

# How to use chmod command in Linux

The “**chmod**” command in **Linux** enables you to control the access of scripts, directories, and your system files. This command is utilized to change the Linux file permissions, which seems a complicated method but is simple once you understand its functionality. Before discussing the **chmod** command, let’s go through the fundamentals of Linux file permission.

## What are file permissions in Linux

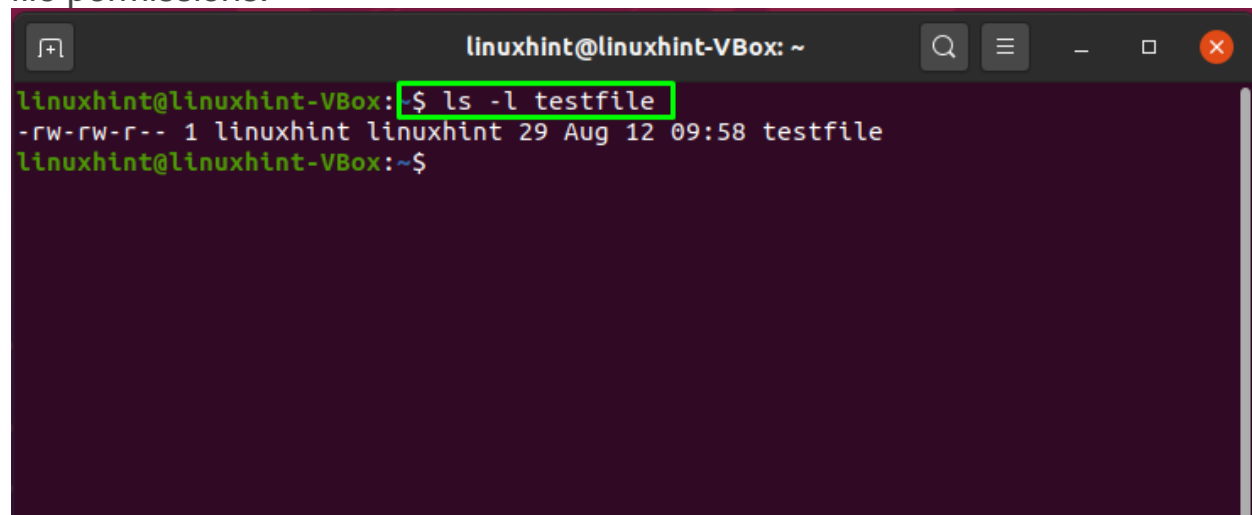
File permission is the type of access associated with a file. Each file in Linux has its owner, a group, and permission access for **three main types of users**: the **file owner**, the **group** members, and **others**. Each of these user classes has **three types of file permissions**: **read**, **write**, and **execute** permissions. Knowing about the file permission helps you specify which users can execute, read, or write the file.

## How to check file permission in Linux

The “**ls**” is used to check the permission of the files present on your system. To view permission of a single file, add its name to the “**ls**” command. For instance, we will execute the below-given command to check the file permissions of the “**testfile**”:

```
$ ls -l testfile
```

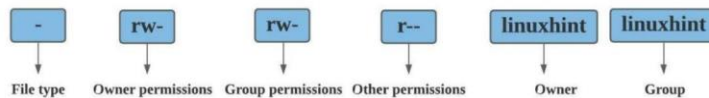
Here, the “**-l**” option is added to get the content of the “**testfile**” including its file permissions:

A screenshot of a Linux terminal window. The window title is 'linuxhint@linuxhint-VBox: ~'. The prompt is 'linuxhint@linuxhint-VBox:~\$'. The command '-\$ ls -l testfile' is entered and highlighted with a green box. The output is '-rw-rw-r-- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile'. The prompt then changes to 'linuxhint@linuxhint-VBox:~\$'.

```
linuxhint@linuxhint-VBox:~$ -ls -l testfile
-rw-rw-r-- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile
linuxhint@linuxhint-VBox:~$
```

The **first character** in the output specifies the **entry type** where “**—**” represents a “**file**”, and “**d**” indicates a “**directory**”. Then we have, **three sets of nine characters**, where **the first three characters set** represent file **owner permissions**, **the next characters set** represent **group permissions**, and **the last set** represents permissions for **other** users that are not considered in the first two categories:

```
-rw-rw-r-- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile
```



Each **set** of permissions contains **three characters**. If the character is a dash “**—**”, it indicates that access permissions are denied. Permission has been given to a user if the character is “**r**”, “**w**”, or “**x**”. The “**r**” in a permission set indicates that the user only has **read permission** which means that the file can be only **opened** and **viewed**, with “**w**,” you will have **write permission** for the specific file, and you will be able to **edit**, **modify** and **delete** the file. Lastly, the “**x**” letter represents the **execute permissions**; for instance, if your file is a C++ program or script, it will have the execute permission with the letter “**x**”.

If the “**ls**” command shows you “**—**” for any set, this means that **no permission** is granted. In another case, “**rwX**” indicates that all types of permission are granted, including reading, writing, and executing. Now you have some background related to file permissions, and it will help you understand the **chmod command** working more efficiently.

## What is chmod command in Linux

The “**chmod**” is an acronym for “**change mode**”. It modifies the access of your system directories, files, and scripts. The “**chmod**” command has specific **modes** that determine the **permission** for modification. These modes are represented by **numerical form (letters)** or **symbolic form (octal numbers)**. When you use the chmod command with numerical form, it follows the below-given syntax:

```
$ chmod [Options] [Filename]
```

In numerical representation, we have these **options**:

- “**0**” represents “**no permission**”.
- “**1**” represents “**execute permission**”.
- “**2**” represents “**write permission**”.
- “**4**” represents “**read permission**”.

If you want to use the symbolic representation, then **chmod** command syntax will be written as follows:

```
$ chmod [Option1] [Operator] [Option2] [Filename]
```

We have the following **options** in the symbolic form:

- “**u**” indicates file **owner**.
- “**g**” indicates **groups**.
- “**o**” indicates **others**.
- “**a**” indicates **all users** as owner, group, and others (ugo).

Whereas the **chmod** command accepts the following **operators**:

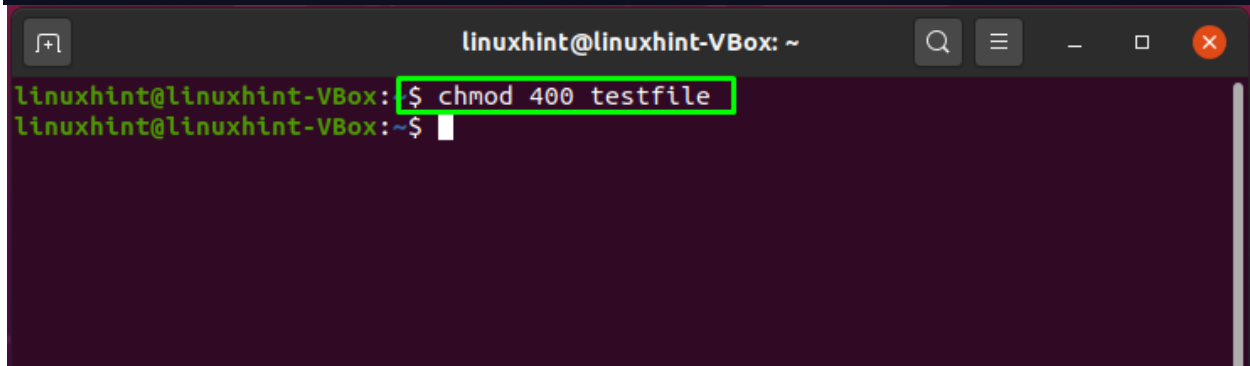
- “**+**”: This operator is utilized to **add specified permissions**.
- “**-**”: This operator is utilized to **remove specified permissions**.
- “**=**”: This operator is utilized to define the **exact file permission** for any user.

Now, let’s check out some **examples** of using the **chmod** command with symbolic form in Linux:

### Example 1: Setting “read by owner only” file permission using chmod command

In this example, we will change the file permissions of “**testfile**” so that only the owner can read it. Other than this permission, no other group or user can read, write or execute this file. Even the owner will not have the access to execute and write something in the file. To do so, use “**4**” as a numerical representation of “**read-only**” and place it at the start of three character set, and adding “**0**” for the “**groups**” and “**others**” mode will not grant any permissions to those users:

```
$ chmod 400 testfile
```

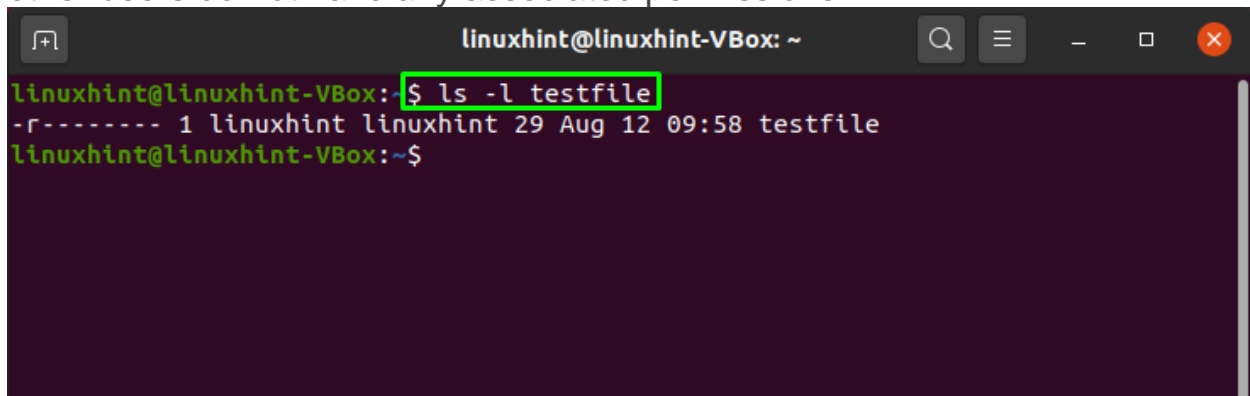
A screenshot of a Linux terminal window. The window title is 'linuxhint@linuxhint-VBox: ~'. The terminal shows the command 'chmod 400 testfile' being entered and executed. The prompt is 'linuxhint@linuxhint-VBox:~\$'. The command is highlighted with a green box. The terminal background is dark purple.

```
linuxhint@linuxhint-VBox:~$ chmod 400 testfile
linuxhint@linuxhint-VBox:~$
```

Now, list the file permission system by using the “**-l**” option in the “**ls**” command:

```
$ ls -l testfile
```

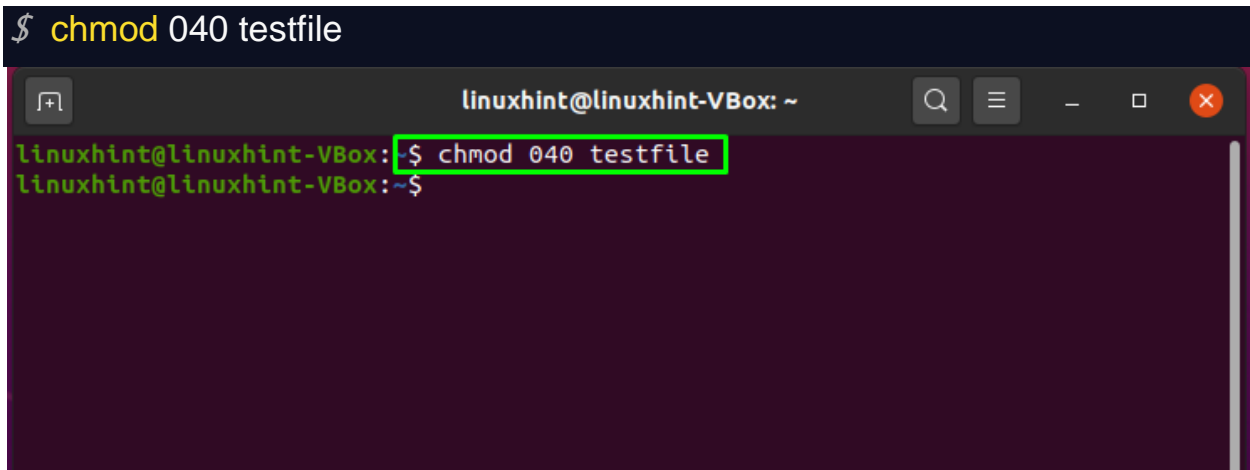
The first “-” indicates that “**testfile**” is a file, and “r” shows that only the file owner has permission to read the file. You can also check that groups and other users do not have any associated permissions:

A terminal window titled 'linuxhint@linuxhint-VBox: ~' with search, menu, and window control icons. The command 'ls -l testfile' is entered and highlighted with a green box. The output shows the file permissions as '-r-----', indicating only the owner has read permission.

```
linuxhint@linuxhint-VBox:~$ ls -l testfile
-r----- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile
linuxhint@linuxhint-VBox:~$
```

## Example 2: Setting “read by group only” file permission using chmod command

Place the “4” as “**group**” mode between the zeroes of “**owner**” and “**others**” mode. This sequence will associate “**read by group only**” permission to the file:

A terminal window titled 'linuxhint@linuxhint-VBox: ~' with search, menu, and window control icons. The command 'chmod 040 testfile' is entered and highlighted with a green box.

```
$ chmod 040 testfile

linuxhint@linuxhint-VBox:~$ chmod 040 testfile
linuxhint@linuxhint-VBox:~$
```

Again check out the changes we made into the “**testfile**” by using the below-given “**ls**” command:

A terminal window showing the command 'ls -l testfile' entered.

```
$ ls -l testfile
```

Here “r” represents the “**read**” permission:

```
linuxhint@linuxhint-VBox: ~  
linuxhint@linuxhint-VBox:~$ ls -l testfile  
----r----- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile  
linuxhint@linuxhint-VBox:~$
```

Similarly, you can allow the “**read by others only**” file permission by defining the “**004**” mode in the `chmod` command.

### Example 3: Setting “**write by owner only**” file permission using `chmod` command

In numerical representation of the modes, “**2**” indicates the “**write**” permissions. Place the “**2**” at the start of the permission set, and add two zeros after that:

```
$ chmod 200 testfile
```

Execution of this command will only allow the file owner to write into the test file:

```
linuxhint@linuxhint-VBox: ~  
linuxhint@linuxhint-VBox:~$ chmod 200 testfile  
linuxhint@linuxhint-VBox:~$
```

Confirm the changed file permission by using the “**ls**” command:

```
$ ls -l testfile
```

Here, “**w**” represents the “**write**” permission:

```
linuxhint@linuxhint-VBox: ~  
linuxhint@linuxhint-VBox:~$ ls -l testfile  
--w----- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile  
linuxhint@linuxhint-VBox:~$
```

Similarly, you can define the “**write by group only**” permission with the “**020**” mode and “**write by other only**” using the “**002**” mode.

#### Example 4: Setting “**execute by owner only**” file permission using **chmod** command

In the **chmod** command, the “**1**” digit represents the **execute** mode. If you want to set the “**execute by owner only**” permission, then execute the below-given command in your terminal:

```
$ chmod 100 testfile  
linuxhint@linuxhint-VBox: ~  
linuxhint@linuxhint-VBox:~$ chmod 100 testfile  
linuxhint@linuxhint-VBox:~$
```

Again, list out the permission of the “**testfile**” by using the “**ls**” command:

```
$ ls -l testfile
```

Here, “**x**” represents the “**execute permission**”:

```
linuxhint@linuxhint-VBox: ~  
linuxhint@linuxhint-VBox:~$ ls -l testfile  
---x----- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile  
linuxhint@linuxhint-VBox:~$
```

In the same manner, if you want to change the file permission to “**execute by group only**,” then define the mode as “**010**,” and for allowing other users to execute the file, add “**001**” as permission mode in the **chmod** command:

```
$ chmod a-x testfile
```

### Example 5: Setting “read by everyone” file permission using **chmod** command

Using symbolic links, if you change the file permission to “**read by everyone**,” then execute the below-given command in your terminal:

```
$ chmod a+r testfile
```

Here “**a**” represents “**all users**”, “**r**” indicates “**read**” permissions, and the “**+**” operator is used to add the read permission to the specified users:

```
linuxhint@linuxhint-VBox: ~  
linuxhint@linuxhint-VBox:~$ chmod a+r testfile  
linuxhint@linuxhint-VBox:~$
```

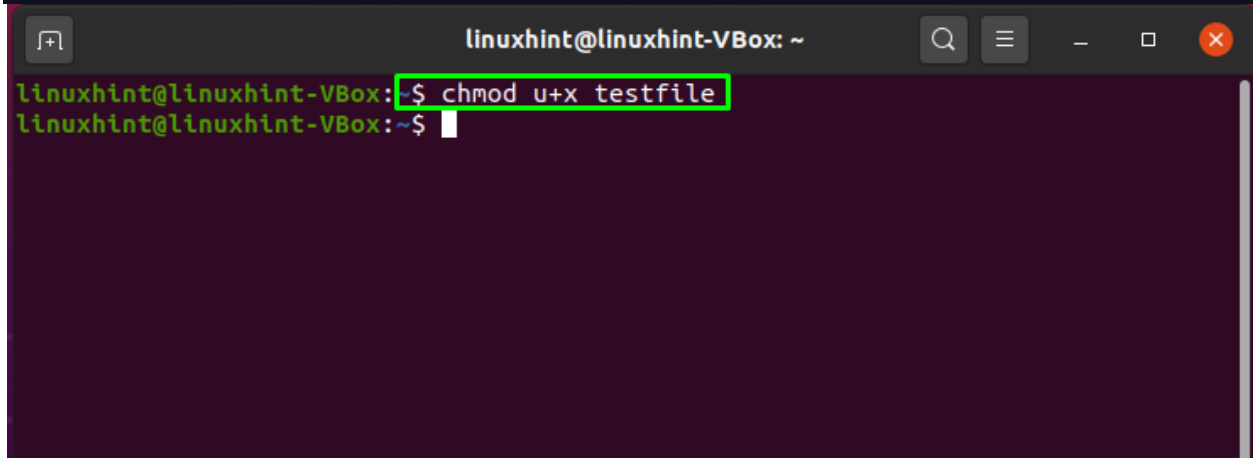
Check the changed file permission by utilizing the “**ls**” command:

```
$ ls -l testfile
```

## Example 6: Setting “execute by owner” file permission using chmod command

The “u+x” permission mode indicates that “**execute**” permission will be granted to the file “**owner**”:

```
$ chmod u+x testfile
```

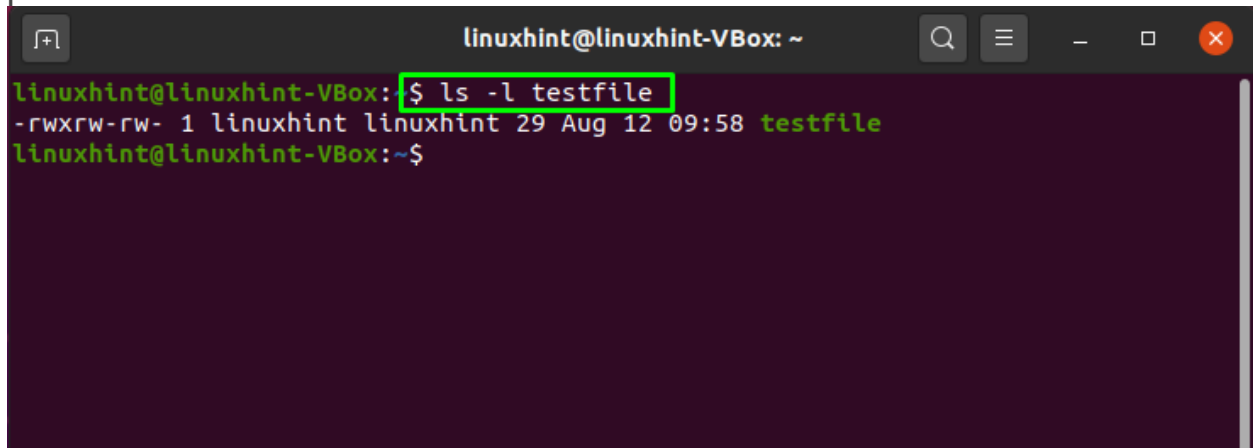
A terminal window titled 'linuxhint@linuxhint-VBox: ~' with search, menu, and window control icons. The prompt is 'linuxhint@linuxhint-VBox:~\$'. The command 'chmod u+x testfile' is entered and highlighted with a green box. The next line shows the prompt 'linuxhint@linuxhint-VBox:~\$' with a cursor, indicating the command has been executed.

```
linuxhint@linuxhint-VBox:~$ chmod u+x testfile
linuxhint@linuxhint-VBox:~$
```

Now, execute the following to verify the file permission changes:

```
$ ls -l testfile
```

“x” in the set of owner permission shows that now the file owner has the permission to execute the file:

A terminal window titled 'linuxhint@linuxhint-VBox: ~' with search, menu, and window control icons. The prompt is 'linuxhint@linuxhint-VBox:~\$'. The command 'ls -l testfile' is entered and highlighted with a green box. The output shows the file permissions '-rwxrw-rw-' for the owner, group, and others, followed by the file name 'testfile'. The next line shows the prompt 'linuxhint@linuxhint-VBox:~\$' with a cursor, indicating the command has been executed.

```
linuxhint@linuxhint-VBox:~$ ls -l testfile
-rwxrw-rw- 1 linuxhint linuxhint 29 Aug 12 09:58 testfile
linuxhint@linuxhint-VBox:~$
```