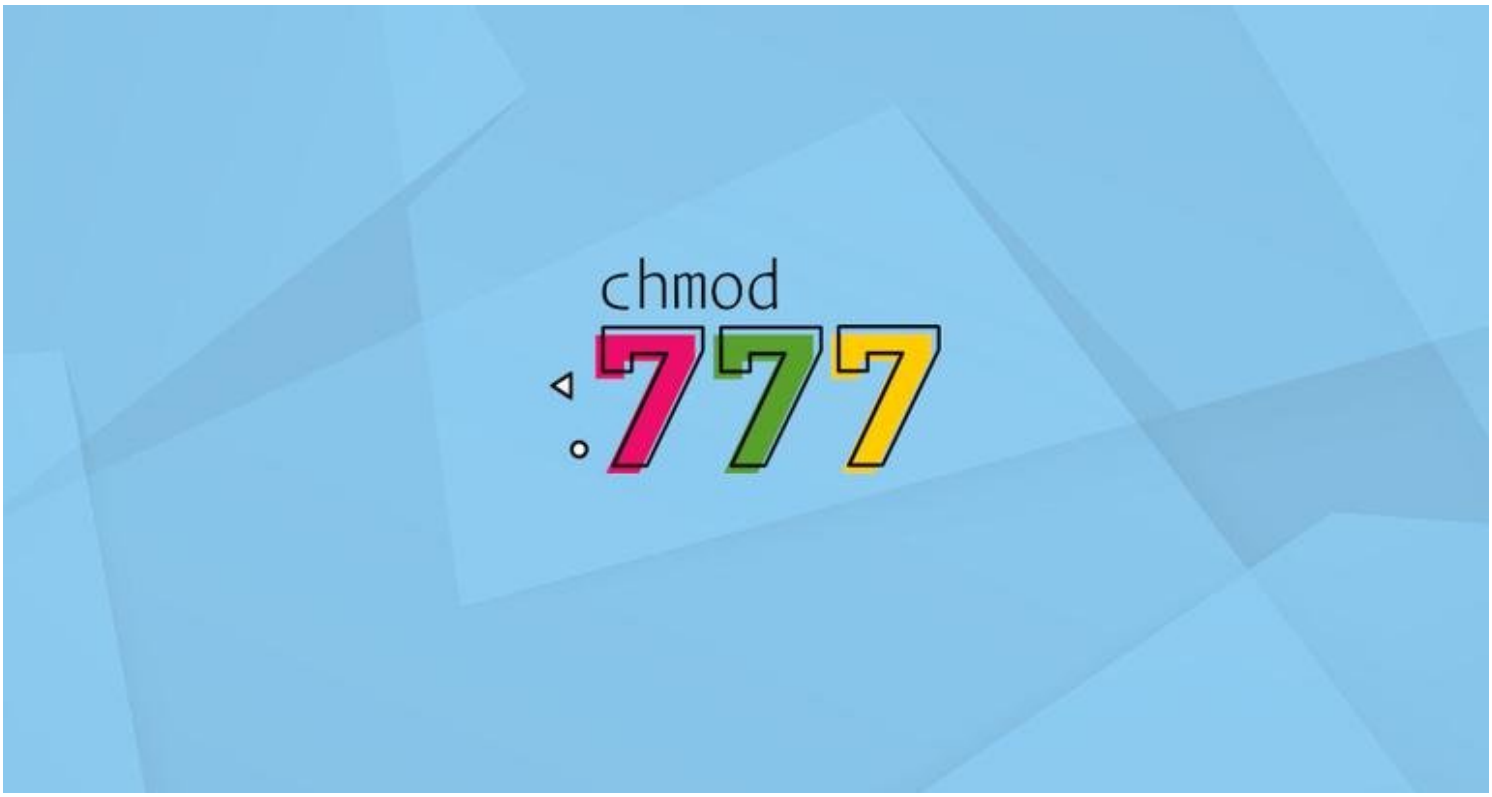
What Does chmod 777 Mean

Posted Mar 8, 2020 • 5 min read



You are trying to fix a permission issue with your web server and found information on the Internet, saying that you need to recursively `chmod 777` the web directory. Before doing that, make sure you understand what does `chmod –R 777` do, and why you should never set permissions to 777.

This article explains the basic Linux permissions model and what the numbers corresponding to the permissions mean.

Understanding Linux File Permissions

In Linux, access to the files is controlled by the operating system using file permissions, attributes, and ownership. Understanding the Linux file system permissions model allows you to restrict access to files and directories only to authorized users and processes and make your system more secure.

Each file is owned by a particular user and a group and assigned with permission access rights for three different classes of users:

- The group members.
- Others (everybody else).

There are three file permissions types that apply to each user class and allows you to specify which users are allowed to read the file, write to the file, or execute the file. The same permission attributes apply for both files and directories with a different meaning:

- The read permission.
  - The file is readable. For example, when the read permission is set, the user can open the file in a text editor.
  - The directory's contents can be viewed. The user can list files inside the directory with the `ls` command.
- The write permission.
  - The file can be changed or modified.
  - The directory's contents can be altered. The user can create new files , delete existing files , move files , rename files ..etc.
- The execute permission.
  - The file can be executed.
  - The directory can be entered using the `cd` command.

File permissions can be viewed using the `ls` command. Here is an example:

```
$ ls -l filename.txt
```

```
Output
-rw-r--r-- 12 linuxize users 12.0K Apr  8 20:51 filename.txt
|[-][-][-]-   [------] [---]
| |  |  | |      |       |
| |  |  | |      |       +----------> 7. Group
| |  |  | |      +------------------> 6. Owner
| |  |  | +-------------------------> 5. Alternate Access Method
| |  |  +---------------------------> 4. Others Permissions
| |  +------------------------------> 3. Group Permissions
```

The first character shows the file type. It can be a regular file (–), directory (d), a symbolic link (l), or any other special type of file.

The next nine characters represent the file permissions, three triplets of three characters each. The first triplet shows the owner permissions, the second one group permissions, and the last triplet shows everybody else permissions.

Permission number

File permission can be represented in a numeric or symbolic format. In this article, we'll focus on the numeric format.

The permission number can consist of three or four digits, ranging from 0 to 7.

When 3 digits number is used, the first digit represents the permissions of the file's owner, the second one the file's group and the last one all other users.

The write, read, and execute permissions have the following number value:

- r (read) = 4
- w (write) = 2
- x (execute) = 1
- no permissions = 0

The permissions digit of a specific user class is the sum of the values of the permissions for that class.

Each digit of the permissions number may be a sum of 4, 2, 1 and 0:

- 0 (0+0+0) – No permission.
- 1 (0+0+1) – Only execute permission.
- 2 (0+2+0) – Only write permission.
- 3 (0+2+1) – Write and execute permissions.
- 4 (4+0+0) – Only read permission.
- 5 (4+0+1) – Read and execute permission.

7 (4+2+1) - Read, write, and execute permission.

For example, if the permission number is set to 750 it means that the file's owner has read, write and execute permission, file's group has read and execute permissions, and other users have no permissions:

- Owner: rwx=4+2+1=7
- Group: r-x=4+0+1=5
- Others: r-x=0+0+0=0

When the 4 digits number is used, the first digit has the following meaning:

- setuid=4
- setgid=2
- sticky=1
- no changes = 0

The next three digits have the same meaning as when using 3 digits number. If the first digit is 0 it can be omitted, and the mode can be represented with 3 digits. The numeric mode `0755` is the same as `755`.

To view the file's permissions in the numeric (octal) notation, use the `stat` command:

```
stat -c "%a" filename
```

```
Output
644
```

Never Use chmod 777

Setting 777 permissions to a file or directory means that it will be readable, writable and executable by all users and may pose a huge security risk.

For example, if you recursively change the permissions of all files and subdirectories under the `/var/www` directory to `777`, any user on the system will be able to create, delete or modify files

If you experience permission issues with your web server, instead of recursively setting the permission to `777`, change the file's ownership to the user running the application and set the file's permissions to `644` and directory's permissions to `755`.

File ownership can be changed using the `chown` command and permissions with the `chmod` command.

Let's say you have a PHP application on your server running as user "linuxize". To set the correct permissions you would run:

```
$ chown -R linuxize: /var/www
$ find /var/www -type d -exec chmod 755 {} \;
$ find /var/www -type f -exec chmod 644 {} \;
```

Only root, the file owner, or user with sudo privileges can change the permissions of a file. Be extra careful when using `chmod`, especially when recursively changing the permissions.

Conclusion

If you are managing a Linux system, it is crucial to know how the Linux permissions work.

You should never set 777 (`rwxrwxrwx`) permissions files and directories permissions. 777 means that anyone can do anything with those files.

Feel free to leave a comment if you have any questions.

chmod terminal

Sign up to our newsletter and get our latest tutorials and news straight to your mailbox.

Your email...

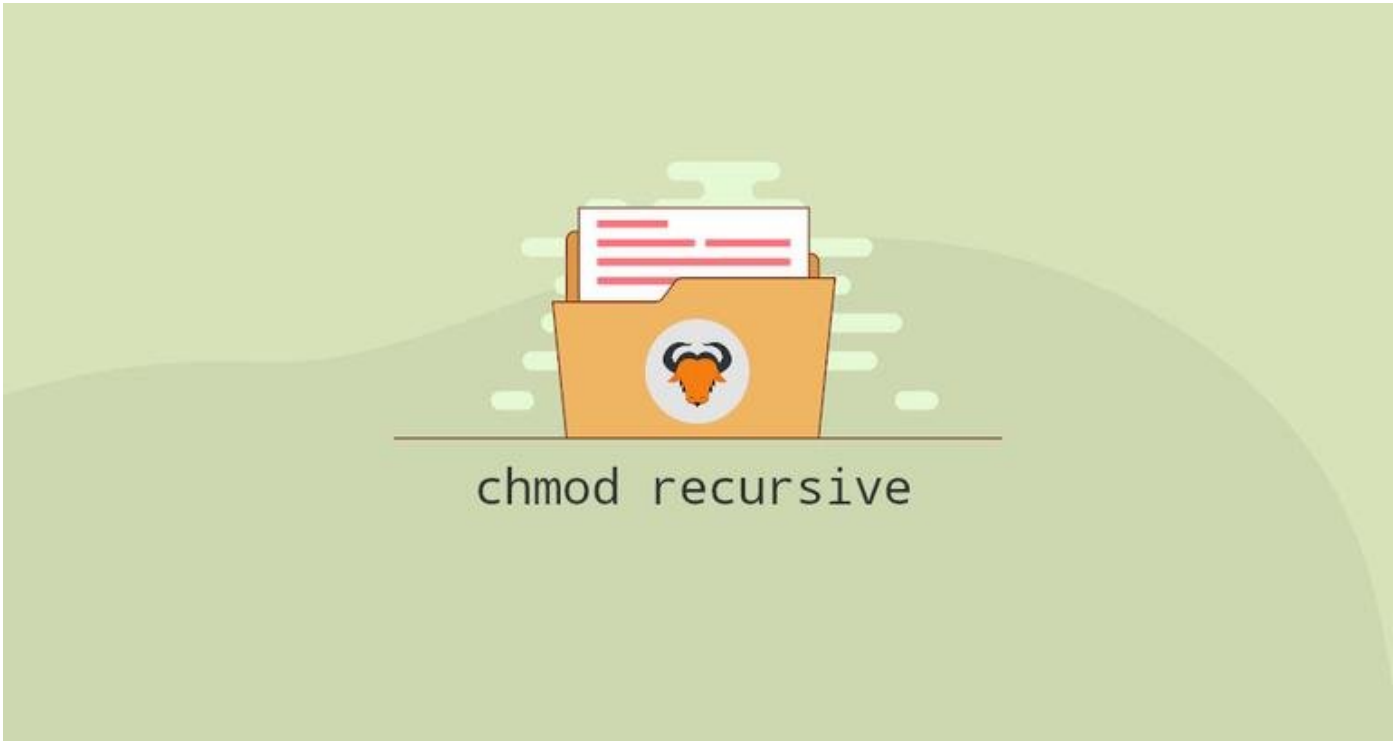
Subscribe

We'll never share your email address or spam you.

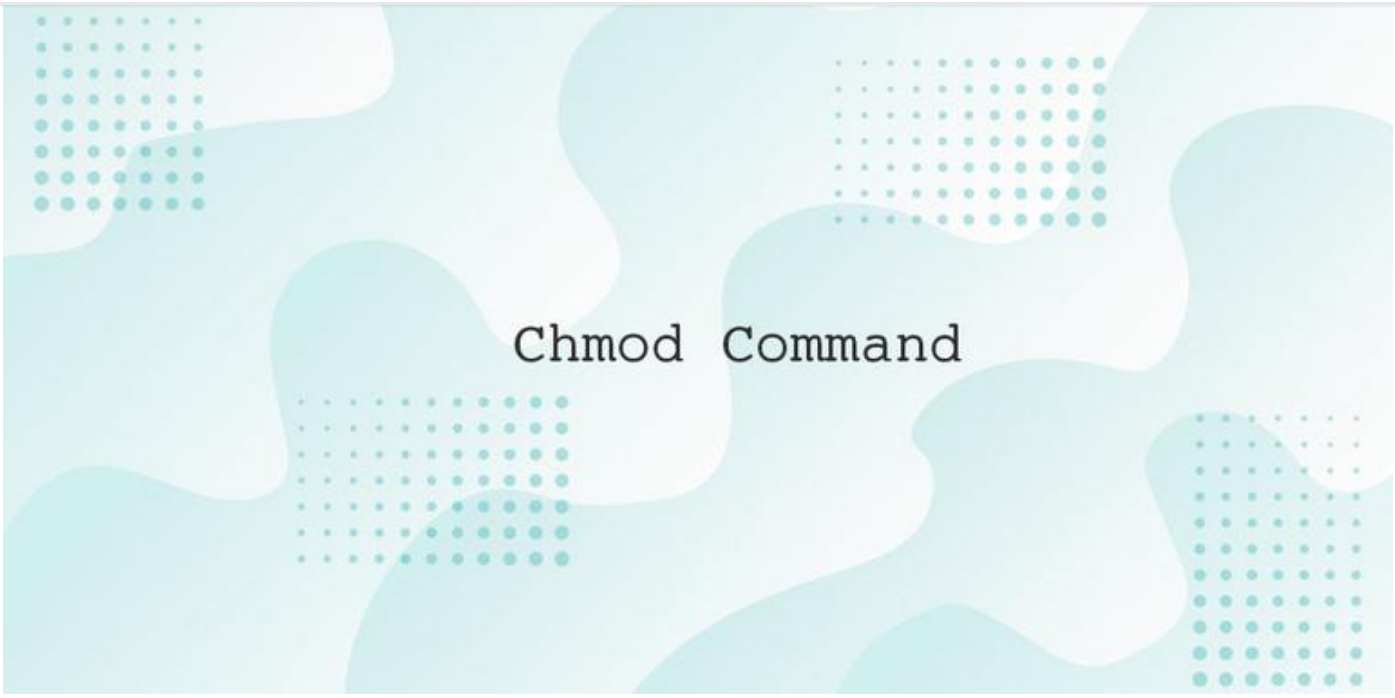
Related Articles

DEC 20, 2019

How to Recursively Change the File's Permissions in Linux

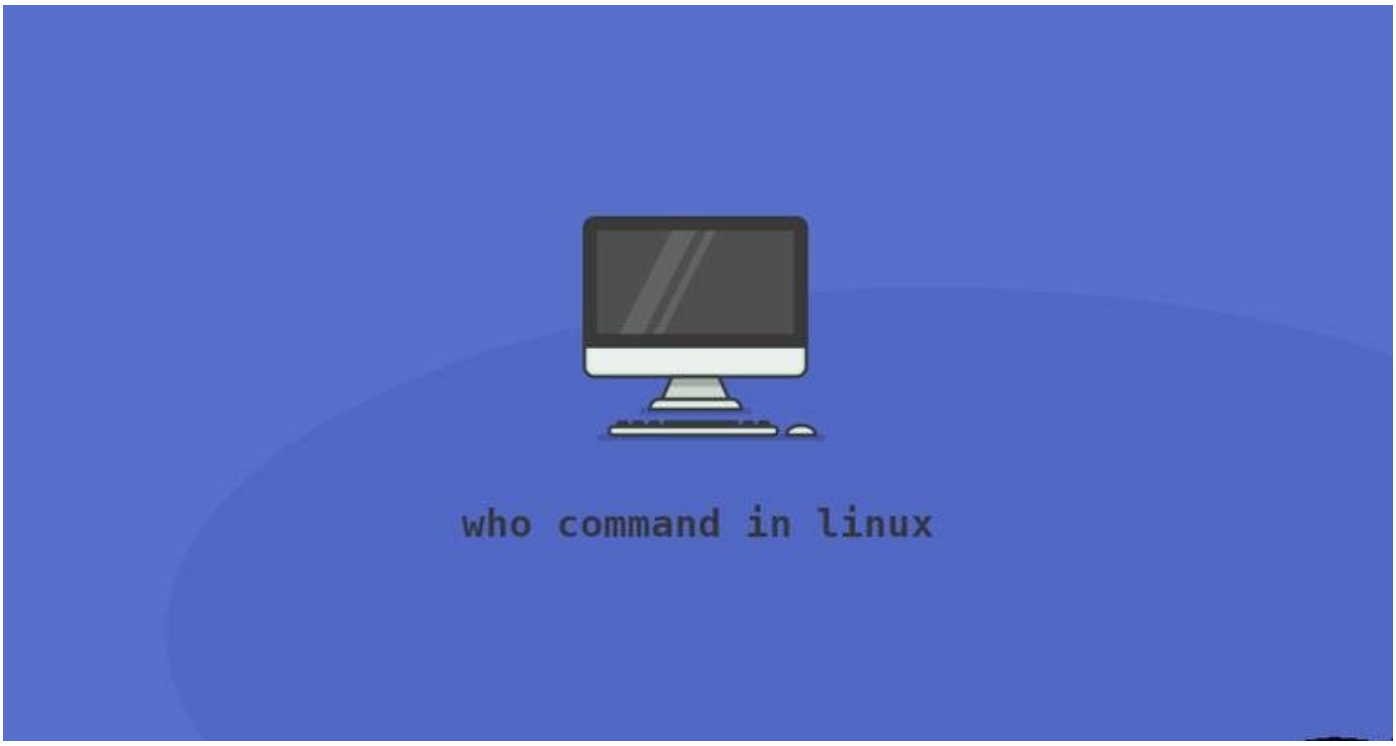


SEP 16, 2019



NOV 17, 2020

who Command in Linux



Write a comment