# 注意：本章内容理解即可

# 第1章 特殊权限概述

## 1.什么是特殊权限

上节课我们详细讲解了Linux系统9位基本权限位，但是通过查看一些系统关键的文件可以发现上节课没有见过的权限字符，比如/usr/bin/passwd文件

```
[root@linux ~]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 6月  10 2014 /usr/bin/passwd
```

可以发现，本该出现在x的位置上出现了从来没有见过的s！这是什么权限？其实这就是传说中的特殊权限suid。
事实上除了我们已经学过的基本9位权限之外，实际上Linux权限位还有额外的3位权限位，总共是12位权限位。
他们分别是：

```
suid (setuid)
sgid (setgid)
sbit (sticky)
```

## 2.特殊权限位对比说明表

| 类别 | suid | sgid | sticky |
| --- | --- | --- | --- |
| 字符表示 | S | S | T |
| 占据位置 | 基本权限位x | 用户组基本权限位x | 其他用户基本权限位x |
| 基本权限位有x | s | s | t |
| 数字表示 | 4 | 2 | 1 |
| 八进制表示 | 4000 | 2000 | 1000 |
| 生效对象 | 针对用户位 | 针对用户组 | 其他用户 |

# 第2章 特殊权限suid

## 1.什么是suid特殊权限

```
suid(setuid)位通过S字符标识，存在于基本权限的用户权限位的x权限对应的位置。
如果用户权限位对应的x权限位上有x权限，则suid就用小写的s标识，如果没有x权限，就用大写的S表示。
suid的s对应的数字权限为4，完整权限用八进制数4000表示。
suid位也是通过chmod命令进行设置的，可以利用字符权限以及数字权限来实现.
```

示例代码如下：

```
[root@linux ~]# touch test.txt
[root@linux ~]# ls -l test.txt
-rw-r--r-- 1 root root 31 3月  30 17:20 test.txt
[root@linux ~]# chmod u+s test.txt      #在用户位增加suid权限
[root@linux ~]# ls -l test.txt
-rwSr--r-- 1 root root 31 3月  30 17:20 test.txt     #权限位为S，下面截图可以看到文件名背景是红色的
```

效果如下：

```
[root@linux ~]# touch test.txt
[root@linux ~]# ls -l test.txt
-rw-r--r-- 1 root root 31 3月  30 17:20 test.txt
[root@linux ~]# chmod u+s test.txt
[root@linux ~]# ls -l test.txt
-rwSr--r-- 1 root root 31 3月  30 17:20 test.txt
[root@linux ~]#
```

# 2.为什么需要suid特殊权限

在Linux系统中，有时执行某个命令(例如普通用户使用passwd修改自身账号密码)时，需要对另一个文件(/etc/shadow)进行操作，而执行passwd修改的文件/etc/shadow又是普通用户没有权限进行操作的。

例如，修改用户密码的命令passwd，该命令文件的所有者和用户组都是root，但是/etc/shadow文件的权限极低，不允许任何普通用户及属主用户访问,具体信息如下:

```
[root@linux ~]# ls -l /etc/shadow
---------- 1 root root 1472 3月  26 18:34 /etc/shadow

[root@linux ~]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 6月  10 2014 /usr/bin/passwd
```

这样的权限，对于普通用户是没有办法更改的，但是系统还需要对普通用户开放自己更改自身账号密码的功能，因此就必须要让普通 用户有权限修改shadow文件，这时就可以通过对passwd命令设置suid 位来解决。

# 3.suid的应用场景

```
简单地说: suid的作用就是让普通用户可以在执行某个设置了suid位的命令或程序时，拥有与root管理员一样的身份和权限(默认情况)。

以/etc/shadow文件举例：
据基本权限的知识，oldboy用户无法修改shadow文件，但是因为passwd具有suid权限。因此，当oldboy用户执行passwd时，就可以取得root管理员的身份和权限(实际上是获取passwd对应用户的身份)，所以就可以修改原本oldboy没权限修改的/etc/shadow了(修改过程中是以root身份操作的，这就是suid的作用)。

聪明的同学就想到了，那是否意味着普通用户就可以使用passwd命令修改任何账户的密码了呢？
其实不然，Linux系统已经考虑到这种情况了，所以普通用户的passwd命令只能修改自己的密码，不能修改其他用户的密码。
```

效果如下所示：

```
[root@linux ~]# su - oldya
上一次登录：一 3月 29 20:40:35 CST 2021pts/0 上
[oldya@linux ~]$ passwd root
passwd：只有根用户才能指定用户名。      #普通用户修改其他用户密码会提示passwd命令只有根用户才能指定用户名
```

# 3.suid核心知识

```
1.suid的功能是是针对二进制命令或程序的，不能用在Shell等类似脚本文件上。
2.用户或属主对应的前三位权限的x位上，如果有s(S)则表示具备suid权限。
3.suid的作用就是让普通用户可以在执行某个设置了suid位的命令或程序时，拥有与命令对应属主(一般为root管理员)一样的身份和权限(默认)。
4.二进制命令程序需要具有可执行权限x配合才能进行相关操作。
5.suid对应的身份和权限仅在程序命令的执行过程中才有效。
6.suid是一把双刃剑，是一个比较危险的功能，对系统安全存在一定的威胁，企业里用户授权可以使用sudo等替代sgid功能。
7.在进行安全优化时，系统中默认设置了suid权限的命令要取消掉。
```

# 4.suid配置案例

如果没有配置sudo,普通用户是不能删除其他用户的文件的，演示如下：

```
[root@linux ~]# touch /opt/root.txt
[root@linux ~]# ll /opt/root.txt
-rw-r--r-- 1 root root 0 3月  30 17:52 /opt/root.txt
[root@linux ~]# su - oldya
[oldya@linux ~]$ rm -rf /opt/root.txt
rm: 无法删除"/opt/root.txt": 权限不够
```

下面我们使用root账户给rm命令授权suid权限，步骤如下：

```
[root@linux ~]# chmod u+s /bin/rm
[root@linux ~]# ll /bin/rm
-rwsr-xr-x. 1 root root 62952 10月 31 2018 /bin/rm
```

然后再切换到普通用户进行删除文件测试，发现可以删除了。步骤如下：

```
[root@linux ~]# su - oldya
上一次登录：二 3月 30 17:53:11 CST 2021pts/0 上
[oldya@linux ~]$ rm -rf /opt/root.txt
```

总结：

```
对rm设置了suid位以后，任何用户执行rm都是以root 的身份和权限来进行的，由此可见这样操作是很不安全的。
所以，建议大家不要在在企业中使用suid功能。
```

# 5.拓展-如何查找Linux系统里具有suid权限的文件

```
find / -type f -perm -4000 -ls
```

# 第3章 特殊权限sgid

## 1.什么是sgid特殊权限

```
1.对于二进制命令或者程序来说，sgid的功能与suid基本相同，唯一的区别是，suid是获得命令所属用户的身份和权限，而sgid是获得命令所属用户组的身份和权限。
2.setgid位主要用于目录中，在为某个目录设置了setgid位以后，在该目录中新创建的文件具有该目录的所属组权限，而不是创建该文件的用户的默认所有者。这就使得在多个用户之间共享一个目录中的文件变得简单。
```

## 2.sgid作用

```
1)与suid不同的是，sgid既可以针对文件，也可以针对目录进行设置
2)sgid的权限是针对用户组权限位的。
```

对于文件来说：

```
1)sgid仅对二进制命令及程序有效。
2)二进制命令或程序，也需要有可执行权限x的配合。
3)执行命令的任意用户可以获得该命令在程序执行期间所属组的身份和权限。
```

对于目录来说：

```
1)Linux里默认情况下所有用户创建文件，默认用户和组都是自身。
2)sgid可以让用户在此目录下创建的文件和目录具有与此目录相同的用户组设置。
```

## 3.sgid配置案例

下面就来看一道RHCE认证考试题(了解即可)。

题目需求：

```
1.创建共享目录/home/admins
2.要求属组为adminuser，adminuser组成员对admins目录有写入、读取和执行的权限
3.其他所有用户均没有任何权限(root除外)
4.在/home/admins目录中创建的文件会自动继承adminuser组的权限。
```

解答步骤：

```
[root@linux ~]# mkdir /home/admins
[root@linux ~]# groupadd adminuser
[root@linux ~]# ls -ld /home/admins
drwxr-xr-x 2 root root 6 3月  30 18:25 /home/admins
[root@linux ~]# chown :adminuser /home/admins/
[root@linux ~]# ls -ld /home/admins
drwxr-xr-x 2 root adminuser 6 3月  30 18:25 /home/admins
[root@linux ~]# touch /home/admins/oldboy.txt
[root@linux ~]# ls -l /home/admins/
总用量 0
-rw-r--r-- 1 root root 0 3月  30 18:26 oldboy.txt
[root@linux ~]# chmod 2770 /home/admins
[root@linux ~]# ls -ld /home/admins
```

```
drwxrws--- 2 root adminuser 24 3月  30 18:26 /home/admins
[root@linux ~]# touch /home/admins/newfile.txt
[root@linux ~]# ls -l /home/admins/
总用量 0
-rw-r--r-- 1 root adminuser 0 3月  30 18:27 newfile.txt
-rw-r--r-- 1 root root      0 3月  30 18:26 oldboy.txt
```

# 第4章 特殊权限sbit

## 1.sbit粘滞位作用

```
粘滞位sbit（sticky bit）的功能现在已经很少用了，不过对于 像/tmp目录这样的，因为其是整个系统用户的临时文件存放地，因此还是有点意义的。
如果查看/tmp的目录属性，就会发现这个目录权限大的离谱。

[root@linux ~]# ll -d /tmp/
drwxrwxrwt. 16 root root 4096 3月  30 19:36 /tmp/

一旦给目录赋予了粘滞位，那么这个目录除了root用户可以删除所有文件，其他用户都只能删除自己建立的文件，不能删除其他用户的文件。
```

总结如下：

```
1.只对目录生效
2.一个目录即使它的所有权限都开放了，即权限为rwxrwxrwx，但设置了粘滞位，那么除非目录的属主和root用户有权限删除它，除此之外其他用户是不能删除这个目录的。
```

## 2.sbit粘滞位配置

```
chmod 1755  /tmp
chmod o+t /tmp
```

# 第5章 文件特殊属性命令

## 1.更改文件特殊属性chattr

命令说明：

```
chattr命令用于改变文件的扩展属性。与chmod命令相比，chmod 只是改变了文件的读、写、执行权限，更底层的属性控制是由chattr 来改变的。
```

命令格式：

```
chattr [选项] [模式] [文件或目录]
```

选项：

```
-R  递归更改目录属性
-V  显示命令的执行过程
```

模式:

```
+   增加参数
-   移除参数
=   更改为指定的参数
A   告诉系统不要修改这个文件的最后访问时间
a   只能向文件里追加数据，不能删除
i   设定的文件不能被删除，改名，写入或新增内容
```

+a 只能追加命令实践：

```
[root@linux ~]# touch test.txt
[root@linux ~]# lsattr test.txt
---------------- test.txt
[root@linux ~]# chattr +a test.txt
[root@linux ~]# lsattr test.txt
-----a---------- test.txt
[root@linux ~]# echo 111 >> test.txt
[root@linux ~]# echo 222 >> test.txt
[root@linux ~]# echo 111 > test.txt
-bash: test.txt: 不允许的操作
[root@linux ~]# rm -rf test.txt
rm: 无法删除"test.txt": 不允许的操作
```

+i 文件加锁命令实践

```
[root@linux ~]# chattr +i test.txt
[root@linux ~]# lsattr test.txt
----ia---------- test.txt
[root@linux ~]# echo 333 >> test.txt
-bash: test.txt: 权限不够
[root@linux ~]# echo > test.txt
-bash: test.txt: 权限不够
[root@linux ~]# rm -rf test.txt
rm: 无法删除"test.txt": 不允许的操作
```

应用场景：

```
避免恶意删除.bash_history历史记录文件或者重定向到/dev/null，又因为系统需要向这个文件写入历史记录，因此采用追加模式，只增不减。

chattr +a .bash_history
```

# 2.查看文件特殊属性lsattr

命令说明：

```
lsattr命令可用于查看文件扩展属性。
```

命令格式：

```
lsattr [选项] [文件或目录]
```

选项：

```
-R  递归查看目录的拓展属性
-a  显示所有文件包括隐藏文件的拓展属性
-d  显示目录的拓展属性
```

查看文件的拓展属性：

```
[root@linux ~]# lsattr test.txt
----ia---------- test.txt
```

查看目录的拓展属性：

```
[root@linux ~]# lsattr -d /tmp/
---------------- /tmp/
```

# 第6章 默认权限掩码umask

## 1.什么是umask

```
通过上一课的学习我们了解到Linux里创建目录的默认权限是755,创建文件的权限是644。
那么为什么目录是755，文件是644呢？实际上是因为有umask在控制默认创建的权限。
```

## 2.umask如何计算

```
umask是通过八进制的数值来定义用户创建文件或目录的默认权限的。
系统会根据预先设定的umask值计算出默认情况下创建的文件或目录权限。
umask默认是022权限，当我们创建一个目录时，正常情况下的权限应该是777，然后减去umask的值，所以默认创建的目的权限是777-022=755。
同理，文件的默认权限应该是666，然后减去umask的值为666-022=644
```

## 3.基于文件的计算-有奇偶之分

```
创建文件默认最大的权限为666(-rw-rw-rw-)，其默认创建的文件没有可执行权限x位。
对于文件来说，umask的设置是在假定文件拥有八进制666的权限上进行的，文件的权限就是666减umask(umask的各个位数字也不能大于6，比如077就不符合条件)的掩码数值。如果得到的3位数字每一位都是偶数，那么这就是最终结果;如果有若干位的数字是奇数，那么这个奇数需要加1变成偶数，最后得到全是偶数的结果。
```

举例说明：

1）假设umask值为022(所有位均为偶数)，那么文件的对应权限计算式为：

```
  666
- 022
------
  644
```

2）假设umask值为045(其他用户组位为奇数)，那么文件的对应权限计算式为:

```
  666
- 045
------
  621
+ 001
------
  622
```

## 4.基于目录的计算-没有奇偶之分

1）假设umask值为022(所有位均为偶数)，那么目录的对应权限计算式为：

```
  777
- 022
------
  755
```