Linux操作系统

11 权限管理

主讲：杨东平
中国矿大计算机学院

用户组

- 在 Linux 中的每个用户必须属于一个组，不能独立于组外
  - 所有者(u)
    - 一般为文件的创建者，谁创建了该文件，就天然的成为该文件的所有者
      - ls －ahl 命令可以看到文件的所有者
      - chown 用户名 文件名 命令可以修改文件的所有者
  - 文件所在组(g)
    - 当某个用户创建了一个文件后，这个文件的所在组就是该用户所在的组
      - ls －ahl 命令可以看到文件的所有组
      - chgrp 组名 文件名 命令可以修改文件所在的组
  - 其它组(o)
    - 除文件的所有者和所在组的用户外，系统的其它用户都是文件的其它组
  - 所有组(a)

文件权限

- Linux 的文件和目录有三种权限：
  - r： 可读(read)
  - w： 可写(write)
  - x： 可执行(execute)
  - -： 没有相应的权限
- 每个用户组都有对文件的三种权限的顺序组合

例：文件的组与权限


```
[root@localhost ~]# ls -ahl
total 68K
dr-xr-x---.  2 root root 4.0K Sep 19 15:37 .
dr-xr-xr-x. 23 root root 4.0K Sep 19 15:33 ..
-rw-------.  1 root root 1.1K Sep  6 16:23 anaconda-ks.cfg
-rw-------.  1 root root  13K Sep 18 22:29 .bash_history
-rw-r--r--.  1 root root   18 May 20  2009 .bash_logout
-rw-r--r--.  1 root root  176 May 20  2009 .bash_profile
-rw-r--r--.  1 root root  176 Sep 23  2004 .bashrc
-rw-r--r--.  1 root root  100 Sep 23  2004 .cshrc
-rw-r--r--.  1 root root  229 Sep  6 20:48 exec
-rw-r--r--.  1 root root 8.7K Sep  6 16:23 install.log
-rw-r--r--.  1 root root 3.4K Sep  6 16:22 install.log.syslog
-rw-r--r--.  1 root root  129 Dec  4  2004 .tcshrc
```


设置权限命令：chmod

- 英文原意：change the permission mode of a file
- 语法：chmod [选项] [参数]
- 功能：变更文件或目录的权限
- 参数
  - 权限模式：指定文件的权限模式
  - 文件：要改变权限的文件

选项	含义
-c或--changes	效果类似 “-v”参数，但仅回报更改的部分
-f或--quiet或--silent	不显示错误信息
-R或--recursive	递归处理，将指令目录下的所有文件及子目录一并处理
-v或--verbose	显示指令执行过程
--reference=<参考文件或目录>	把指定文件或目录的所属群组全部设成和参考文件或目录的所属群组相同
<权限范围>+<增加的权限>	开启权限范围的文件或目录的该选项权限设置
<权限范围>-<取消的权限>	关闭权限范围的文件或目录的该选项权限设置
<权限范围>=<设定权限>	指定权限范围的文件或目录的该选项权限设置

设置权限命令 chmod 举例

- chmod a=rwx test.av
  - 给 test.av 文件的所有用户设置 rwx 权限
- chmod u=rwx,g=rx,o=rx abc
  - 对 abc 文件，为所有者设置 rwx 权限，组用户和其它用户设置 rx 权限
- chmod u-x，g+w abc
  - 对 abc 文件，去除所有者执行权限，增加组的写权限
- chmod a+r abc
  - 给所有用户添加读权限

权限的数字表示

- 权限的数字表示
  - r 4
  - w 2
  - x 1
- 例：rwxr-xr-x
  7 5 5
- 例： chmod a=rwx file
  等价于 chmod 777 file
- 例： chmod ug=rwx,o=x file
  等价于 chmod 771 file

权限对文件的作用

- r： 读取文件内容(cat more head tail)
- w：编辑、新增、修改文件内容(vi echo)
  - 但不包括删除文件
- x： 可执行

权限对目录的作用

r： 可以查看目录内的文件列表(ls)

w： 可创建删除目录内的文件，但必须有 x 权限才可真正执行(touch rm mv cp)

x： 可以进入目录内(cd)

- 最高权限
  - 文件：最高权限是 x
  - 目录：最高权限是 w，可用权限权限只有 0、5(rx)和7(rwx)，而4(r)没有意义(进不去)，1(x)或6(rw)也没有意义
- 注意：只有对文件的目录有 w 权限，才能删除该文件

修改文件的所有者命令：chown

- 英文原意：Change owner
- 语法：chown [选项] [所有者[:组]] 文件列表
  - 只有文件主和超级用户才可以使用该命令
  - 参数
    - 可用用户名或UID，组名或GID。省略“:组”时仅改变所有者
    - 文件列表：用空格分隔，支持shell通配符

选项	含义
-c或--changes	效果类似 “-v”参数，但仅回报更改的部分
-f或--quite或--silent	不显示错误信息
-h或--no-dereference	只对符号连接的文件作修改，而不更改其他任何相关文件
-R或--recursive	递归处理，将指定目录下的所有文件及子目录一并处理
-v或--verbose	显示指令执行过程
--dereference	效果和 “-h”参数相同
--reference=<参考文件或目录>	把指定文件或目录的拥有者与所属群组全部设成和参考文件或目录的拥有者与所属群组相同

修改文件的所属组命令：chgrp

- 英文原意：Change group
- 语法：chgrp [选项] [参数]
- 选项
  - -c或--changes：效果类似 ”-v” 参数，但仅回报更改的部分
  - -f或--quiet或--silent：不显示错误信息
  - -h或--no-dereference：只对符号连接的文件作修改，而不是该其他任何相关文件
  - -R或--recursive：递归处理，将指令目录下的所有文件及子目录一并处理
  - -v或--verbose：显示指令执行过程
  - --reference=<参考文件或目录>：把指定文件或目录的所属群组全部设成和参考文件或目录的所属群组相同
- 参数
  - 组：指定新工作组名称
  - 文件：指定要改变所属组的文件列表，用空格隔开

举例

- 将目录 /usr/meng 及其下面的所有文件、子目录的文件主改成 liu：
  - chown -R liu /usr/meng
- 将 /usr/meng 及其子目录下的所有文件的用户组改为 mengxin
  - chgrp -R mengxin /usr/meng

文件的默认权限

- 当进入 Linux 系统后新创建的文件或者目录总是会有一个默认的权限

- 查看与设置默认权限命令:umask
  - 语法：umask [选项] [参数]
  - 功能：查看与设置新建文件权限的掩码
    - 掩码：指定哪些权限将在新文件的默认权限中被删除
  - 选项：
    - -p：输出的权限掩码可直接作为指令来执行
    - -S：以符号方式输出权限掩码
  - 参数：
    - 权限掩码：指定权限掩码

举例：查看默认权限

- 直接查看，

```
[root@localhost ~]# umask
0022
```

- 以模式方式显示

```
[root@localhost ~]# umask -S
u=rwx,g=rx,o=rx
```

- 输出可被调用，重定向

```
[root@localhost ~]# umask -p
umask 0022
```

- 其中：
  - 0022 表示的权限是 rwxr-xr-x，第一位的 0 是特殊权限，这里先不做考虑

举例：设置默认权限

- umask u=, g=w, o=rwx
  - 执行该命令以后，对于后续创建的新文件，其文件主的权限未做任何改变，而组用户没有写权限，其他用户的所有权限都被取消

- 注意：操作符 "=" 在 umask 命令和 chmod 命令中的作用恰恰相反

文件的默认权限(续)

- 文件默认不能建立为执行文件，必须手工赋予执行权限
- 文件默认权限最大为 666 (-rw-rw-rw-)

- 文件默认权限的计算：
  - 最大权限减去 umask 码
  - 结果为奇数，则默认权限为各奇数位+1
    - 例：如果 mask=135
      666-135=431，奇数+1为442，转换为权限r--r----x
  - 结果为偶数，则等于默认权限
    - 例：666-022=644，转换为权限 rw-r--r--

文件的默认权限(续)

- umask 命令中权限掩码可以是八进制数
- 例：umask 026
  - 执行该命令以后所创建的文件权限将变为640(按八进制对应减：666-026=640)
  - 执行命令前

```
[root@localhost ~]# ls -l
total 24
-rw-------. 1 root root 1899 Sep  6 16:23 anaconda-ks.cfg
-rw-r--r--. 1 root root  229 Sep  6 20:48 exec
-rw-r--r--. 1 root root 8815 Sep  6 16:23 install.log
-rw-r--r--. 1 root root 3384 Sep  6 16:22 install.log.syslog
```

  - 执行命令并创建文件后

```
[root@localhost ~]# ls -l
total 28
-rw-------. 1 root root 1899 Sep  6 16:23 anaconda-ks.cfg
-rw-r--r--. 1 root root  229 Sep  6 20:48 exec
-rw-r-----. 1 root root   45 Sep 19 18:29 hello.sh
-rw-r--r--. 1 root root 8815 Sep  6 16:23 install.log
-rw-r--r--. 1 root root 3384 Sep  6 16:22 install.log.syslog
```

  - 视频：33 文件默认权限
- 系统默认的掩码是 0022

目录的默认权限

- 目录默认权限最大为 777 (drwxrwxrwx)

- 目录默认权限的计算：
  - 最大权限减去 umask 码等于默认权限
  - 例：777-022=755 转换为权限 rwxr-xr-x

- 例：
  - 目录默认最大权限为777，umask 值为 022
  - -rwxrwxrwx减去-----w--w-等于-rwxr-xr-x

修改 umask 值

- 临时修改
  - 语法：umask 默认权限掩码值
  - 说明：重启机器后失效
- 永久修改
  - 默认权限掩码值存储在 /etc/profile 文件中
    - 用 vi 命令编辑 /etc/profile 文件中的 umask 值即可
  - 也可以在 /etc/.bashrc 文件添加或修改以下格式内容：
    umask 0002

权限掩码小结

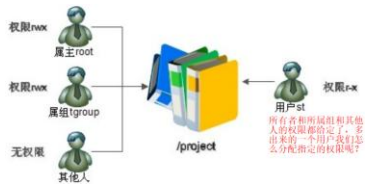
- 一般 root 的 umask 值为 022
- 一般普通用户的 umask 值为 002
- umask 值越小权限越大

ACL (Access Control List，访问控制列表)权限

- 1) ACL 权限简介与开启
- 2) 查看与设置 ACL 权限
- 3) 最大有效权限与删除 ACL 权限
- 4) 默认 ACL 权限和递归 ACL 权限

什么是 ACL 权限？

- 假设场景：
  - 某大牛在 QQ 群内直播讲解 Linux 系统的权限管理，并将资料存放到一个公有的 Linux 系统的 /project 目录中
    - /project 目录的所有者是大牛，有 rwx 权限
    - QQ 群内的所有用户分配在一个所属组里面，也有对 /project 目录的 rwx 权限
    - QQ 群外的其他人，无任何访问 /project 目录的权限

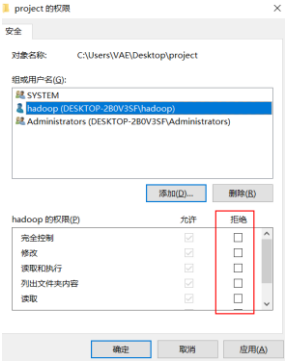


什么是 ACL 权限？(续)

- 问题：
  - 如果不属于 QQ 群内的直播旁听者听完之后，被允许访问 /project 目录查看资料，但是不能进行修改(有 r-x 权限)，该如何解决？
    - 1) 一个文件只能有一个所属组，若将旁听者分配到 QQ 群所属组内，那么他就有了写权限，这是不被允许的
    - 2) 如果将该旁听者视为目录 /project 的其他人，并且将 /project 目录的其他人权限改为 r-x，那么不是旁听的人也就能访问 /project 目录了，这显然也是不被允许的。该如何解决呢？

什么是 ACL 权限？(续)

- Windows 系统给某个文件分配权限的办法：
  - 想要让某个用户不具备某个权限，直接不给他分配这个目录的相应权限就行了
  - Linux 系统也可以为指定的用户分配指定目录的指定权限，也就是 ACL 权限分配



什么是 ACL 权限？(续)

- ACL，又称存取控制串列，是使用以访问控制矩阵为基础的访问控制方法，每个对象对应一个串列主体
  - ACL描述每个对象各自的访问控制，并记录可对此对象进行访问的所有主体对象的权限
- CentOS7 默认创建 xfs 和 ext4 文件系统具有 ACL 功能，而之前的版本，默认手工创建的 ext4 文件系统没有 ACL 功能，需要手动增加

查看分区 ACL 权限是否开启：dump2fs

- 语法：dumpe2fs [-bfhixV] [-o superblock=superblock] [-o blocksize=blocksize] device
- 功能：查询指定分区详细文件系统信息

选项	描述
-b	打印文件系统中的坏块
-o	不常用，检查严重损坏文件系统时指定
-f	强制显示所有信息，即便dumpe2fs对有些文件系统功能标识不能识别
-i	显示image文件系统信息。device指定image文件的路径
-h	只显示超级块信息
-x	将已分组的块的数量用十六进制显示
-v	显示dumpe2fs的版本号并推出

查看分区 ACL 权限是否开启：dump2fs(续)

- 查看某个文件是否支持 ACL 权限，首先要看文件所在的分区是否支持 ACL 权限
  - 1) 查看当前系统有哪些分区：df -h

```
[root@localhost ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root   18G  755M   16G   5% /
tmpfs                     499M     0  499M   0% /dev/shm
/dev/sda1                 485M   32M  428M   7% /boot
```

  - 2) 查看指定分区详细文件信息：dumpe2fs -h 分区路径

```
[root@localhost ~]# dumpe2fs -h /dev/sda1 | more -15
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name:   <none>
Last mounted on:          /boot
Filesystem UUID:          6794d89b-d833-431c-b628-1d1576eb5ba5
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype n
eeds_recovery extent flex_bg sparse_super huge_file uninit_bg dir_nlink extra_is
ize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl   这里有 ACL，一般分区都默认支持 ACL 权限
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              128016
Block count:              512000
--More--
```


开启分区 ACL 权限

- 临时开启分区 ACL 权限
  - 语法：mount -o remount,acl /
  - 功能：重新挂载根分区，并挂载时加入 acl 权限
  - 注意：这种命令开启方式，如果系统重启了，那么根分区权限会恢复到初始状态

开启分区 ACL 权限(续)

- 永久开启分区 ACL 权限
  - 1) 分区 ACL 权限存储在 /etc/fstab 中
  - 2) 修改配置文件 /etc/fstab (可以用 vi 编辑修改)

  - 上面是修改根分区拥有 acl 权限

```
UUID=490ed737-f8cf-46a6-ac4b-b7735b79fc63 /          ext4    defaults,acl    1 1
```

  - 3) 重新挂载文件系统或重启系统，使得修改生效
    - mount -o remount /

查看 ACL 权限：getfacl

- 语法：getfacl [aceEsRLPtpndevh] 文件名 …
- 功能：获得某个文件或目录的 ACL 权限
- 选项：

-a/--access	显示文件的ACL
-d/--default	仅显示默认的ACL
-c/--omit-header	不显示带有#的信息
-e/--all-effective	显示所有有效的权限
-E/--no-effective	显示无效的权限
-s/--skip-base	跳过只有基础词目的文件
-R/--recursive	递归
-L/--logical	跟踪符号链接，默认情况下只跟踪符号链接文件，跳过符号链接目录
-P/--physical	跳过所有符号链接，包括符号链接文件
-t/--tabular	使用列表输出格式
-n/--numeric	打印数值形式的用户或组身份
-p/--absolute-names	不要剥去路径上的 "/"

查看 ACL 权限：getfacl(续)

➢例：

```
[pt@node3 project]$ getfacl /project
getfacl: Removing leading '/' from absolute path names
# file: project
# owner: root
# group: QQgroup
user::rwx
user:pt:r-x
group::rwx
mask::rwx
other::---
```


设定 ACL 权限：setfacl

➢语法：setfacl [-bkndRLP] {-m|-M|-x|-X…} 文件名 …

❖常用选项：

-b/--remove-all	删除所有的 ACL 权限(所有者，群组，其他)将被保留
-m	设定 ACL 权限
-x	删除指定的 ACL 权限
-k/--remove-default	删除默认的 ACL 权限。如果没有缺省规则，将不提示
-d/--default	设定默认的 ACL 权限
-R/--recursive	递归设定所有文件及目录的 ACL 权限

➢ 注意：

❖setfacl 给用户或用户组设定的 ACL 权限不是真正的最终权限

❖实际权限是 ACL 权限与 mask 权限“相与”之后的权限

❖一般默认的 mask 权限是 rwx

设定 ACL 权限：setfacl (续)

➢ 用法1：setfacl -m u:用户名:权限 指定文件名

❖给用户设定 ACL 权限

➢ 用法2：setfacl -m g:组名:权限 指定文件名

❖给用户组设定 ACL 权限

最大有效权限 mask

➢mask 用来指定最大有效权限，我们给用户赋予的 ACL 权限是需要和 mask 权限“相与”才能得到用户的真正权限

➢用 getfacl 命令可以查看 mask 权限

➢设置 mask 权限

❖语法：setfacl -m m:权限 文件名

删除 ACL 权限

➢删除指定用户的 ACL 权限

❖语法：setfacl -x u:用户名 文件名

➢删除指定用户组的 ACL 权限

❖语法：setfacl -x g:组名 文件名

➢删除文件的所有 ACL 权限

❖语法：setfacl -b 文件名

递归 ACL 权限

➢当给目录通过 -R 选项设定 ACL 权限时，该目录下的所有子文件和子目录也会拥有相同的 ACL 权限

➢语法：setfacl -m u:用户名:权限 -R 文件名

➢注意：

❖递归权限只对目录有用

❖递归 ACL 权限仅对目录中已经有的文件有效，但是对新建的文件无效

❖警告，少用ACL权限

默认 ACL 权限

- 如果给某目录设定了默认的 ACL 权限，那么该目录中所有新建的子文件会继承目录的 ACL 权限
- 语法：setfacl -m d:u:用户名:权限 文件名
- 注意：
  - 默认权限对目录有效
  - 默认权限对此后在该目录下新建的文件生效
- 默认和递归ACL一般是针对目录的，对文件无意义

sudo 权限

- sudo 是 Linux 系统管理指令，它是允许系统管理员让普通用户执行一些或者全部的 root 命令的一个工具，如 halt、reboot、su等
  - 目的：从而不仅减少了 root 用户的登录和管理时间，也提高了安全性
- sudo 预设的身份为root，sudo 的配置文件是 /etc/sudoers

sudo 的工作过程

- 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
- 4) root 执行 sudo 时不需要输入密码(sudoers文件中有配置 root ALL=(ALL) ALL 这样一条规则)
- 5) 若欲切换的身份与执行者的身份相同，也不需要输入密码

/etc/sudoers 文件格式

- /etc/sudoers 文件格式包括
  - 别名类型
  - 放权格式

/etc/sudoers 文件格式：别名类型

- 别名类型有四种：

Host_Alias	定义主机名别名
User_Alias	用户别名，可以是用户和用户组(组名前面要加 % 号)
Runas_Alias	定义 runas 别名，即指定“目的用户”，也就是 sudo 允许转换至的用户
Cmnd_Alias	定义命令别名

- 需要注意：
  - 在每一种 Alias 后面定义的别名 NAME 可以是包含大写字母、下划线连同数字，但必须以一个大写字母开头
  - 配置文件中的 Default env_reset 表示重置(就是去除)用户定义的环境变量，也就是说，当你用 sudo 执行一个命令的时候，你当前用户设置的所有环境变量都将无效

/etc/sudoers 文件格式：放权格式

- 放权格式：

  授权用户/组 主机名=(允许转换至的用户) NOPASSWD:命令动作

- 其中
  - 授权用户/组、主机名和命令动作这三个要素缺一不可
  - 在动作之前能够指定转换到特定用户下，指定转换的用户要用 ( ) 号括起来，
  - 假如无需密码直接运行命令的，应该加 NOPASSWD: 参数，不需要时方可省略

sudo 配置文件 /etc/sudoers 专用编辑器：visudo

- visudo 的好处是在添加规则有误时，保存退出会提示给我们错误信息

给用户赋予 sudo 权限

- 命令：visudo
  - 添加或修改的内容格式之一：

用户名 被管理主机的地址=(可使用的身份) 绝对路径的授权命令

- 例：授权普通用户可以重启服务器

user1 ALL=(ALL) /sbin/shutdown -r now

  - 允许 user1 执行 shutdown -r now 重启命令
  - 此处写得越详细，普通用户得到的权限就越小，如果只写 /sbin/shutdown，那么普通用户就可以使用 shutdown命令的所有参数了

sudo 命令

- 语法：sudo [选项] [参数]
- 选项

-b：在后台执行指令
-H：将 HOME 环境变量设为新身份的 HOME 环境变量
-k：结束密码的有效期限，即下次再执行 sudo 时需输入密码
-l： 列出目前用户可执行与无法执行的指令
-p：改变询问密码的提示符号
-s<shell>：执行指定的shell
-u<用户>：以指定的用户作为新的身份。若无此参数，则预设以 root 作为新的身份
-v：延长密码有效期限5分钟
-V ：显示版本信息

- 参数

指令：需要运行的指令和对应的参数

给用户组赋予 sudo 权限

- 格式类似上例，只是把用户名换成组名的同时在组名前加 %
- 例：

%group1 ALL=(ALL) /sbin/shutdown -r now

用户查看与执行可用的 sudo 命令

- 查看用户可用的 sudo 命令
  - 语法：sudo -l
- 普通用户执行 sudo 命令
  - 语法：sudo 被授权的绝对路径的命令
- 例：

```
[root@localhost ~]# su - user1
[user1@localhost ~]$ sudo -l
#查看可用的sudo命令
[user1@localhost ~]$ sudo /sbin/shutdown -r now
#普通用户执行sudo赋予的命令
```


ACL 示例：某大牛的 /project 处理方法(视频：34 用户的ACL权限)

- 1) 用 dumpe2fs -h 查看分区是否开启 ACL
- 2) 开始分区的 ACL 权限(这里是临时开启)
- 3) 用 mkdir 创建目录(/project)
- 4) 用 useradd 命令创建用户1(zhangsan)和用户2(lisi)
- 5) 用 groupadd 命令创建用户组(QQgroup)
- 6) 用 gpasswd 命令将用户1(zhangsan)和用户2(lisi)加入组(QQgroup)
- 7) 用 chown 命令将目录(/project)的所属组改为用户组(QQgroup)
- 8) 用 chmod 命令将目录(/project)的权限修改为770
- 9) 用 useradd 加入旁听者用户(pt)
- 10) 用 passwd 为旁听者设置密码
- 11) 用 setfacl 设置旁听者(pt)的ACL权限为 rx
- 12) 用 getfacl 查看目录(/project)的ACL权限，观察旁听者的权限是否满足要求
- 13) 为验证旁听者(pt)对目录(/project)没有写权限，用 su 命令切换到旁听者(pt)用户，然后进入目录(/project)，并创建文件和目录，此时应该是失败的

```
[root@localhost ~]# mount -o remount,acl /
[root@localhost ~]# dumpe2fs -h /dev/sda1
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name:   <none>
Last mounted on:          /boot
Filesystem UUID:          6794d09b-d033-431c-b628-1d1576eb5ba5
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype n
eeds_recovery extent flex_bg sparse_super huge_file uninit_bg dir_nlink extra_is
ize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              128016
Block count:              512000
[root@localhost ~]# mkdir /project
[root@localhost ~]# useradd zhangsan
[root@localhost ~]# useradd lisi
[root@localhost ~]# groupadd QQgroup
[root@localhost ~]# gpasswd -a zhangsan QQgroup
Adding user zhangsan to group QQgroup
[root@localhost ~]# gpasswd -a lisi QQgroup
Adding user lisi to group QQgroup
[root@localhost ~]# chown root:QQgroup /project
[root@localhost ~]# chmod 770 /project
[root@localhost ~]# ll -d /project
```

```
[root@localhost ~]# mkdir /project
[root@localhost ~]# useradd zhangsan
[root@localhost ~]# useradd lisi
[root@localhost ~]# groupadd QQgroup
[root@localhost ~]# gpasswd -a zhangsan QQgroup
Adding user zhangsan to group QQgroup
[root@localhost ~]# gpasswd -a lisi QQgroup
Adding user lisi to group QQgroup
[root@localhost ~]# chown root:QQgroup /project
[root@localhost ~]# chmod 770 /project
[root@localhost ~]# ll -d /project
drwxrwx---. 2 root QQgroup 4096 Sep 20 19:52 /project
[root@localhost ~]# useradd pt
[root@localhost ~]# passwd pt
Changing password for user pt.
New password:
BAD PASSWORD: it is WAY too short
BAD PASSWORD: is too simple
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: it is WAY too short
BAD PASSWORD: is too simple
```

```
[root@localhost ~]# ll -d /project
drwxrwx---. 2 root QQgroup 4096 Sep 20 19:52 /project
[root@localhost ~]# useradd pt
[root@localhost ~]# passwd pt
Changing password for user pt.
New password:
BAD PASSWORD: it is WAY too short
BAD PASSWORD: is too simple
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: it is WAY too short
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]# setfacl -m u:pt:rx /project
[root@localhost ~]# ll -d /project
drwxrwx---+ 2 root QQgroup 4096 Sep 20 19:52 /project
[root@localhost ~]# getfacl /project
getfacl: Removing leading '/' from absolute path names
# file: project
# owner: root
# group: QQgroup
user::rwx
user:pt:r-x
```

```
[root@localhost ~]# su -pt
su: invalid option -- 't'
Try `su --help' for more information.
[root@localhost ~]# su - pt
[pt@localhost ~]$ cd /project
[pt@localhost project]$ mkdir a
mkdir: cannot create directory `a': Permission denied
[pt@localhost project]$ _
```

文件的特殊权限

- 文件除了一般权限外，在执行位 x 上还有三种特殊权限
  - SUID(Set UID)：出现在文件所有者的 x 权限上
    - u 有 x 权限时显示为 s，u 没有 x 权限时显示为 S
  - SGID(Set GID)：出现在文件所属群组的 x 权限上
    - g 有 x 权限时显示为 s，g 没有 x 权限时显示为 S
  - SBIT(Sticky Bit)：出现在文件其他用户的x权限上
    - o 有 x 权限时显示为 t，o 没有x权限时显示为 T
  - 特殊权限位值：SUID=4　　SGID=2　　SBIT=1
    (它们放在普通权限前面)

为什么要使用特殊权限

- 例如：
  - 二进制文件 /usr/bin/passwd 的权限是 -rwsr-xr-x，我不是所有者，但我具有 “x“ 权限，我执行它时，获得了它的所有者(即 root)的权限
  - 所以在该二进制程序执行时，我可以用它来读到我平时没有权限访问的 /etc/shadow 文件(-r--------)，从而能更改我自己的密码

SUID

- SUID 必须具备以下几个条件(前提)：
  - 1) 只有可执行的二进制程序才可以设置 SUID
  - 2) 所有者必须对欲设置 SUID 的文件具备可执行(x) 权限
  - 3) 命令执行过程中，其它用户获取所有者的身份
  - 4) SUID 具有时间限制，即完成该程序执行后就消失
- 设置 SUID 权限
  - 语法：chmod u{+|-}s file...　#给文件添加或去除suid权限
- 例：chmod 4755 hello.sh
  或　chmod u+s hello.sh

```
[root@localhost ~]# ll hello.sh
-rwxr-----. 1 root root 45 Sep 19 18:47 hello.sh
[root@localhost ~]# chmod 4755 hello.sh
[root@localhost ~]# ll hello.sh
-rwsr-xr-x. 1 root root 45 Sep 19 18:47 hello.sh
```

SUID 举例

- 1) /usr/bin/passwd 命令拥有 SUID 权限，所以普通用户可以用它修改自己的密码

```
[root@localhost ~]# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 30768 Feb 22  2012 /usr/bin/passwd
```

- 2) /bin/cat 命令没有 SUID权限，所以普通用户不能用它查看 /etc/shadow 文件的内容

```
[root@localhost ~]# ll /bin/cat
-rwxr-xr-x. 1 root root 45224 Nov 22  2013 /bin/cat
```


取消 SUID

- 语法1：chmod u-s 文件名
- 语法2：chmod SUID位为0的权限 文件名

```
[root@localhost ~]# ll hello.sh
-rwsr-xr-x. 1 root root 45 Sep 19 18:47 hello.sh
[root@localhost ~]# chmod u-s hello.sh
[root@localhost ~]# ll hello.sh
-rwxr-xr-x. 1 root root 45 Sep 19 18:47 hello.sh
```


危险的 SUID

- 关键目录应严格控制写权限，比如："/"、"/usr"等
- 用户的密码设置要严格遵守密码三原则
- 对系统中默认应该具有 SUID 权限的文件作一列表，定时检查有没有这之外的文件被设置了 SUID 权限

SGID

- SGID 条件：
  - 针对文件：
    - 可执行的二进制文件
    - 命令执行者(即所属组)对该文件具备 x 权限
    - 在执行时，组身份升级为该程序文件的数组
    - 权限只在执行过程中有效
  - 针对目录：
    - 普通用户对目录具备 r 和 x 权限，才能进入此目录
    - 普通用户在此目录中的有效组会变成此目录的所属组
    - 如普通用户对该目录具备 w 权限，新建的文件的默认所属组为该目录的所属组

设置和取消 SGID

- 设置 SGID
  - 语法1：chmod 2xxx 文件名
  - 语法2：chmod g+s 文件名
- 取消 SGID
  - 语法1：chmod 0xxx 文件名
  - 语法2：chmod g-s 文件名

SGID 举例

- [root@localhost ~]# ll /usr/bin/locate
  -rwx--s--x. 1 root slocate 40496 6月 10 2014 /usr/bin/locate
- [root@localhost ~]# ll /var/lib/mlocate/mlocate.db
  -rw-r-----. 1 root slocate 6306909 7月 30 19:15 /var/lib/mlocate/mlocate.db
- 用普通用户进行locate查看：
  - [niesh@localhost root]$ locate mlocate.db
    /usr/share/man/man5/mlocate.db.5.gz
- 去掉 locate 的 s 权限：
  - [root@localhost ~]# chmod g-s /usr/bin/locate
  - [root@localhost ~]# ll /usr/bin/locate
    -rwx--x--x. 1 root slocate 40496 6月 10 2014 /usr/bin/locate
- [niesh@localhost root]$ locate mlocate.db
  locate: 无法执行 stat () `/var/lib/mlocate/mlocate.db': 权限不够
- 也就是：当执行 locate 命令时，普通用户 niesh 自动升级为 slocate 的组成员

SBIT (即粘滞位)

- 作用：
  - 只对目录有效
  - 普通用户对该目录有 w 和 x 权限
  - 若没有粘滞位，则普通用户可以对目录下的文件/子目录进行删除操作(因为普通用户对目录具有 w 权限)，包括其它用户建立的目录/文件；但若赋了SBIT，则普通用户只能删除自己创建的文件/目录，而不能删除不属于自己的文件/目录！

设置和取消 SBIT

- 设置 SBIT
  - 语法1：chmod 1xxx 目录名
  - 语法2：chmod o+t 目录名
- 取消 SBIT
  - 语法1：chmod 0xxx 目录名
  - 语法2：chmod o-t 目录名

SBIT 举例

- 以 /tmp 为例：
  - [niesh@localhost tmp]$ ll -d /tmp/ #查看 /tmp 的权限
    drwxrwxrwt. 8 root root 4096 7月 30 19:40 /tmp/
  - 用其它用户创建两个文件：
    - [Jimmy@localhost tmp]$ touch test-file
    - [Jimmy@localhost tmp]$ mkdir test-dir
    - [Jimmy@localhost tmp]$ ll

    总用量 0
    drwxrwxr-x. 2 Jimmy Jimmy 6 7月 30 19:44 test-dir
    -rw-rw-r--. 1 root Jimmy 0 7月 30 19:44 test-file

SBIT 举例(续)

- 以 /tmp 为例：
  - 切换到另外一个用户niesh:
    - [niesh@localhost tmp]$ ll

    总用量 0
    drwxrwxr-x. 2 Jimmy Jimmy 6 7月 30 19:44 test-dir
    -rw-rw-r--. 1 root Jimmy 0 7月 30 19:44 test-file
  - 在 niesh用户下，删除/tmp目录下的文件：
    - [niesh@localhost tmp]$ rm -rf test-dir/ test-file

    rm: 无法删除"test-dir/": 不允许的操作
  - 切换到root，去掉/tmp的粘滞位：
    - [niesh@localhost tmp]$ su -

    密码：
    上一次登录：日 7月 30 19:43:21 CST 2017pts/0 上

SBIT 举例(续)

- 以 /tmp 为例：
  - [root@localhost ~]# chmod o-t /tmp/
  - [root@localhost ~]# ll -d /tmp/

    drwxrwxrwx. 9 root root 4096 7月 30 19:48 /tmp/
  - 切换到普通用户niesh，再次删除/tmp下的文件：
  - [niesh@localhost root]$ rm -rf /tmp/test-dir/ /tmp/test-file
  - [niesh@localhost root]$ ll /tmp/

    总用量 0

不可改变位权限：chattr

- 这个权限是限制修改文件内容的，如 SBIT 权限可以保护文件不被删除，却不能保证文件内容不被修改
- 语法：chattr [+-=] [选项] 文件或目录名
- 功能：在 SBIT 基础上实现更多功能
  +：增加权限 -：删除权限 =：等于某权限
- 选项：
  - i：如果对文件设置了 i 属性，那么不允许对文件进行删除，改名，也不能添加和修改数据；如果对目录设置了 i 属性，那么只能修改目录下文件的数据，但不允许建立和删除文件
  - a：如果对文件设置了 a 属性，那么只能在文件中增加数据，但是不能删除也不能修改数据；如果对目录设置了 a 属性，那么只允许在目录中建立和修改文件，但是不允许删除
- 注意：对 root 用户生效

查看文件系统属性：lsattr

- 语法：lsattr 选项 文件名
- 选项：
  - -a 显示所有文件和目录
  - -d 若目标是目录，仅列出目录本身的属性，而不是子文件的属性