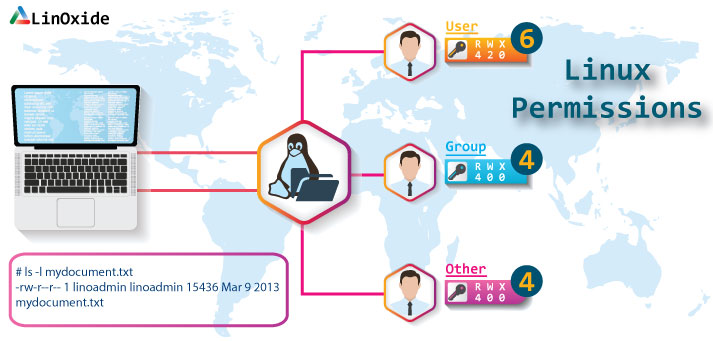
**Linux File Permissions for Beginners**

Updated December 14, 2019By [Bobbin Zachariah](https://linoxide.com/author/bobbin/)

In Linux operating system, everything is organized in the form of files and directories. By setting permissions on files and directories, one can make sure that only authorized users are allowed to access a specific data. Each file in Linux is owned by a user and group. The user is the one that creates the file and group is the one to which the user (owner of the file) belongs to.

**Understand file permissions**

File permissions consist of three permissions that you can apply to files and directories. In this section, you’ll learn how the system works and how to modify these permissions. Before doing this, let’s have a look at how to read the current permissions. The best method to do so is by using ls -l, which will show you a [list of all files and directories](https://linoxide.com/linux-command/20-ls-command-linux/) in the current directory

For example, you can list the files under the directory /home/sam as follows.

ls -l /home/sam

drwxrwxrwx 3 sam admin 80 2012-08-20 21:37 tmp

-rw-rw-r-- 1 sam admin 8187 2012-08-25 13:35 file1

-rwxr-x--- 1 sam admin 10348 2012-08-21 20:31 file2

The result is displayed on 7 columns but we will just concentrate on the first, the third and the fourth column.

* the first column shows the file permissions,
* the third column shows the user owner of the file,
* the fourth column shows the group owner of the file.

To understand the file permissions easily, we need to understand the first column which is on this form

[d][rwx][rwx][rwx]

Here the first character indicates the type of file. For instance, it gives

* **d** : directory
* **-** : regular file
* **l** : symbolic link
* **p** : named pipe
* **s** : Unix domain socket
* **c** : character device file
* **b** : block device file

Next are nine characters to specify the permissions that are set to the file or directory:

* the first set of three are the user owner permissions,
* the next set of three are the group owner permissions,
* the last set of three refers to the permissions granted to others.

The three basic permissions allow you to read, write, and execute files but it also exist special permissions. The effect of these permissions will be different when applied to files or directories. There are:

* **read permission (r**or**4)**: means you will be able to read a file and list the content of a directory
* **write permission (w** or **2**): means you will be able to edit a file and add, delete or rename files in a directory.
* **execute permission (x** or **1**): means you will be able to execute a program or shell script and move to a directory (cd to the directory).
* **Set User ID (SUID) permission (u+s or 4):**can only be set to a user*. It means that any user can execute program/script with permissions of file owner.*
* **Set Group ID (GUID) permission (g+s or 2):**can only be set to a group. *It**means that any user can execute a program/script with permissions of group owner and any file created in a directory gets the same group owner.*
* **Sticky bit permission (t or 1):**can only be applied on directory and prevent users from deleting files from other users.

For the mode detail on ***chmod*** concept command, you can read this article for newbies and advanced Linux users.

**1. Symbolic *chmod* permission examples**

The [linux command *chmod*](https://linoxide.com/linux-command/chmod-command/) can be used to change the permission of a file or directory. When you want to set permissions, you can use the symbolic mode *(r, w, x, s, t)*. To apply it to a directory with its content (recursive), you use -R option of ***chmod*** command.

**a. chmod +x**

To add an execute permission on a script or a program in order to run it, we can use chmod +x command which will set the permission to user, group and the other. The + operator add a permission to the existing ones. For example:

# ./hello

bash: ./hello: Permission denied

Now let's apply the permission

# chmod +x hello

# ./hello

Hello... How are you?

To access (move) on a directory, we use execute permission

$ cd test/

-bash: cd: test/: Permission denied

$ chmod -R +x test

$ cd test/

[papso@centos-01 test]$

**b. chmod u=rx**

You can just change the user owner’s permissions with = operator followed by the permission. This operator replaces the last permissions by the newest permissions. For example

$ ls -ld test/

drwxrwxr-x 2 papso papso 4096 May 15 20:18 test/

You can see that the user owner has the *w* permission. Now let's apply the permission below

$ chmod -R u=rx test

$ ls -ld test/

dr-xrwxr-x 2 papso papso 4096 May 15 20:18 test/

**c. chmod g+w,o-x**

It is possible to add permission to an entity and remove (- operator) permission to another entity on a single command. For example, we will add write permission to group entity and remove only execute permission to the others

# chmod g+w,o-x hello

# ls -l hello

-rwxrwxr-- 1 root root 66 May 15 20:12 hello

**2. Numeric or octal *chmod* permission examples**

You can also use numeric mode *(4, 2, 1)* when you want to set permissions. But with this mode, you use three digits and you need to calculate the value of each entity in order to set the good permissions.

**a. chmod 755**

If you want to set permission to a directory such that the user should be able to read, write and execute the directory, the group and the others should be only able to read and execute it, the permission should be like drwxr-xr-x. We can now find the octal value to use

For user part -> rwx = 4+2+1 = 7  
For Group -> r-x = 4+0+1 = 5  
For others -> r-x = 4+0+1 = 5

In some case, it's the default permission when you create a folder.

# chmod -R 755 folder1/

# ls -ld folder1/

drwxr-xr-x 2 root root 4096 May 16 01:54 folder1/

It is possible to combine chmod command with another command. for example, we will find all 777 permission directories and set permissions to 755.

# find /home -type d -perm 777 -print -exec chmod -R 755 {} \;

/home/papso/script-test/folder1

/home/papso/course

/home/patrick/toto

**b. chmod 754**

It is possible to make a program readable by the other to limit danger, give read and execute permission to group and all the permission to the user.

# chmod 754 hello

It is possible to set the permission to all script files on the current folder supposing you gave .sh extension to theses files

# chmod 754 \*.sh

**c. chmod 640**

If you want to set the permission of a file such that the user should be able to read and write the file, the group should be able to read the file and others should not have any access to the file, permission should be like -rw-r-----.

# chmod 640 bootstrap

We will find all 777 permission files and use chmod command to set permissions to 640.

# find /home -type f -perm 0777 -print -exec chmod 640 {} \;

/home/papso/script

/home/papso/script-test/recovery

/home/patrick/compta/finances

**3. Special bit chmod permission examples**

We can use the [SUID, GUID and sticky bits](https://linoxide.com/how-tos/stickbit-suid-guid/) to apply special permission on Linux file with chmod command. We will see some examples below

**a. chmod u+s and chmod 4655**

You can permit any user to run a program as if he was the user owner with the SUID permission. You can use the symbolic mode as below

# chmod u+s hello

# ls -l hello

-rwsr-xr-x 1 root root 66 May 15 20:16 hello

If you want to set the permission with numeric mode, you will four digits and you need to begin with the SUID value *(4)* followed by the set of three to set permission to the file as below

# chmod 4655 hello

**b. chmod +t and chmod 1777**

We will now prevent users from deleting file from the others user with the sticky bit. It is only set on folder, not file. Suppose that the folder has already all the permissions applied to any entity. You can use it as below

$ chmod +t test1

To use the numeric method, we need to begin with numeric sticky bit value *(1)* followed by the set of three to set

$ chmod 1777 -R test1

Let's check the result

$ ls -ld test1

drwxrwxrwt 2 papso papso 4096 May 16 18:54 test1

Now will log as patrick user and we will  try to delete a file

# su patrick

[patrick@centos-01 test1]$ ls -l proj

-rwxrwxrwx 1 papso papso 0 May 16 19:00 proj

$ rm file

rm: cannot remove ‘file’: Operation not permitted

You can see that in spite of all rwxrwxrwx permission, *patrick* user was not able to delete papso's file

**c. chmod g+s and chmod 2664**

You can set SGID permission to set default group ownership on files and sub-directories created in that directory. For example, if you have a shared group environment, this is not very useful, because no one else will be able to modify the files you’re creating, even if they’re member of the same group.

$ chmod g+s rh\_department

$ ls -ld rh\_department/

drwxrwsr-x 2 papso papso 4096 May 16 20:19 rh\_department/

The numeric mode of SGID permission is set as below

$ chmod 2755 rh\_departement

**Conclusion**

When using chmod, you can set permissions for user, group, and others. You can use this command in two modes: symbolic mode (relative mode) and numeric mode (absolute mode). In absolute mode, three or four digits (depending on special bit or not) are used to set the basic permissions and remember that you should calculate the value that you need.

**Sticky Bit, SUID and SGID in Linux with Examples**

Updated August 8, 2019By [Raghu](https://linoxide.com/author/raghu/)



In this article, we explain special permissions that work on files and directories named as Sticky bit, SUID and SGID.

The sticky bit works on directories only. If a user wants to create or [delete a file/directory](https://linoxide.com/security/delete-files-permanatly-linux/) in some directory, he needs write permission on that directory. The write permission on a directory gives a user the privilege to create a file as well as the privilege to remove it.

The [/tmp directory](https://linoxide.com/file-system/use-tmpfs-rhel-centos-7-0/) is the directory for temporary files/directories. This directory has all the rights on all the three levels because all the users need to create/delete their temporary files. But as the users have write permission on this directory, they can delete any file in this directory. The permissions of that file do not have any effect on deletion.

But with sticky bit set on a directory, anyone can create a file/directory in it, but can delete his own files only. Files owned by other users cannot be deleted.

## Sticky bit - How to view and set

You could notice t tag added to /tmp directory and it means bit is set for this directory.

$ ls -ld /tmp/

drwxrwxrwt 4 root root 4096 Aug 19 02:29 /tmp/

In Linux sticky bit can be set with ***chmod*** command. You can use +t tag to add and -t tag to delete sticky bit.

$ chmod o-t dir1

$ ls -l

total 8

drwxr-xr-x 2 root root 4096 Aug 19 03:08 dir1

$ chmod o+t dir1

$ ls -l

total 8

drwxr-xr-t 2 root root 4096 Aug 19 03:08 dir1

Alternatively,

$ chmod 1777 dir1/

$ ls -l

total 8

drwxrwxrwt 2 root root 4096 Aug 19 03:08 dir1

Note: In Unix flavored OS, sticky bit has a different purpose but we are not discussing it here.

## What is SUID Bit and How to set it

When an executable file runs, it runs *under the ownership of the user who has executed it*. It means that when student user runs [ls command](https://linoxide.com/linux-command/linux-ls-command/), then the corresponding process will run under the ownership of student. The SUID bit, also known as Set User ID bit, overwrites this behavior. If SUID bit is set on a program, then that program will run as the owner of that file, irrespective of who is executing it.

The passwd command in Linux has SUID bit set.

$ ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 23420 Aug 3 2010 /usr/bin/passwd

This can be seen in the third field of permissions. The 's' in place of 'x' indicates that SUID bit is set. With SUID bit set, when a normal user (say student) runs the passwd command, the command runs with the ownership of 'root', and not as student, because root is the owner of this file. This behavior is required because the passwords are stored in the /etc/shadow file, which has no permission on group or other level.

$ ls -l /etc/shadow

-r-------- 1 root root 1027 Jul 13 21:56 /etc/shadow

You need to understand that all users cannot be given read or write permission on this file for security reasons; otherwise, they will read/change the passwords of other users. So this causes a problem that if the users don't have permission on this file, then how will they change their own passwords? So SUID bit solves the problem. The [passwd command](https://linoxide.com/usr-mgmt/understand-each-entries-of-linux-passwd-etcpasswd-file/" \t "_blank) has SUID bit set, so when normal users execute this command, they run it with the ownership of root, i.e. the owner of passwd command.

## How to Set and unset SUID bit

This is to be noted that SUID bit works on files only. To set the SUID bit on a file, use the [chmod command](https://linoxide.com/linux-command/chmod-command/" \t "_blank) as follows

$ ls -l

total 8

-rwxr--r-- 1 root root 104 Aug 19 01:26 hello.sh

$ chmod u+s hello.sh

$ ls -l

total 8

-rwsr--r-- 1 root root 104 Aug 19 01:26 hello.sh

The numeric method for changing permissions can also be used. Suppose if the normal permissions for a file are 744, then with SUID bit set, these will become 4744. SUID bit has value 4.

$ ls -l

total 8

-rwxr--r-- 1 root root 104 Aug 19 01:26 hello.sh

$ chmod 4744 hello.sh

$ ls -l

total 8

-rwsr--r-- 1 root root 104 Aug 19 01:26 hello.sh

## How SGID Bit work on file and directory

Unlike SUID bit, SGID bit works on both files and directories, but it has a different meaning in both cases.

**On files:**

For file, it has similar meaning as the SUID bit, i.e. when any user executes a file with SGID bit set on it, it will always be executed with the group ownership of that file, irrespective of who is running it. For example, the file /sbin/netreport has SGID bit set, which can be seen in the 's' instead of 'x' in group permissions.

$ ls -l /sbin/netreport

-rwxr-sr-x 1 root root 6020 Oct 13 2010 /sbin/netreport

This file has group ownership of root group. So when a user (say student) executes it, then the corresponding process will not have group ownership of student, but that of root group.

**On directories:**

Now let’s talk about SGID on directories. SGID on directories is used for creating collaborative directories. To understand SGID bit on directories, consider the following scenario:

Suppose three users jack, jones and jenny are working together on some project. All of them belong to a group named javaproject. For the course of the project, they need to share all the files related to the project. All of them must be able to see each other's file. This can be done simply by providing read permission on group level. Further, suppose that the directory used for the project is "/javaproject".

Here, a problem arises that when a file is created, it belongs to the primary group of the user who created the file. So, when different users create their files in this directory, those files will not have group ownership of javaproject group.

What we do for our problem is that we set the group of /javaproject directory to be javaproject group, and set the SGID bit set on it. When SGID bit is set on a directory, all the files and directory created within it has the group ownership of the group associated with that directory. It means that after setting SGID bit on /javaproject directory, all the files and directories being created in this directory will have the group ownership of "javaproject" group. Moreover, this behavior is recursive, i.e. the directories created in this directory will have SGID bit set as well. The permissions for the new directory will also be same as that of /javaproject directory.

The SGID bit can be set with chmod command as follows:

$ ls -ld /javaproject

drwxrwxr-x 2 root javaproject 4096 Aug 19 02:33 /javaproject

$ chmod g+s /javaproject

$ ls -ld /javaproject

drwxrwsr-x 2 root javaproject 4096 Aug 19 02:33 /javaproject

Now when jones user creates a file in this directory, it is created under the group ownership of javaproject group.

$ touch /javaproject/jones1.txt

$ mkdir /javaproject/jones1dir

$ ls -l /javaproject/

total 12

drwxrwsr-x 2 jones javaproject 4096 Aug 19 02:38 jones1dir

-rw-rw-r-- 1 jones javaproject 0 Aug 19 02:37 jones1.txt

The numeric value corresponding to SGID bit is 2. So to add SGID bit numerically, use the following command:

$ ls -ld /shared/

drwxrwxr-x 2 root adm 4096 Aug 19 02:47 /shared/

$ chmod 2775 /shared/

$ ls -ld /shared/

drwxrwsr-x 2 root adm 4096 Aug 19 02:47 /shared/

Thanks for reading this article and refer [sticky bit wiki page](https://en.wikipedia.org/wiki/Sticky_bit) as well.