连接和权限

主要内容

- su和sudo
- 硬连接和软连接
- 权限管理

su （substitute userid）

su [options] [username] 如果没有用户名，则相当于root

- 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 - test2或su –l test2 全新的环境变量
切换为root应该使用su -或su -l

- 开启的新会话，缺省以非登录shell方式启动新的shell，会继承当前用户的大部分环境变量，除了与用户身份和安全相关的环境变量外，即HOME, SHELL, USER, LOGNAME, PATH和IFS
- 选项 -l, --login, - 以登录shell方式启动新的shell，除了TERM和DISPLAY等，其他环境变量都被重置
- 选项 -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

```
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大部分环境变量
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
```

```
demo@mars:~$ su - test2
Password:
test2@mars:~$ echo $MYENV

test2@mars:~$ exit
demo@mars:~$ su -c 'ls -l' - test2
Password:
total 12
-rw-r--r-- 1 test2 test2 8980 4月  27 22:46
examples.desktop
```

sudo 以另一个用户身份执行程序

sudo [options] [cmd]

以另外一个用户（缺省超级用户）身份执行一个命令（甚至包括su），一般会重置环境

-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

-E, --preserve-env 继承环境变量

-u user　　以user身份执行，缺省为root

-s, --shell　启动一个新的shell(当前环境变量SHELL指定的shell) 。如果有cmd，执行cmd后退出

-i, --login　就像新的用户登录一样，以登录shell方式启动新的shell。如果有cmd，执行cmd后退出

--　　表示选项部分结束，后面为参数部分

- 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

```
dlmao@mars:~$ ls -a /root
ls: cannot open directory '/root': Permission denied
dlmao@mars:~$ sudo ls -a /root
.  ..  .bash_history  .bashrc  .cache  .profile
dlmao@mars:~$ sudo -s
root@mars:~# echo $USER $LOGNAME $HOME
root root /home/dlmao
root@mars:~# exit
exit
dlmao@mars:~$ sudo -i
root@mars:~# echo $USER $LOGNAME $HOME
root root /root
```

考虑到安全因素，许多发行版不允许root身份密码登录，建议

- 使用sudo临时执行某个命令
- 可以通过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

- /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

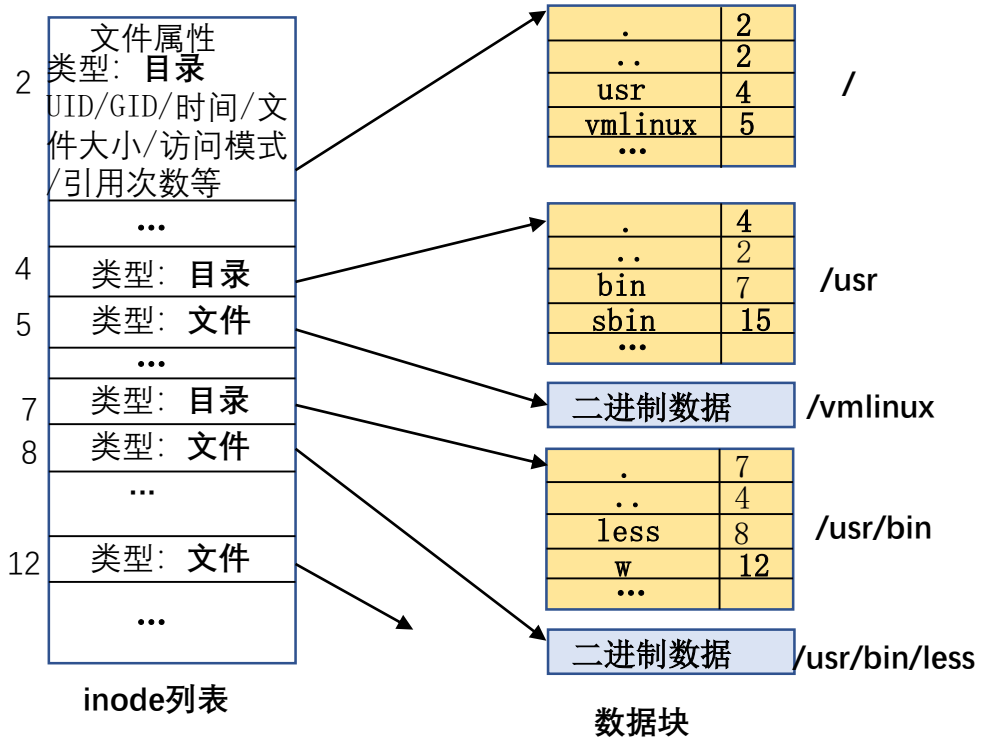
文件系统

- 文件系统的组织： mkfs命令用于创建文件系统
  - 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（包括子目录）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

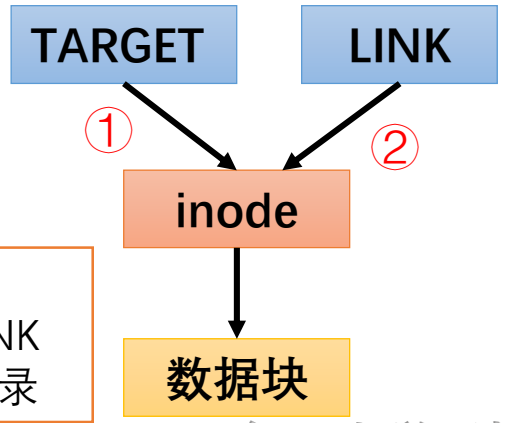
- 注意：**文件的文件名并不在inode节点中，而是保存在包含该文件的目录文件的数据块中**
- **文件和目录等通过inode编号(而不是文件名)唯一标识**
- 存储设备上的文件系统的根目录文件的inode编号缺省为2




```
test2@mars:~$ ls -ai
1053231 .                1071908 .bashrc
1048577 ..               1071910 .profile
1071902 .bash_history    1071900 .viminfo
1054161 .bash_logout     1055303 examples.desktop
```

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）？
  - 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，**即建立一个到inode节点所标识的文件的**硬连接（Hard Link），**或者说一个inode节点可以有多个名字
  - 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节点中的**引用计数**记录指向它的硬连接的个数，引用计数为0时，inode节点被回收
  - 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节点来实现，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



- 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

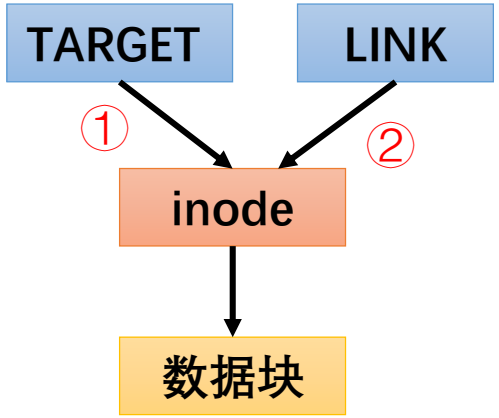
ln TARGET LINK
创建一个到TARGET的硬连接，该硬连接的文件名为LINK
两者必须在同一个文件系统，且TARGET一般不能是目录

硬连接(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

- 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
- 要整理文件，该文件可以归并到不同的目录，使用硬连接可以节省存储空间

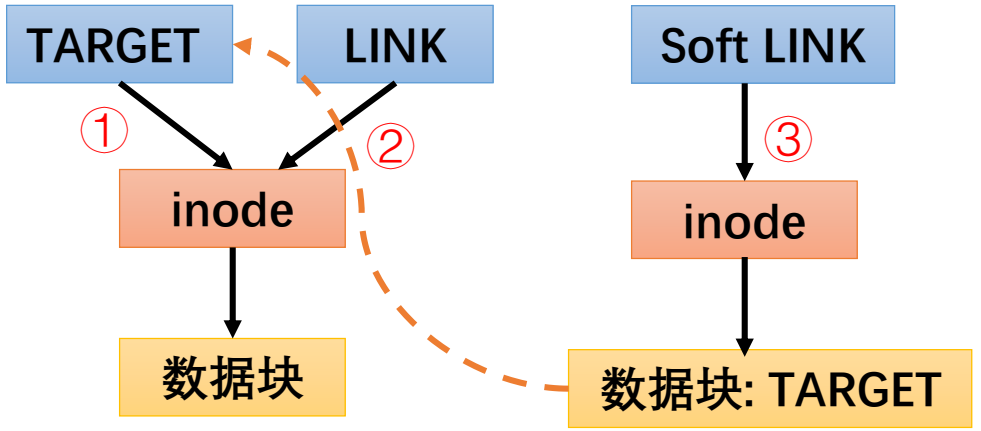
```
$ touch hosts
$ ln hosts hosts.lnk
$ ls -li hosts*
1049704 -rw-r--r-- 2 dlmao dlmao 219 Apr 28 12:39 hosts
1049704 -rw-r--r-- 2 dlmao dlmao 219 Apr 28 12:39 hosts.lnk
$ mkdir dir-a
$ ln dir-a d
ln: dir-a: hard link not allowed for directory
```



符号连接（symbolic link），也称为软连接

- 硬连接通过目录项实现，而符号连接通过inode节点实现,类似于windows的快捷方式
- 分配一个新的inode节点，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
- 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。权限控制属性是无法修改的，因为修改权限没有任何意义，软连接在访问时并不检查权限等属性
- 平时在打开符合连接进行读写时，缺省会**展开该符号链接**，直到找到对应的文件为止

创建符号连接 **ln –s TARGET LINK_NAME**




```
$ ls -F hosts.*
hosts.lnk  hosts.slnk@  hosts.slnk2@
```

```
$ ln -s hosts hosts.slnk
$ ln -s hosts.slnk hosts.slnk2
$ ls -li hosts*
1049704 -rw-r--r-- 2 dlmao dlmao 219 Apr 28 12:39 hosts
1049704 -rw-r--r-- 2 dlmao dlmao 219 Apr 28 12:39 hosts.lnk
1071916 lrwxrwxrwx 1 dlmao dlmao   5 Apr 28 12:40 hosts.slnk -> hosts
1071918 lrwxrwxrwx 1 dlmao dlmao  10 Apr 28 12:40 hosts.slnk2 -> hosts.slnk
$ 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... 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(即basename)
- ln [OPTION]... TARGET　等价于 ln 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 /etc/hosts
- **-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
- -f, --force　-i, --interactive 要覆盖时是强制覆盖还是询问用户

```
$ mkdir lnk
$ ln -s hosts* lnk
$ ls -F lnk
hosts@  hosts.lnk@  hosts.slnk@  hosts.slnk2@
$ ln -sr  /etc/passwd passwd
$ ls -l
lrwxrwxrwx 1 dlmao dlmao   25 Nov 13 20:54 passwd -> ../../../../../etc/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。有些命令会有如下选项:
  - -L, --dereference 展开符号连接，一般为缺省行为
  - -P, --no-dereference 不展开符号连接
- 删除(rm)、移动(mv)等命令不展开符号连接，根本就没有相应选项，如果目标是符号连接，对符号连接进行操作
- 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选项展开符号连接
- **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，缺省展开符号连接

```
$ 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 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 按照指定格式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

```
dlmao@mars:~$ stat /etc/hosts
  File: /etc/hosts
  Size: 219             Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 262298      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2020-04-27 22:13:07.351999703 +0800
Modify: 2020-02-23 12:47:37.049929788 +0800
Change: 2020-02-23 12:47:37.049929788 +0800
 Birth: -
dlmao@mars:~$ stat -c '%i %n %a' /usr/bin/k*
655941 /usr/bin/kbdinfo 755
655942 /usr/bin/kbxutil 755
655589 /usr/bin/kernel-install 755
655944 /usr/bin/kerneloops-submit 755
687747 /usr/bin/key-mon 755
655945 /usr/bin/keyring 755
655946 /usr/bin/killall 755
655947 /usr/bin/kmodsign 755
dlmao@mars:~$ stat -f ~
  文件："/home/dlmao"
    ID：afac32e800ea6948 文件名长度：255      类型：ext2/ext3
块大小：4096         基本块大小：4096
    块：总计：5127316     空闲：3390662     可用：3124448
Inodes：总计：1310720     空闲：1112907
```

分配8个block(每个512字节)

I/O读写时的基本单位

inode 文件名 访问模式(数字)

主要内容

- su和sudo
- 硬连接和软连接
- 权限管理

文件和目录权限

- i-node节点中包含了访问控制相关信息，这些信息是文件拥有者的UID和GID、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
- 某个进程(用户) 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
  - 如果进程(用户）的有效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是否等于文件的GID属性，如果不等，判断有效的UID是否在文件GID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

rw-r--r--表示允许用户读和写，组用户读，其他用户读

```
$ ls -ld /etc/{passwd,shadow} ~
-rw-r--r--  1 root  root   2585 Apr 27 22:46 /etc/passwd
-rw-r-----  1 root  shadow 1566 Apr 27 22:46 /etc/shadow
drwxr-xr-x 24 dlmao dlmao  4096 Apr 28 14:30 /home/dlmao
```

文件和目录的读写和执行权限

- 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节点所指向的数据块中
  - 文件的访问不仅仅与**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路径名中的各级目录都有x权限

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(rm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	搜索权限，允许进入该目录，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一般要求拥有**超级用户**的权限

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

改变文件的拥有者(uid)或者所属用户组(gid)，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 :wheel  1.txt    # 改变1.txt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
```

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

-R, --recursive 表示递归

改变文件访问模式chmod

- -v, --verbose 给出权限变化的详细信息
- -R, --recursive 递归改变目录及各级子目录中的文件的访问模式

- 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...　　符号方式

chmod [OPTION]... OCTAL-MODE FILE...　　八进制数字方式

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

权限改变	含义
+	增加某些权限
-	去除某些权限
=	设置为某些权限

对象	含义
u(user)	文件或目录的拥有者的权限
g(group)	文件或目录所属用户组的权限
o(other)	其他用户的权限
a(all)	用户/用户组和其他三组的权限。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	删除其他的读写权限
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

```
$ touch readme
$ ls -l
total 0
-rw-rw-r-- 1 dlmao dlmao 0 Apr 28 18:23 readme
$ chmod -v ug-w readme
mode of 'readme' changed from 0664 (rw-rw-r--) to 0444 (r--r--r--)
```

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

rwxrwxrwx

user group other

- 仍然通过3组9个字符描述权限（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

* chmod u+s FILE 可增加setuid权限
* chmod g+s FILE 可增加setgid权限
* chmod o+t FILE 可增加sticky权限

chmod u+sx FILE 文件拥有者增加执行权限和setuid权限

chmod o=rtx FILE 其他设置为读、执行和sticky权限

每组权限中第三个用于表示执行权限的字符

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，无执行权限
s	**有setuid或setgid权限再加上执行权限**
T	表示sticky权限，无执行权限
t	**表示sticky权限加上执行权限**

```
$ 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
```

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

r	w	x
1	1	1

r	w	-
1	1	0

r	-	x
1	0	1

r	-	-
1	0	0

r	w	x
0	0	0

- 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
  - 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， rx对应着数字5， r对应着数字4，而0表示没有任何权限
  - 4位数字时第1个数字描述特殊的权限，从最高位开始分别对应着setuid、setgid和sticky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
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

setuid	setgid	sticky
√	√	
1	1	0

setuid	setgid	sticky
		√
0	0	1

```
$ chmod -v 644 readme
mode of 'readme' changed from 0444 (r--r--r--) to 0644 (rw-r--r--)
```

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
- 不加参数的umask会输出当前的默认权限设置
- umask的设置不会影响已有的文件

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

r	w	x	r	w	x	r	w	x
0	0	0	0	1	0	1	1	1

umask 0002，表示不允许other的w权限，即其他用户不允许写

- 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	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
077	group和other不允许rwx		

```
$ umask
0002
$ umask 022
$ umask -S
u=rwx,g=rx,o=rx
```

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rwxrwxrwx(777)，文件的默认权限为rw-rw-rw-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—

特殊权限:setuid/setgid

- 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真实的和有效的用户ID和组ID
  - 真实用户ID：调用该程序时当前用户的ID，谁在负责正在运行的进程
  - **有效用户ID（set user ID)**：用于权限控制，是否可以访问文件，新创建的文件的权限是什么，是否可以发送信号等
- 超级用户可以调用setuid/seteuid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**用户组属性设置为所在的有setgid权限的目录的用户组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
  - 可以设置其他用户不允许写， group中的用户允许写
  - 该目录中新建文件和目录时，这些文件和目录的用户组都相同，都是group，而且目录也具有SETGID权限
  - 通过设置合适的umask，group中的用户在新建文件和目录时授予用户组group相应的权限，比如group用户对应的umask可设置为：
    - 0: 新建的文件，组可读写，用户可以阅读和修改其他用户的文件
    - 2: 新建的文件，组不可写，但是读，用户可以阅读其他用户的文件

特殊权限: 目录的setgid权限

sgid-dir为shared用户组的共享目录，允许shared用户组的用户读写其他人的文件

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	创建一个新目录
sudo chown :shared sgid-dir	改变该目录的用户组为shared
chmod u=rwx,g=rwxs,o= sgid-dir	改变权限, 拥有setgid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，该目录下的文件或目录的删除以及改名受到限制
  - 该目录一般允许他人进行读写，但是一旦创建了文件或目录，**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
  - 其他用户即便从该文件所在目录的权限设置(比如设置为用户组或任何人都可以读写)来看允许删除或改名时，也不允许执行该操作，即无法删除另外一个用户的文件
- 常用于控制对于共享目录（比如临时文件目录/tmp）的访问

```
dlmao@mars:~$ sudo -u test2 touch /tmp/tmp.txt
dlmao@mars:~$ ls -ld /tmp/{,tmp.txt}
drwxrwxrwt 12 root  root  4096 Apr 28 22:23 /tmp/
-rw-rw-r--  1 test2 test2    0 Apr 28 22:26 /tmp/tmp.txt
dlmao@mars:~$ rm /tmp/tmp.txt
rm: remove write-protected regular empty file '/tmp/tmp.txt'? y
rm: cannot remove '/tmp/tmp.txt': Operation not permitted
```