**[Interview Questions on Linux Permissions with Answers](http://www.golinuxhub.com/2014/01/interview-questions-on-linux-permissions.html)**

POSTED BY DEEPAK PRASAD SUNDAY, JANUARY 26, 2014 [5 COMMENTS](http://www.golinuxhub.com/2014/01/interview-questions-on-linux-permissions.html#comment-form)

**1.What is the difference between SUID and SGID?**  
Answe-

[Understanding special permission SUID](http://www.golinuxhub.com/2013/03/understanding-special-permission-suid.html)

### [Understanding Special Permission SUID in Linux](http://www.golinuxhub.com/2013/03/understanding-special-permission-suid.html)

POSTED BY DEEPAK PRASAD THURSDAY, MARCH 07, 2013 [1 COMMENT](http://www.golinuxhub.com/2013/03/understanding-special-permission-suid.html#comment-form)

There are various blogs and websites available explaining about SUID and SGID now I won't say I will  
tell you something extra instead just adding one more to the list see if it can make any [difference](javascript:void(0);).

Now above what I said you can skip that and returning to the point I will try to help you understand SUID, [SGID](http://www.golinuxhub.com/2013/03/understanding-special-permission-sgid.html) and [Sticky Bit](http://www.golinuxhub.com/2013/03/understanding-special-permission-sticky.html) along with some useful tips from my side in understanding the same.

### SUID:

This is a abbreviation used for Set User ID which means that you are assigning a special permission on a user owner of any particular file or [directory](javascript:void(0);).

**Now what does it means and why do we use SUID?**

Assigning a user SUID means that you are giving him additional permission i.e user owner permission over the normal permission which he already has to run any executable file inside some directory on with the suid is applied.

**Let me be more clear about the above statement with an example:**

By default there are a number of executable binary files in Linux which we use as commands which has to be run only as a [super](javascript:void(0);) user i.e. [root](javascript:void(0);) user but what if a normal user is allowed to run those commands. Now in that [case](javascript:void(0);) even if you have given him sudo access he won't be able to run those commands as these are predefined to be run only as root and no one else.

**For example :**

mount, umount,ping, passwd, chage etc.

Now the above mentioned commands can only be run as a super user. So when you check the permission of these commands

# ls -l /bin/ping  
-rwsr-xr-x. 1 root root 36892 Jul 19 2011 /bin/ping  
  
# ls -l /bin/mount  
-rwsr-xr-x. 1 root root 73996 Dec 8 2011 /bin/mount  
  
# ls -l /bin/ping  
-rwsr-xr-x. 1 root root 36892 Jul 19 2011 /bin/ping  
Now as you see in the user permission section additional [small](javascript:void(0);) (s) is there at the place of execute permission

This means that all these commands have SUID set on it and if a normal user is given sudo permission to run these commands, they will run it as user owner's permission which in all the above case is root.

**Why do we use SUID?**

I hope I have explained this part above but still let me add a description with another example. Now there is some executable file whose owner is Deepak and it can only be run by deepak but still you want Amit to run the file so in that case instead of changing the owner of that file I will assign a SUID on it so that Amit can also run that file using Deepak's permission. So this is going to remove the complexity part and not only Amit any other random user who is allowed to run that file will run the same using ownership of deepak as I do not need to change the owner every time for some one to run the same executable file.

### ****Assigning SUID permission****

There are two ways to assign suid

1. Octal (4)
2. Symbolic (u+s)

**Octal**[**method**](javascript:void(0);)**:**

# chmod 4755 /myscrip.sh  
# ls -l  
-rwsrw-rw-. 1 root root 0 Oct 16 11:33 /myscript.sh

**Symbolic method:**

# chmod u+s /myscript.sh  
  
# ls -l  
-rwsrw-rw-. 1 root root 0 Oct 16 11:34 /myscript.sh

### ****Removing SUID permission****

**Octal method:**

# chmod 0755 /myscript.sh  
  
# ls -l  
-rwxrw-rw-. 1 root root 0 Oct 16 11:35 /myscript.sh

**Symbolic method:**

# chmod u-s /myscript.sh  
  
# ls -l  
-rwxrw-rw-. 1 root root 0 Oct 16 11:36 /myscript.sh

### Understanding the difference between Capital (S) and small (s) in SUID

There might be time when you see a capital S and a small s after applying SUID on any file or directory

Now it does not makes much difference on its working but instead it tells you something about the permission currently applied on the file.

If the file where you are assigning SUID has user executable permission then after applying SUID you will get a small (s) but if the user doe not have execute permission before you apply SUID then you will end up with capital (S) after applying SUID.

Let me show you the same with an example

**Before applying SUID without executable permission on user owner**

# chmod 655 /myscript.sh  
  
# ls -l-rw**-**rw-rw-. 1 root root 0 Oct 16 11:35 /myscript.sh  
**After applying SUID without executable permission on user owner**  
# chmod 4655 /myscript.sh  
  
# ls -l  
-rw**S**rw-rw-. 1 root root 0 Oct 16 11:35 /myscript.sh

**Before applying SUID with executable permission on user owner**

# chmod 755 /myscript.sh  
  
# ls -l  
-rw**x**rw-rw-. 1 root root 0 Oct 16 11:36 /myscript.sh  
**After applying SUID with executable permission on user owner**  
# chmod 4755 /myscript.sh  
  
# ls -l  
-rw**s**rw-rw-. 1 root root 0 Oct 16 11:36 /myscript.sh  
So I hope you have got my point of view and must have understood the difference between **capital (S)** and **small (s)**

**Finding all the executable files with SUID**  
# find / -perm +4000  
where +4000 is the ID we use for assigning permission in octal method.

[Understanding special permission SGID](http://www.golinuxhub.com/2013/03/understanding-special-permission-sgid.html)

### Understanding difference between Capital (S) and small (s) in SGID

Now when you assign SGID permission you might sometimes see a Capital (S) instead of a [small](javascript:void(0);) (s) in the group permission section. This does not makes much difference instead if gives you an additional information if that file is having group executable permission or not. If you get Capital S it means there is not executable permission and the same if you have small s it means the file is having group executable permission.  
  
**For example:**  
Before applying SGID without executable permission on user owner  
# chmod 655 /myscript.sh  
# ls -l  
 -rwxrw**-**rw-. 1 root root 0 Oct 16 11:35 /myscript.sh  
After applying SGID without executable permission on user owner  
# chmod 2655 /myscript.sh  
# ls -l  
-rwxrw**S**rw-. 1 root root 0 Oct 16 11:35 /myscript.sh  
Before applying SGID with executable permission on user owner  
# chmod 755 /myscript.sh  
# ls -l  
-rwxrw**x**rw-. 1 root root 0 Oct 16 11:36 /myscript.sh  
After applying SGID with executable permission on user owner  
# chmod 2755 /myscript.sh  
# ls -l  
-rwxrw**s**rw-. 1 root root 0 Oct 16 11:36 /myscript.sh  
So I hope you have got my point of view and must have understood the difference between capital (S) and small (s)  
  
Finding all the executable files with SGID  
# find / -perm +2000

**2. What is Sticky**[**Bit**](javascript:void(0);)**permission?**  
Answer-

[Understanding special permission Sticky Bit](http://www.golinuxhub.com/2013/03/understanding-special-permission-sticky.html)

### [Understanding Special Permission Sticky Bit in Linux](http://www.golinuxhub.com/2013/03/understanding-special-permission-sticky.html)

POSTED BY DEEPAK PRASAD THURSDAY, MARCH 07, 2013 [1 COMMENT](http://www.golinuxhub.com/2013/03/understanding-special-permission-sticky.html#comment-form)

Now there are few permissions which are used in Linux and are referred as special permission due to [SUID](http://www.golinuxhub.com/2013/03/understanding-special-permission-suid.html) and [SGID](http://www.golinuxhub.com/2013/03/understanding-special-permission-sgid.html) in their respective link their different functionality in terms of their usage. You can find details about

**Sticky Bit**  
This special permission becomes very useful in most the cases. This is used when you are the owner of a particular file and you have give full permission to that file for all others but still you don't want any one of them to delete that file apart from the user and [group](javascript:void(0);) owner. In that case sticky bit plays a very important role as once you assign this permission to some file or [directory](javascript:void(0);) no one else apart from the user and group owner will be able to delete that file or directory.  
  
Before showing you any example let me give you some helpful and important tips.  
Sticky Bit can be [assigned](javascript:void(0);) using two ways  
1. Octal (1)  
2. Symbolic (t)  
  
Octal (1):  
If you want to use octal method then this is the syntax which you need to follow  
# chmod **1**XXX /dirname  
Here 1 means assigning sticky bit and XXX means the permission to be applied  
  
**For example:**  
# chmod **1**775 /statusupdate  
Here I am assigning full permission to user and group owner and read and execute permission to others including a sticky bit given by 1 at the beginning of permission.  
  
**Symbolic (t) :**  
If you want to assign sticky bit using symbolic way then this will be the syntax  
# chmod +t /dirname  
**For example**  
# chmod o+t /statusupdate  
Here I am not meshing with any other existing permission instead additionally I am assigning a sticky bit permissions for all others for statusupdate directory  
  
Let me show you some practical example.  
  
**Scenario:**  
I have 2 [users](javascript:void(0);) namely ***user1*** and ***user2***. A common directory is assigned to both of them by the root to put up their status update at the end of the day in this directory. Now being a root I will assign sticky bit to the main directory along with any sub directories if there is any.  
# mkdir /statusupdate  
# chmod 1777 /statusupdate  
**user1 statusupdate**  
$ cd /statusupdate  
$ mkdir mywork  
$ chmod 1777 mywork  
  
$ ls -l  
total 4  
drwxrwxrw**t**. 2 user1 user1 4096 Oct 17 07:04 mywork  
Now as in my case for the demo purpose I have given full permission to mywork directory which I don't think most will do but this is just an example. Now as you see addition **(t)** option is visible marked in red color in the permission section for others.  
  
**Now log in as user2**  
It seems ***user2***is not so friendly with ***user1***and wants to delete his statusdata to create his [impression](javascript:void(0);) on the boss. Lets see if he can do that  
$ cd /statusupdate  
$ ls -l  
total 4  
drwxrwxrwt. 2 deepak deepak 4096 Oct 17 07:04 mywork  
$ rm -rf mywork  
rm: cannot remove `mywork': Operation not permitted  
Ooops the operation is not permitted. So it seems ***user2***will have to honestly work hard to create an impression over his boss.  
  
So this is how sticky bit works the same could have been done using symbolic way as well.

### Removing sticky bit

# chmod 0775 /statusupdate  
# ls -l  
drwxrwxr-x.   3 root root  4096 Oct 17 07:07 statusupdate  
The same can be done in symbolic way using the below command  
# chmod -t /statusupdate  
**IMPORTANT NOTE**  
Many a times you will observe a **capital (T)** at the others permission section instead of **small (t)**now you do not have to get confused regarding this as both of them signify sticky bit but with a little [difference](javascript:void(0);) that if others have executable permission on them then after applying sticky bit you will get small (t) but if others do not have executable permission then others will get capital (T).  
  
Let me show you with the help of one example  
  
Before applying Sticky Bit with executable permission  
# chmod 775 /statusupdate  
# ls -l  
drwxrwxr-**x**.   3 root root  4096 Oct 17 07:07 statusupdate  
After Sticky Bit with executable permission  
# chmod 1775 /statusupdate  
# ls -l  
drwxrwxr-**t**.   3 root root  4096 Oct 17 07:07 statusupdate  
Now as you see a small (t) since the directory had executable permission  
  
Before applying sticky bit without executable permission  
# chmod 774 /statusupdate  
# ls -l  
drwxrwxr-**-**.   3 root root  4096 Oct 17 07:07 statusupdate  
After Sticky Bit without executable permission  
# chmod 1774 /statusupdate  
# ls -l  
drwxrwxr-**T**.   3 root root  4096 Oct 17 07:07 statusupdate  
So I hope I cleared my point on all the possible cases with sticky bit.  
  
Now in case you want to search all the files and directories with sticky bit permission  
# find / -perm +1000  
where 1000 signifies files or dir having sticky bit as per the octal value we use. Now again you can use additional switch with find command like -type d or f to search more accurately.

**3. What is umask?**  
Show/Hide Answer

In computing, umask is a command that determines the settings of a mask that controls which file permissions are set for files and directories when they are created. It also refers to a function that sets the mask, and to the mask itself, which is formally known as the file mode creation mask.

**4. What is the default umask value for useradd command and where is it defined?**  
Show/Hide Answer

Default umask value for useradd: 077  
/etc/login.defs

**5. Will you be able to cd into a directory with only read permission?**  
Show/Hide Answer

No, we need execute permission along with read to cd into directory

**6. What is -R argument used for along with chmod command?**  
Show/Hide Answer

To recursively apply the permission to all the directories including sub directories and files

**7. How can you restrict a normal as well as**[**root**](javascript:void(0);)**user from making any changes as well as deleting any file?**  
Show/Hide Answer

chattr command  
**For more information follow the below link**  
[Increase security using extended file attributes](http://www.golinuxhub.com/2012/10/increase-security-with-extended-file.html)

**8. What is the + plus sign you see at the end of permissions for some directories?**

Answer-

[What is (+) plus sign in permission section for any directory/file in Linux?](http://www.golinuxhub.com/2013/12/what-is-plus-sign-in-permission-in-linux.html)

### [What is the plus (+) sign in permission in Linux ?](http://www.golinuxhub.com/2013/12/what-is-plus-sign-in-permission-in-linux.html)

POSTED BY DEEPAK PRASAD WEDNESDAY, DECEMBER 25, 2013 [NO COMMENTS](http://www.golinuxhub.com/2013/12/what-is-plus-sign-in-permission-in-linux.html#comment-form)

So do you see a **plus** sign in the permission section in any of your [directory](javascript:void(0);).  
  
No need to get confused, well it just means that the directory has extra acl permission. We use acl to give individual permission for users or groups on any directory.  
  
**For eg.**  
I have a directory deepak inside my [root](javascript:void(0);) home directory  
# ls -l | grep deepak  
drwxrwxr-x 2 deepak deepak  4096 Sep 24 16:14 deepak  
Now I want to give a different user [test](javascript:void(0);) additional rwx full permission on this directory so what am I suppose [to do](javascript:void(0);)?  
 One option is that I will have to add test user to [group](javascript:void(0);) owner of deepak directory but I really don't want to do that then in those case setfacl plays its role  
  
# setfacl -m u:test:rwx /root/deepak  
# ls -l | grep deepak  
drwxrwxr-x+ 2 root root  4096 Sep 24 16:14 deepak

Now as you can see an extra plus (**+**) sign has appeared. But still it does not [shows](javascript:void(0);) me the acl implemented on that directory.

# getfacl /root/deepak  
# file: root/deepak/  
# owner: root  
# group: root  
user::rwx  
user:test:rwx  
group::r-x  
mask::rwx  
other::r-x  
So as you can see all the acl details using the above command. For more detailed help and examples on setfacl follow the below link  
**9. How do you give acl in Linux?**  
Answer

[Give individual permission using setfacl](http://www.golinuxhub.com/2012/10/give-individual-permission-on.html)

### [Give individual permission on directories using setfacl in linux](http://www.golinuxhub.com/2012/10/give-individual-permission-on.html)

POSTED BY DEEPAK PRASAD FRIDAY, OCTOBER 05, 2012 [NO COMMENTS](http://www.golinuxhub.com/2012/10/give-individual-permission-on.html#comment-form)

Generally we use chmod and chown to give [user](javascript:void(0);) wise or group wise permission and ownership on [directories](javascript:void(0);) and files in linux but again if you want to five some special permission to a particular user or group on any particular [directory](javascript:void(0);) then both the above commands won't help you. In such conditions 'setfacl' utility plays its vital role and is very useful. Using 'setfacl' you can give individual permission on the basis of user or group name on any particular directory or files.

**Syntax:**

To set the permission for any user  
# setfacl -m u:username:permission /path/to/directory  
To set the permission for any group  
# setfacl -m g:groupname:permission /path/to/directory

To view the permission  
# getfacl /path/to/directory  
To [remove](javascript:void(0);) individual acl for any user  
# setfacl -x username /path/to/directory  
To remove all the acl added by setfacl  
# setfacl -b /path/to/directory  
To remove the default acls on any directory  
# setfacl -d /path/to/directory  
**Examples:**  
To add an acl for user deepak with read and execute permission on mydata directory  
# setfacl -m u:deepak:r-x /mydata  
To add an acl for group admin on any directories  
# setfacl -m g:admin:rwx /mydata  
To add the acl recusively on all the sub directories  
# setfacl -Rm -u:deepak:r-x /mydata/  
To view the acl entries on mydata  
# getfacl /mydata  
# file: new  
# owner: [root](javascript:void(0);)  
# group: root  
user:deepak:r-x  
group:admin:rwx  
group::r-x  
mask::r-x  
other::r-x

# ls -l / | grep mydata  
drwxr-xr-x+ 2 root root 4096 Oct 3 16:49 mydata  
So here you can see '+' sign is added at the last of permission section of the directory which means that acl is active on that directory.  
  
To remove a particular acl from the directory  
# setfacl -x u:deepak /mydata  
To remove all the acls from any directories  
# setfacl -b /mydata  
For further examples you can go to man page for setfacl

**10. What is the difference between small t and capital T when applying sticky bit permission?**

Show/Hide Answer

Before applying Sticky Bit with executable permission  
# chmod 775 /statusupdate  
# ls -l  
drwxrwxr-x. 3 root root 4096 Oct 17 07:07 statusupdate  
After Sticky Bit with executable permission  
# chmod 1775 /statusupdate  
# ls -l drwxrwxr-**t**. 3 root root 4096 Oct 17 07:07 statusupdate  
Now as you see a small (t) since the directory had executable permission  
  
Before applying sticky bit without executable permission  
# chmod 774 /statusupdate  
# ls -l  
drwxrwxr--. 3 root root 4096 Oct 17 07:07 statusupdate  
After Sticky Bit without executable permission  
# chmod 1774 /statusupdate  
# ls -l  
drwxrwxr-**T**. 3 root root 4096 Oct 17 07:07 statusupdate