- 通过ls的-l选项或者stat命令可以查看文件的访问模式。相应字段的最后9个字符表示相关权限，分别是用户、用户组和其他用户的权限
- 3个字符如果某个位置为-字符，表示不支持该位置对应的访问权限。rw-r--r--表示允许用户读和写，组用户读，其他用户读。

```
$ ls -ld . 1.txt
drwxrwxr-x 2 demo demo 4096 Oct 23 17:16 .
-rw-rw-r-- 1 demo demo    0 Oct 23 17:16 1.txt
```

rwxrwxrwx

user group other

# 如何进行权限判别

- 某个用户（进程）要访问某个文件时，系统会从i-node节点读取相关信息，然后决定该用户的类型
  - 如果其有效UID与文件的拥有者UID相同，则该用户属于用户类型
  - 如果不属于用户类型，判断该用户的有效GID是否在文件的GID属性给出的用户组中，如果在，则该用户属于组类型
  - 否则属于其他用户类型
  - 尽管模式包含了9个权限设置，但是真正起作用的是其中某3个权限
- 文件和目录的访问模式给出的权限指的是对于包含其具体内容的数据块操作的权限
  - 文件和目录都对应着一个i-node节点，通过该节点唯一标识，而文件和目录的内容则保存在i-node节点所指出的数据块中
  - 文件的访问控制不仅仅与该文件本身的权限相关，也与各级子目录的权限有关
    - 要求用户对于该文件的pathname中的各级目录都有x权限

# 如何进行权限判别

- 目录权限：
  - 搜索权限x表示可以切换到该子目录
  - 读权限r表示可读取该子目录中的目录项，即可以列目录
  - 写权限(w)，表示可以更改其目录项，也就是说可以创建文件、删除文件、移动或改名文件，至于文件内容的更改（即修改文件）则由该文件本身的权限决定。

Q1: 如果文件只读，但是其所在目录可写，能否修改该文件?
Q2: 一个目录仅仅有x权限，但是没有r权限，有什么意义?

| 属性 | 文件 | 目录 |
| --- | --- | --- |
| r | 文件可以读 | 在x权限也打开的情况下允许查看目录中的内容(列目录)，仅仅r属性不够 |
| w | 允许文件修改、截取。但是本属性并不允许文件改名或删除，要求其所在目录有w权限才允许 | 在x权限也打开时允许对于目录项进行操作，包括创建、删除和改名。即便该文件本身没有w权限，但是仍然可以删除、改名(会询问用户是否删除，除非采用-f选项) |
| x | 允许执行。如果该文件为脚本，由于采用解释执行，需要同时有r权限才允许执行 | 允许搜索进入该目录，比如通过cd命令切换到该目录 |

# 改变文件的拥有者chown和chgrp

- 改变文件的身份可能导致获得新的权限。chown和chgrp一般要求超级用户的权限

chown [OPTION]... [OWNER][:[GROUP]] FILE...

改变文件的拥有者或者所属用户组，或者两者都同时改变

常用的选项主要包括:

-R, --recursive 递归地调用chown命令改变目录及各级子目录中文件和目录的拥有者身份

```
sudo chown tony 1.txt     # 改变1.txt的拥有者为tony
sudo chown tony:wheel 1.txt # 改变1.txt的拥有者为tony，所属用户组为wheel
sudo chown tony:  1.txt   # 改变1.txt的拥有者为tony，所属用户组为tony的主用户组
sudo chown :wheel  1.txt    # 改变1.txt所属用户组为wheel
```

chgrp [OPTIONS] GROUP  FILE
仅仅改变用户组，等价于chown :GROUP FILE

# 改变文件访问模式chmod

- 只有文件的拥有者或者超级用户才允许更改访问模式

chmod [OPTION]... MODE[,MODE]... FILE...　　符号模式

chmod [OPTION]... OCTAL-MODE FILE...　　八进制数字模式

第一种格式：可包含多个符号模式，中间以逗号,分割

chmod ug=rw,o=r 1.txt 表示更改权限：用户和组有rw,其他有r权限

改变权限的对象　如何改变　权限

| 对象 | 含义 | 权限改变 | 含义 |
|---|---|---|---|
| u(user) | 文件或目录的拥有者的权限 | + | 增加某些权限 |
| g(group) | 文件或目录所属用户组的权限 | - | 去除某些权限 |
| o(other) | 其他用户的权限 | = | 设置某些权限，未指定权限去除 |
| a(all) | ugo三者的权限。不指定特定的对象时隐含为a | | |

| 符号 | 含义 |
|---|---|
| u-x | 文件拥有者删除执行权限 |
| +x | 等价于a+x，文件拥有者、用户组和其他增加执行权限 |
| o-rw | 删除文件拥有者的读写权限 |
| go=rw | 设置用户组和其他的权限为可读写 |
| u+x,go=rx | 文件拥有者增加执行权限，用户组和其他权限为读和执行权限 |

# 特殊权限

- setuid在执行程序时设置有效用户ID
- setgid在执行时设置有效用户组ID
- sticky权限：限制删除权限

- chmod仍然通过3组9个字符来描述权限（包括特殊权限）：
  - 用户的执行权限也可用来显示setuid权限
  - 用户组的执行权限也可用来显示setgid权限
  - 其他组的执行权限也可用来显示sticky权限

每组权限中第三个字符

| 字符 | 含义 |
|---|---|
| - | 表示没有执行和特殊权限 |
| x | 仅有执行权限 |
| S | 有setuid或setgid权限 |
| **s** | **有setuid或setgid权限再加上执行权限** |
| T | 表示sticky权限 |
| **t** | **表示sticky权限加上执行权限** |

* chmod u+s FILE 可设置setuid权限
* chmod g+s FILE 可设置setgid权限
* chmod o+t FILE 可设置sticky权限

chmod u+sx FILE 文件拥有者增加执行权限和setuid权限

```
$ ls -ld /usr/bin/{passwd,crontab} /tmp
-rwxr-sr-x 1 root crontab 39352 Nov 16 13:29 /usr/bin/crontab
-rwsr-xr-x 1 root root    54256 Mar 29  2016 /usr/bin/passwd
drwxrwxrwt 14 root root   4096  May 15 13:20 /tmp
```

# 特殊权限

- setuid在执行程序时设置有效用户ID
- setgid在执行时设置有效用户组ID
- sticky权限：限制删除权限

- 每个执行的进程维护了三个用户ID和组ID
  - 真实用户ID：调用该程序时当前用户的ID，谁为正在运行的进程负责
  - 有效用户ID：用于为新创建的文件分配权限、检查文件访问许可，以及发送信号的权限控制等
  - 保存的setuid: 改变有效用户ID时记录原来的有效用户ID
- 超级用户可以调用setuid函数等改变真实用户和有效用户ID
- **普通用户执行setuid权限的程序时进程的有效用户ID设置为该程序的拥有者ID**
- **setgid与setuid类似**，执行程序时该进程的有效用户组ID设置为程序文件的**用户组ID**，从而拥有文件所属用户组的权限

注意：许多Linux操作系统对可以执行的shell脚本设置setuid和setgid而获得的权限会加以限制，不改变有效用户ID和组ID

* chmod u+s FILE 可设置setuid权限
* chmod g+s FILE 可设置setgid权限

```
$ ls -l /usr/bin/{passwd,crontab}
-rwxr-sr-x 1 root crontab 39352 Nov 16 13:29 /usr/bin/crontab
-rwsr-xr-x 1 root root    54256 Mar 29  2016 /usr/bin/passwd
```

执行时有效用户ID设置为passwd的拥有者root，从而可读写/etc/shadow文件

# 特殊权限: 目录的setgid权限

- 目录也可以设置setgid权限： **chmod g+s directory**
  - 该目录下**新创建**文件或目录的**用户组属性设置为该目录的用户组gid**
  - **新创建的目录同样也拥有setgid权限**
  - 一个共享目录可更改用户组属性为某个用户组(group)，同时设置SETGID权限
    - 属于group中的用户在该目录中新建文件和目录时，这些文件的用户组都相同
    - group中的用户访问共享目录中的文件时可根据用户组的权限来进行控制。比如可设置为：
      - 新建的文件，组可读写，这样用户都可以阅读和修改其他用户的文件
      - 新建的文件，组可读，这样用户可以阅读其他用户的文件

# 特殊权限: 目录的setgid权限

| | |
|---|---|
| sudo groupadd shared | 创建一个新的group shared |
| sudo usermod –a –G shared demo | 将demo用户加入到shared组中 |
| mkdir sgid-dir; chmod g+s sgid-dir | 创建一个新目录，并且设置sgid权限 |
| sudo chown :shared sgid-dir | 改变该目录的用户组为shared |
| chmod u=rwx,g+rwx,o-rwx sgid-dir | 改变权限,用户和用户组拥有所有权限，其他用户没有权限 |
| umask 007 | 用户创建的新文件和目录，其他用户无权限，用户和用户组拥有所有权限 |
| cd sgid-dir; mkdir dir; touch 1.txt; ls -al | 创建新文件和目录， 查看目录的相关信息，以后shared组中的用户可以在该子目录创建、修改和删除文件和目录，且其用户组为shared |

```
drwxrws---  3 demo shared 4096 Nov 20 21:34 .
drwxrwxr-x 10 demo demo   4096 Nov 20 21:18 ..
-rw-rw----  1 demo shared    0 Nov 20 21:34 1.txt
drwxrws---  2 demo shared 4096 Nov 20 21:34 dir
```

# 特殊权限: sticky

* chmod o+t FILE 可设置sticky权限

- sticky权限在早期用于执行文件，表示执行该文件的进程退出后仍然在交换分区中保留，这样再次执行该文件时可直接从交换分区中加载，以提高经常执行的程序的加载速度。但现在已经不再使用
- sticky权限目前只对目录有效，该权限也称为限制删除权限
- 目录设置了sticky权限时，其目录下的文件或目录的删除以及改名受到限制
  - 即便从该文件所在目录权限设置(比如设置为可写）来看允许删除或改名时，也不允许执行该操作
  - 只有该文件的拥有者、其所在目录（设置sticky权限）的拥有者或超级用户才有可能删除或改名
- 常用于控制对于共享目录（比如临时文件目录/tmp）的访问
  - 该目录一般允许任何人进行读写，但是一旦创建了文件或目录，只有超级用户、创建者或共享目录的拥有者才可以删除
  - 其他用户尽管从目录的权限设置来看允许删除（任何人都可以读写），但是无法删除另外一个用户的文件

```
$ ls -ld /tmp
drwxrwxrwt 14 root root   4096  May 15 13:20 /tmp
```

# 改变文件访问模式chmod：数字模式

chmod [OPTION]... MODE[,MODE]... FILE...　　符号模式

chmod [OPTION]... OCTAL-MODE FILE...　　八进制数字模式

- 符号模式可以增加、删除和设置某些权限
- 数字模式的chmod只能一次设置所有权限
- 访问模式通过3个或4个八进制数字描述
  - 后面3个八进制数字分别对应用户、用户组和其他用户的访问权限
  - 4位数字时第一个数字描述描述特殊的权限，从最高位开始分别对应着setuid、setgid和sticky权限
  - 一个八进制的3个比特(对应着rwx)，如果某个比特为1，表示拥有对应的权限；如果没有权限，则相应的比特为0
  - 比如rwx对应着数字7，rw对应着数字6，r对应着数字4，rx对应着数字5，而0表示没有任何权限

stat -c '%a' ~ 查看HOME目录的权限

| 数字 | 含义 |
| --- | --- |
| 755 | 文件拥有者拥有读写和执行权限，用户组和其他有读和执行权限 |
| 666 | 文件拥有者、用户组和其他都有读写权限 |
| 664 | 文件拥有者、用户组有读写权限，其他有读权限 |
| 6775 | 文件拥有者拥有读写执行以及SETUID权限，用户组有读写执行以及SETGID权限，其他有读和执行权限 |
| 1777 | 文件拥有者、用户组、其他有读写执行权限，拥有sticky权限 |

# 设置默认权限umask

- 内置命令umask用于控制新文件或者目录的默认权限
- 已有的文件不受影响

chmod命令使用符号模式来改变对象a(所有人)的权限时，需要考虑umask的设置!

umask [-S] [octet-mode]

- 八进制表示法的数字描述哪些权限不允许
- 即如果某个比特为1，表示不允许对应位置的权限

**umask 002，表示不允许other的w权限，即其他用户不允许写**

- 不加参数的umask会输出当前的默认权限设置
- 也可采用符号表示法描述和设置默认权限，采用-S选项，给出了允许的权限，即那些没有指出的权限不允许

umask 002也可使用 umask –S u=rwx,g=rwx,o=rx

| umask | 含义（不允许） | umask | 含义 |
|---|---|---|---|
| 022 | group和other都不允许w | 002 | other不允许w |
| 027 | group不允许w，other不允许rwx | 007 | other不允许rwx |
| 026 | group不允许w，而other不允许rw | 006 | other不允许rw |
| 077 | group和other不允许rwx | | |

# 设置默认权限umask

- 内置命令umask用于控制新文件或者目录的默认权限，限定了哪些权限不允许，或者哪些权限是允许的
- 目录的默认权限为777，文件的默认权限为666(即不包括x权限)
- 新目录或新文件的权限为默认权限与对应位置的umask进行“减法”运算后的结果
- 相减并不是真正整数的减法，类似于集合的差运算
  - umask某个位置为1，表示实际的权限中应该移走该位置对应的权限
  - 6-3 = 4 → (rw-) - (wx) = r
- umask 002时
  - 新目录的最终权限为775 rwxrwxr-x
  - 新文件的权限为664 rw-rw-r—