

5+ practical examples to learn sticky bit in Linux

 goinuxcloud.com/sticky-bit-linux

March 25, 2020

In this article we will learn about Unix or [Linux sticky bit with examples](#) and everything related to this special permission

Below are the most common tools **we use to enforce permission and restrictions to files and directories** in Linux and Unix

- chmod
- chown
- setfacl
- chattr

But apart from this, there are three other types of **special permission** which can be used in Linux or Unix

- setuid
- setgui
- sticky bit

Each of these special permission have their own benefit and you can decide to choose them based on your requirement. I will share various examples to understand each of the special permission separately.

Let us learn about sticky bit special permission in Linux and Unix.

Why we don't use sticky bit for files any more?

From the [man page of chmod](#),

For regular files on **older Linux and Unix systems**, the bit saves the program's text image on the swap device so it used to load more quickly when executed; this is called the sticky bit. The name sticky came about because the text portion of the file stuck around in the swap area until the system was rebooted.

- On older versions of the Linux and UNIX System that predated demand paging, this bit was known as the **sticky bit**.
- If it was set for an executable program file, then the first time the program was executed, **a copy of the program's text was saved in the swap area when the process terminated**.
- The program would then load into memory more quickly the next time it was executed, because the swap area was handled as a contiguous file, as compared to the possibly random location of data blocks in a normal UNIX file system.
- The Unix sticky bit was often set for common application programs, such as the text editor and the passes of the C compiler.

- Naturally, there was a limit to the number of sticky files that could be contained in the swap area before running out of swap space, but it was a useful technique.
- The name sticky came about because the text portion of the file stuck around in the swap area until the system was rebooted.
- With today's newer Linux and UNIX systems, most of which have a virtual memory system and a faster file system, **the need for this technique has disappeared.**
- Hence today **we use sticky bit mostly on directories and not on files.**

What is sticky bit special permission?

The Linux sticky bit for directories, **it prevents unprivileged users from removing or renaming a file in the directory** unless they own the file or the directory. This is called the restricted deletion flag for the directory.

The directories `/tmp` and `/var/tmp` are typical candidates for the linux sticky bit - they are directories in which any user can typically create files. The permissions for these two directories are often read, write, and execute for everyone (user, group, and other). But users should not be able to delete or rename files owned by others.

- By adding the linux sticky bit permission, the write permission for directories changes meaning.
- Instead of meaning “add and delete all files in the directory,” **the only users who can now delete files in this directory are**
 - *The owner of the file or*
 - *The owner of the directory (which is normally the root user) or*
 - *The root user*
- Anyone can still add a file into this directory, as long as **it doesn't result in overwriting another user's file.**

As you can see `/tmp` and `/var/tmp` , by default have sticky bit permission

```
# ls -ld /var/tmp/ /tmp/
drwxrwxrwt. 9 root root 4096 Mar 23 18:45 /tmp/
drwxrwxrwt. 3 root root 4096 Mar 22 14:50 /var/tmp/
```

NOTE:

Solaris 10 places special meaning on the Unix sticky bit if it is set on a regular file. In this case, if none of the execute bits is set, the operating system will not cache the contents of the file.

How to apply sticky bit in Linux or Unix

There are two methods to apply Linux or Unix sticky bit special permission using `chmod`

1. Octal method (1)
2. Symbolic method (t)

For the sake of this article I will assume that you are familiar with the different permission fields of user, group and others.

Set sticky bit using Octal method (1)

Below are some examples to apply linux sticky bit using the Octal method with chmod in Linux and Unix. I have created a directory /tmp/marketing on which I will apply linux sticky bit special permission

```
# mkdir /tmp/marketing
```

To apply sticky bit with 755 permission

```
# chmod 1755 marketing/
```

As you see now we have "t" shown in the "execute field of others"

```
# ls -ld marketing
drwxr-xr-t 2 root root 4096 Mar 23 17:47 marketing
```

Next apply sticky bit with 750 permission

```
# chmod 1750 marketing
```

If you verify the permission, you will see uppercase "T" unlike lowercase "t" we saw above

```
# ls -ld marketing
drwxr-x--T 2 root root 4096 Mar 23 17:47 marketing
```

Unset sticky bit using Octal method (1)

So now how do we remove the " t " permission using octal method? Similar to above commands, we can also remove or unset linux sticky bit special permission with **chmod** . To remove sticky bit permission using octal method, you just need to avoid using " 1 " while applying the permission to the directory or file For example, here linux sticky bit is set on " **marketing** " directory:

```
# ls -ld marketing
drwxr-x--T 2 root root 4096 Mar 23 17:47 marketing
```

So to unset sticky bit we just apply the new permission **without specifying octal value of sticky bit**:

```
# chmod 755 marketing
```

Verify the permission, as you see the linux or unix sticky bit permission is removed

```
# ls -ld marketing
drwxr-xr-x 2 root root 4096 Mar 23 17:47 marketing
```

Alternatively, you can use "o" octal value to **remove sticky bit special permission** as shown below:

```
# chmod 0755 marketing
```

Next you can verify the permission of your directory:

```
# ls -ld marketing
drwxr-xr-x 2 root root 4096 Mar 23 17:47 marketing
```

As you see, we have successfully unset linux or unix sticky bit special permission

HINT:

You must use octal method to apply sticky bit only if you also plan to change or modify permission of the directory or I would suggest you to use symbolic method to set sticky bit in Linux or Unix.

I am sure you will be wondering at this stage, *what is the difference between small 't' and capital 'T'*. I will address this question at later part of this article.

Set sticky bit using Symbolic method (t)

Below are some examples to set linux sticky bit using the Symbolic method with `chmod` in Linux and Unix. I have created a directory `/tmp/marketing` on which I will apply unix sticky bit special permission

```
# mkdir /tmp/marketing

# ls -ld marketing
drwxr-xr-x 2 root root 4096 Mar 23 17:47 marketing
```

To **set sticky bit special permission using symbolic method**, use the following command:

```
# chmod +t marketing
```

Next verify the permission:

```
# ls -ld marketing
drwxr-xr-t 2 root root 4096 Mar 23 17:47 marketing
```

As expected we have **small "t"** in the execute section of the permission

Alternatively you can also use "`o+t`" with `chmod` to **set sticky bit permission**. Here "**o**" means "**others**" field

```
# chmod o+t marketing
```

HINT:

You cannot use "`u+t`" or "`g+t`" i.e. you cannot apply sticky bit for users and groups section, sticky bit can only be applied to "others" section of permission

Unset sticky bit using symbolic method (t)

So now how do we remove the "`t`" permission using symbolic method? Similar to above commands, we can also remove or unset Unix sticky bit permission using symbolic method (`t`). **To remove sticky bit you just need to use (`-t`)** instead of (`+t`) as we used earlier:

For example, for the same directory you can execute below command **to unset sticky bit special permission**:

```
# chmod -t marketing/
```

Verify the permission:

```
# ls -ld marketing
drwxr-xr-x 2 root root 4096 Mar 23 17:47 marketing
```

Alternatively you can also use (**o-t**) i.e. **remove sticky bit permission** for "others"

```
# chmod o-t marketing/
```

NOTE:

The **advantage of using symbolic method** is that you do not modify the existing permission of the directory, and you only apply or remove the sticky bit permission

What is the difference between uppercase 'T' and lowercase 't' in Unix and Linux Sticky Bit permissions?

When Unix or Linux sticky bit is combined with execute permission for *others* field you will see lowercase "t" i.e. **drwxr-xr-t** and if there is no execute permission set for *others* then you will see uppercase "T" i.e. **drwxr-x--T**

I hope you are aware of the different section of permission field.

For example for uppercase "T" i.e. "**drwxr-x--T**"

First Field	Second Field	Third Field	Fourth Field
d/-	rwX	r-x	--T
d is for directory - is for file	r → read w → write x → execute	r → read - → No write permission x → execute	- → No read permission - → No write permission T → No execute permission + Sticky Bit
The first field is to identify a file or directory	Permissions for User Owner	Permissions for Group Owner	Permissions for Others

Similarly for lowercase "t" i.e. "**drwxr-x--t**"

First Field	Second Field	Third Field	Fourth Field
d/-	rwX	r-x	--t

First Field	Second Field	Third Field	Fourth Field
d is for directory - is for file	r → read w → write x → execute	r → read - → No write permission x → execute	- → No read permission - → No write permission t → execute permission + Sticky Bit
The first field is to identify a file or directory	Permissions for User Owner	Permissions for Group Owner	Permissions for Others

So to summarise:

- If the "others" section contains "**execute permission + sticky bit**" then you will get **lowercase "t"**
- If the "others" section **does not contains execute permission and only sticky bit** then you will get **uppercase "T"**

How does the sticky bit work? Why we use sticky bit in Linux or Unix?

Now up til now we learned what is sticky bit in theory, let us see some practical examples of Linux/Unix sticky bit in action.

I will apply sticky bit on my " `/tmp/marketing` " directory

```
# mkdir /tmp/marketing

# chmod +t /tmp/marketing/

# ls -ld /tmp/marketing/
drwxrwxrwt 2 root root 4096 Mar 23 17:45 /tmp/marketing/
```

I ave two users on this system, **user1** and **user2**. Now both users are supposed to place their files under " `/tmp/marketing` "

```
[user1@centos-8 ~]$ touch /tmp/marketing/user1_file
[user2@centos-8 ~]$ touch /tmp/marketing/user2_file
```

So both the users have created a file under `/tmp/marketing` directory.

```
[user1@centos-8 marketing]$ ls -l
total 0
-rw-rw-r-- 1 user1 user1 0 Mar 23 17:47 user1_file
-rw-rw-r-- 1 user2 user2 0 Mar 23 17:44 user2_file
```

But what if `user1` plans to delete `user2's` file?

```
[user1@centos-8 marketing]$ rm -f user2_file
rm: cannot remove 'user2_file': Operation not permitted
```

As you see, since we had sticky bit permission on the parent directory, **a non privilege user is allowed to remove or modify files which are owned by self only.**

How to find files and directories with sticky bit permission?

We know that `/tmp` and `/var/tmp` directories contains sticky bit special permission by default,. Now to find other files and directories with sticky bit permission you can us

" `find` " command with `-perm -1000`

When the `-perm` argument has a minus sign, all of the permission bits are examined, including the set user ID, set group ID, and sticky bits.

For example: to find all the **directories** under "/" with Linux sticky bit permission, use below command

```
# find / -perm -1000 -type d
/sys/fs/bpf
/dev/mqueue
/dev/shm
/tmp
/tmp/.font-unix
/tmp/.X11-unix
```

Lastly I hope the steps from the article to understand what is sticky bit, how to apply and remove Linux or Unix sticky bit permission, understanding the difference between small 't' and capital 'T' on Linux and Unix was helpful. So, let me know your suggestions and feedback using the comment section.

Related Searches:

chmod sticky bit. understanding linux special permission. what is linux sticky bit and explain with example. unix file permissions Unix sticky bit. What is the "t" letter in the output of "ls -ld /tmp"? What is difference between small "t" and capital "T" in Linux permissions?