The UNIX s-bit and t-bit (sticky bit)

Unix and Linux systems (including Mac OS X and other POSIX compliant systems) have a distinct system for controlling access to files and directories. Since devices such as disks, ports, etc. also have file names (under /dev) you control access to them the same way.

This systems works by assigning a user and a group to every file. Then users of that file are put into one of three classes:

- the owner (the process' UID matches the user of the file),
- the group (not the owner, but the process' GID is a member of the file's group),
- and other (everyone else).

For each class of users there are three possible permissions that can be granted:

- read,
- write, and
- execute.

In addition to these standard permissions there are three standard attributes that can be set on any file or directory (these are commonly also referred to as permissions):

- The set user ID (or SUID),
- the set group ID (or SGID), and
- the text (or sticky) attributes.

Finally, there are non-standard attributes and additional permissions (access control lists or ACLs) that may or may not be available on some systems.

All permission and attribute information about a file is kept in the file's "inode". Only the owner of a file (or the root user) can modify information in the inode such as the permission bits and group. Note the owner needs no permissions set to change permissions; it is enough to be the owner. Also, only the super-user root can change the owner of a file on most Unix and Linux systems.

A presentation of 9 codes (or keys) or 3 digits show file's or directory's access permission:

owner			Group			others		
r|-	w|-	x|-	r|-	w|-	x|-	r|-	w|-	x|-
4	2	1	4	2	1	4	2	1
shows r|w|x|- equals bit 0-7 [4+2+1]			shows r|w|x|- equals bit 0-7 [4+2+1]			shows r|w|x|- equals bit 0-7 [4+2+1]		

In addition to these nine mode bits (r, w, and x, for each of three categories of owner, group, and others), there are three others:

- the set User ID (SUID or setuid), the set Group ID (SGID or setgid), called s-bit
- and the sticky (or text) bit, called t-bit.

The effect of these three bits differs for plain files and directories and differ between different versions of UNIX. You should check the manual page man sticky to find out about your system! The following is common behaviour under most unix systems.

Octal	Text	Name
4000	chmod u+s	setuid bit
2000	chmod g+s	setgid bit
1000	chmod +t	sticky bit

s-bit

Most programs run with the user and group access rights of the user who invoked them. Program owners can associate the access rights of the user who invoked them by making the program a **setuid** or **setgid** program; that is, a program with the setuid or setgid bit set in its permissions field. When that program is run by a process, the process acquires the access rights of the owner of the program. A **setuid** program runs with the access rights of its owner, while a **setgid** program has the access rights of its group, and both bits can be set according to the permission mechanism.

Setuid: Who runs the program owns perrmissions and rights of the file owner.
Setgid: Who run s the program owns the group execution rights.

Although the process is assigned the additional access rights, these rights are controlled by the program bearing the rights. Thus, the setuid and setgid programs allow for user-programmed **access controls in which access rights are granted indirectly**. The **program acts as a trusted subsystem**, guarding the user's access rights.

Although these programs can be used with great effectiveness, there is a security risk if they are not designed carefully. In particular, the program must never return control to the user while it still has the access rights of its owner, because this would allow a user to make unrestricted use of the owner's rights.

In BSD unix, if the setgid bit is set on a directory then any new files created in that directory assume the group ownership of the parent directory and not the login group of the user who created the file. This is standard policy under system 5.

In AIX for security reasons, the operating system does not support setuid or setgid program calls within a shell script.

t-bit

A directory for which the **sticky bit** is set restrict the deletion of files within it. A file or directory inside a directory with the **t-bit** set can only be deleted or renamed by its owner or the superuser. This is useful for directories like the mail spool area and /tmp which must be writable to everyone, but should not allow a user to delete another user's files.

(AIX and Ultrix) If an executable file is marked with a sticky bit, it is held in the memory or system swap area. It does not have to be fetched from disk each time it is executed. This saves time for frequently used programs like ls.

The Linux kernel ignores the sticky bit on files.

(Solaris) If a non-executable file is marked with the sticky bit, it will *not* be held in the disk page cache -- that is, it is never copied from the disk and held in RAM but is written to directly. This is used to prevent certain files from using up valuable memory.

On some systems (e.g. ULTRIX), only the superuser can set the sticky bit. On others (e.g. SunOS) any user can create a sticky directory.

In AIX The sticky bit is visible for running processes through the ps –[ef]P command.

Presentation

The **group-execute** permission character is **s** if the file has **set-group-ID** mode. The **user-execute** permission character is **s** if the file has **set-user-ID** mode.

Octal	Text	Name
4000	chmod u+s	setuid bit
2000	chmod g+s	setgid bit
1000	chmod +t	sticky bit

The last character of the mode (usually x or -) is **t** if the 1000 (octal) bit of the mode is set. The mode **t** indicates that the sticky bit is on for the file or the directory.

object type and additional permissions			owner			group			others		
set user id	set group id	set stic-ky bit	r|-	w|-	x|{**s**|**S**}|-	r|-	w|-	x|{**s**|**S**}|-	r|-	w|-	x|{**t**|**T**}|-
4	2	1	4	2	1	4	2	1	4	2	1
shows type: **-** file **d** directory equals bit 0-7 [4+2+1]			shows r|w|{x|s|S}|- equals bit 0-7 [4+2+1]			shows r|w|{x|s|S}|- equals bit 0-7 [4+2+1]			shows r|w|{x|t|T}|- equals bit 0-7 [4+2+1]		

The indications of set-ID and 1000 bit of the mode are capitalized (**S** and **T**) if the corresponding *execute permission* is not set.

The command **chmod** can, when used in numeric mode (octal code) define the base permissions and attributes. The extended permissions are inactivated, if you use the numeric code of the command chmod a file that has an ACL. (For AIX:) The symbolic mode of the chmod command inactivates the advanced permissions not if, the corresponding ACL type AIXC. For more information on numeric and symbolic mode, see your Unix system man pages of chmod.

Grant permissions

Use the **chmod** command to grant access permission to files and directories.

```
chmod {u|g|o}{=|+|-}{r|w|x|s|t} file|directory
```

Show permissions

The command **ls –l** displays the mode (including security information), number of links, owner, group, size (in bytes), time of last modification, and name of each file.

Examples

chmod 6754 file chmod a=r file	-rwsr-sr-- ... file -r--r--r-- ... file
chmod 7777 directory chmod a-t directory chmod o-x directory chmod a+t directory	drwsrwxrwt ... directory drwsrwxrwx ... directory drwsrwxrw- ... directory drwsrwxrwT ... directory

File test conditions for Korn and Posix shell

-g file	true, if file exists and groupid s-bit is on (setgid)
-k file	true, if file exists and t-bit (sticky bit) is on
-u file	true, if file exists and userid s-bit is on (setuid)