LINUX ACCESS MODES AND STICKY BITS EXAMPLES

Sticky Bits - Examples

Access modes
When you load your terminal or SSH in, the new shell process runs using your user and group ids. These permissions determine your access to files on the system. This generally means that you can't access files that belong to others and can't access various system files. The programs you start inherit your user id. This means that they can't for example access any filesystem objects for which you haven't been given permission.

The /etc/passwd file is a good example of this. This file can't be changed by normal users directly, it can only be changed by the root user. However, the caveat is that normal users need to be able to modify /etc/passwd when they have to change their password. This issue is solved in Linux via the permissions modes suid (set user id) and sgid (set group id).

```
raspberry-VirtualBox ~ # ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 45420 Jul 26  2013 /usr/bin/passwd
```

When the suid access mode is set for an executable program, it will run as if it had been started by the file's owner rather than by the user who really started it. The sgid access modes works in a similar way. The executable program will run as if the initiating user belonged to the file's group rather than to its own group.

These access modes are used to allow users on a Linux machine to run programs with "temporarily elevated privileges" thus allowing them to perform a task requiring extra permissions that they would otherwise not have.

Note that in place of an x (executable) in the user's permissions, there's an s. This tells us that the suid and executable bits are set. So when passwd runs, it will execute as if the root user (i.e. the Owner) had launched it with full superuser access (root), rather than that of the user who ran it. Because passwd runs with root access, it can modify /etc/passwd. The suid and sgid bits occupy the same space as the x bits for user and group in a long directory listing. If the file is executable, the suid or sgid bits, if set, will be displayed as lowercase s, otherwise they are displayed as uppercase S. i.e.

If suid or sgid bits are set then: Lowercase “s” => the file is executable UPPERCASE “S” => the file is NOT executable

Setting suid And sgid

In the octal number format, suid has the value 4 in the first (high order) digit, while sgid has the value 2.

Directories And sgid

A directory has the sgid mode enabled means that any files or directories created within it will inherit the group id of the directory. This can be very useful for directory that is used by a group of people working on say the same project...

sgid Sticky Bit Example

Below shows how user “greg” could set up a directory that all users of the development group could use, along with an example of how user “gretchen” could create a file in the directory. As created, the file gretchen.txt allows groups members to write to the file.

Create user “greg” and user “gretchen”:

```
useradd -m greg
useradd -m gretchen
```
=> Set their passwords to “123”
```
passwd greg
passwd gretchen
cd /home/greg/
mkdir Our_Big_Project_Together
```

Setup a group called “developer” and make these new users members of this group:

```
raspberry-VirtualBox ~ # groupadd developer
raspberry-VirtualBox ~ # sudo usermod -a -G developer gretchen
raspberry-VirtualBox ~ # sudo usermod -a -G developer greg
```

The first digit in the above mode number is used to set setuid, setgid, or the sticky bit. Each remaing digit set permission for the owner, group, and world is as follows:

sticky/setgid/setuid Table

sticky/setgid/setuid where: 1 = sticky bit 2 = setgid (set group id) 4 = setuid (set user id)

Login as User “greg”, create a new directory in User greg’s home directory called “Our_Project_Together”

```
mkdir Our_Project_Together
```

```
gretchen@raspberry-VirtualBox /home/greg $ su - greg
Password:
$ bash
greg@raspberry-VirtualBox ~ $ cd /home/greg/
```

```
greg@raspberry-VirtualBox ~ $ mkdir Our_Project_Together
greg@raspberry-VirtualBox ~ $ ls -l
total 4
drwxrwxr-x 2 greg greg 4096 Jan 15 10:19 Our_Project_Together
```

Now set the group name of the directory to “developer”:

```
greg@raspberry-VirtualBox ~ $ chown :developer Our_Project_Together/
greg@raspberry-VirtualBox ~ $ ls -l
total 4
drwxrwxr-x 2 greg developer 4096 Jan 15 10:19 Our_Project_Together
```

Now set the “setgid” (Set Group ID) bit. We need in place of the “x” in the group to be an “s” (LOWER CASE “s” NOT UPPER CASE “S”). Therefore to generate say rwx rwx r-x would be merely 775. Based on this we just have to change the group “x” component via the setgid (set group id).
Now when Gretchen (who is a member of the group called “developer”) enters this /home/greg/Our_Project_Together/ directory, when she creates a file, that file inherits the group name of the directory (i.e. “developer”) – see in orange below in the Terminal output:

sticky/setgid/setuid where: 1 = sticky bit = t 2 = setgid (set group id) = s or S 4 = setuid (set user id) = s or S	Owner	Group	Other
	rwx	rws	r-x
2	**7**	**7**	**5**

```
greg@raspberry-VirtualBox ~ $ cd /home/greg/
greg@raspberry-VirtualBox ~ $ ls -l
total 4
drwxrwxr-x 2 greg developer 4096 Jan 15 10:52 Our_Project_Together
greg@raspberry-VirtualBox ~ $ chmod 2775 Our_Project_Together/
greg@raspberry-VirtualBox ~ $ ls -l
total 4
drwxrwsr-x 2 greg developer 4096 Jan 15 10:52 Our_Project_Together
greg@raspberry-VirtualBox ~ $ su - gretchen
Password:
$ bash
gretchen@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 0
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ touch gretchen.txt
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 0
-rw-rw-r-- 1 gretchen developer 0 Jan 15 10:53 gretchen.txt
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $
```

However, as usual, if gretchen creates a file in her own home directory then the Owner and Group name of the file is both “gretchen”:

```
gretchen@raspberry-VirtualBox /home/greg $ cd /home/gretchen/
gretchen@raspberry-VirtualBox ~ $ touch AnothergretchenFile.txt
gretchen@raspberry-VirtualBox ~ $ ls -l
total 0
-rw-rw-r-- 1 gretchen gretchen 0 Jan 15 11:12 AnothergretchenFile.txt
```

Any member of the development group can now currently create files in user greg's Our_Project_Together directory.
Currently, the file gretchen.txt allows members of the group “developer” to write to her file Gretchen.txt because the group permissions are set to rw- (read and write but not execute). That is to say, the Owner (gretchen) can rw- (Read,write) the file, the Group can rw- (Read,write) and the Others cant do anything --- .

```
-rw-rw-r-- 1 gretchen developer 0 Jan 15 11:12 gretchen.txt
```

To prevent this so that other members of group “developer” can’t write to her file Gretchen.txt but can read it, do the following (rw-r-- ---):

Owner	Group	Other
rw-	r--	---
6	4	0

```
chmod 640 gretchen.txt
```

```
raspberry-VirtualBox Our_Project_Together # su - gretchen
$ bash
gretchen@raspberry-VirtualBox ~ $ pwd
/home/gretchen
gretchen@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ chmod 640 gretchen.txt
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 4
-rw-r-----  1 gretchen developer 10 Jan 15 11:25 gretchen.txt
```

Now add the group “developer” to User “fred”:

```
gretchen@raspberry-VirtualBox ~ $ su - root
Password:
raspberry-VirtualBox ~ # sudo usermod -a -G developer fred
```

Now User “fred” is a member of the group called “developer”:

```
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - root
Password:
raspberry-VirtualBox ~ # groups fred
fred : fred sudo defjam developer
```

User “fred” can’t write to this Gretchen.txt file, but he can read it:

```
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - fred
Password:
$ bash
fred@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 4
-rw-r----- 1 gretchen developer 10 Jan 15 11:25 gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ echo "some text ..." >> gretchen.txt
bash: gretchen.txt: Permission denied
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ cat gretchen.txt
something
```

Note: I can easily give User “fred” the ability to write to this file via rw- rw- --- i.e.

Owner	Group	Other
rw-	rw-	---
6	6	0

```
chmod 660 gretchen.txt
```

```
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - fred
Password:
$ bash
fred@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 4
-rw-rw---- 1 gretchen developer 10 Jan 15 11:25 gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ echo "some text ..." >> gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ cat gretchen.txt
something
some text ...
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ rm gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 0
```

To set the setuid (set user id) so that anyone can execute the script (as if they were the root user) via ./script.sh

```
nano script.sh
```

```
#!/bin/bash
echo “Hi”
exit 0
```

rwx r-x r-x => 755
Therefore, from the table above, we place the 4 in front

```
chmod 4755 script.sh
```

```
-rwsr-sr--  1 root  root  37 Aug  8 13:03 scripttest.sh
```

Linux ignores the setuid bit on all interpreted executables (i.e. executables starting with a #! line).

The Sticky Bit

We have just demonstrated that anyone with the write permission to a directory can delete files in it. This might be acceptable for the above situation whereby a group of people working on a project together need this facility, however this situation might not be a such a great idea for say globally shared file space such as the /tmp directory.

The sticky bit "t" solves this problem. It is represented symbolically by t and numerically as an Octal 1.

```
sticky/setgid/setuid
where:
1 = sticky bit = t
2 = setgid (set group id) = s or S
4 = setuid (set user id) = s or S
```

The "t" is displayed in a long directory listing where the executable "x" would normally be seen for the Other user. This has the same meaning as suid and sgid in terms of the upper and lower case nomenclature. If set for a directory, it permits only the owning user or the superuser (root) to delete or unlink a file. Below shows how user “greg” could set the sticky bit on his Our_Projects_Together directory.

rwx rws r-t

For rwx rwx r-x we would need 775, for rwx rws r-x (i.e. “s” merely replaces an “x” in the group column for setgid) this is 2775 as shown above (where the “2” sets the setgid). i.e. rwx rws r-x
Now to also have the sticky bit “t” we add the left most column numbers i.e. we need the “2” for the setgid and we also need a “1” for the sticky bit i.e. 2+1 = 3

```
chmod 3775 Our_Projects_Together
```

sticky/setgid/setuid where: 1 = sticky bit = t 2 = setgid (set group id) = s or S 4 = setuid (set user id) = s or S	Owner	Group	Other
	rwx	rws	r-t
3	**7**	**7**	**5**

```
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - greg
Password:
$ bash
greg@raspberry-VirtualBox ~ $ cd /home/greg/
greg@raspberry-VirtualBox ~ $ ls -l
total 4
drwxrwsr-x 2 greg developer 4096 Jan 15 11:53 Our_Project_Together
greg@raspberry-VirtualBox ~ $ chmod 3775 Our_Project_Together/
greg@raspberry-VirtualBox ~ $ ls -l
total 4
drwxrwsr-t 2 greg developer 4096 Jan 15 11:53 Our_Project_Together
```

As explained above, the sticky bit being set here means that If set for a directory, it permits only the owning user or the superuser (root) to delete or unlink a file:

```
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - fred
Password:
$ bash
fred@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 0
-rw-rw-r-- 1 gretchen developer 0 Jan 15 12:17 gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ rm gretchen.txt
rm: cannot remove âgretchen.txtâ: Operation not permitted
```

So, even though User “fred” is a member of the group “developer” and that group has rw- privalages here for Gretchen.txt, because the Our_Projects_Together directory’s sticky bit is set, User “fred” can’t delete this Gretchen.txt file. User “fred” can still write to the file as normal here:

```
gretchen@raspberry-VirtualBox ~ $ su - fred
Password:
$ bash
fred@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 0
-rw-rw-r-- 1 gretchen developer 0 Jan 15 12:28 gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ echo "more text to add here..." >> gretchen.txt
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ cat gretchen.txt
more text to add here...
```

Although User “gretchen” can remove it too:

```
fred@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - gretchen
Password:
$ bash
gretchen@raspberry-VirtualBox ~ $ cd /home/greg/Our_Project_Together/
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ rm gretchen.txt
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ ls -l
total 0
```

User “greg” can delete it too:

```
gretchen@raspberry-VirtualBox /home/greg/Our_Project_Together $ su - greg
Password:
$ bash
greg@raspberry-VirtualBox ~ $ cd Our_Project_Together/
greg@raspberry-VirtualBox ~/Our_Project_Together $ ls -l
total 0
-rw-rw-r-- 1 gretchen developer 0 Jan 15 12:20 gretchen.txt
greg@raspberry-VirtualBox ~/Our_Project_Together $ rm gretchen.txt
greg@raspberry-VirtualBox ~/Our_Project_Together $ ls -l
total 0
```

NOTE: Also, you can see that the root directory /tmp has this bit is set by default:

```
ls -l /
```

```
drwxrwxrwt   7 root root 69632 Jan 15 11:57 tmp
```

NOTE: The setuid permission set on a directory is ignored on UNIX and Linux systems.

setuid On Executables

When setuid is set on an executable file, normal users who have permission to execute this file get their privileges elevated to that of the user who owns the file (often the root user).
Note that many Linux operating systems ignore setuid when applied to executable shell scripts. This is due to obvious security issues.

Key points w.r.t setuid:

• The setuid permission set on a directory is ignored on UNIX and Linux systems
• Many operating systems ignore the setuid attribute when applied to executable shell scripts.

Another Example With setgid

I want a directory whereby only members of group “developer” can view inside it. I also want this directory to force new files and subdirectories created within it to inherit its group ID rather than the primary group ID of the user who created the file:
d wrx wrx ---

sticky/setgid/setuid where: 1 = sticky bit = t 2 = setgid (set group id) = s or S 4 = setuid (set user id) = s or S	Owner	Group	Other
	rwx	rws	---
2	**7**	**7**	**0**

```
raspberry-VirtualBox greg #  cp -a Our_Project_Together/ New_Our_Project_Together/
raspberry-VirtualBox greg # ls -l
total 8
drwxrwsr-t 2 greg developer 4096 Jan 15 13:40 New_Our_Project_Together
drwxrwsr-t 2 greg developer 4096 Jan 15 13:40 Our_Project_Together
```

```
raspberry-VirtualBox greg # chmod 2775 Our_Project_Together/
```

```
raspberry-VirtualBox greg # ls -l
total 8
drwxrws--- 2 greg developer 4096 Jan 15 13:40 New_Our_Project_Together
drwsrwsr-x 2 greg developer 4096 Jan 15 13:40 Our_Project_Together
```

e.g. Now User “Jon” who isn’t a member of Group “developer” CAN’T see inside this directory.
If user “gretchen” creates a file inside this New_Our_Project_Together directory, then like above we get:

```
gretchen@raspberry-VirtualBox /home/greg/New_Our_Project_Together $ ls -l
total 4
-rw-rw-r-- 1 greg      developer  0 Jan 15 13:40 a
-rw-rw-r-- 1 gretchen developer  0 Jan 15 13:54 anotherfilebygretchen.txt
-rw-rw-r-- 1 greg      developer  0 Jan 15 13:40 filebygreg.txt
-rw-rw-r-- 1 gretchen developer 25 Jan 15 12:29 gretchen.txt
-rw-rw-r-- 1 gretchen developer  0 Jan 15 13:39 somefile.txt
```

i.e. gretchen’s new file “anotherfilebygretchen.txt” inherits the group ID of the directory New_Our_Project_Together (i.e. it inherits the group “developer)” here.

RemovingSticky Bits

If for example you have the following directory called Testy with these sticky bit permissions set:

```
drwxrwsrwx  2 root     root     4096 Aug  3 11:59 Testy
```

Note that the following WILL NOT remove it:

```
sudo chmod 777 Testy/
```

This however will remove it:

```
sudo chmod g-s Testy/
```

Terminal Output:

```
drwxrwxrwx  2 root     root     4096 Aug  3 11:59 Testy
```

The reason for this is that you can set or clear the bits with symbolic modes such as say u+s and g-s, and you can set (but not clear) the bits with a numeric mode.