

# Day 6: File Permissions and Access Control Lists

Day6 of #90DaysofDevops challenge



Kunal Maurya · Feb 28, 2023 · 📖 5 min read

## File Permissions and Access Control Lists



### ☰ TABLE OF CONTENTS

- Linux file ownership
  - User owner
  - Group owner
  - Other
- Linux File Permissions
  - Chmod
    - How to Change Permissions using Symbolic Mode
    - How to Change Permissions using Absolute Mode
  - Chown
  - Chgrp
- Access control lists
  - getfacl
  - setfacl

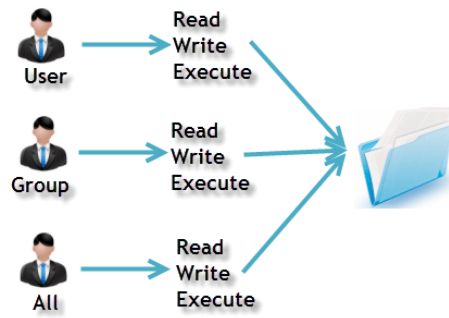
In Linux, file permissions are a crucial aspect of ensuring that your files are secure and can only be accessed or modified by authorized users.

In this blog, we will explain what file permissions are, how they work, and provide some examples to illustrate how they can be used.

## Linux file ownership

In Linux, there are three types of file owners: the user owner, the group owner, and the other (or world) owner.

### Owners assigned Permission On Every File and Directory



## User owner

The user owner is the person who created the file or directory and has complete control over the file. The user owner can read, write, and execute the file, as well as change the permissions and ownership of the file.

## Group owner

The group owner is a collection of users who have similar permissions for accessing the file or directory. The group owner can be changed by the user owner, but only members of the group can be assigned as group owners.

## Other

The other owner, also known as the world owner, refers to all other users who are not the user owner or group owner.

## Linux File Permissions

File permissions are represented using a **10-character string**, where the first character represents the file type (file, directory, or link), and the next nine characters represent the file permissions.

The nine characters are divided into three groups, with each group representing the permissions for the **owner**, **group**, and **other** users, respectively. Within each group, the three characters represent **read**, **write**, and **execute** permissions, in that order.

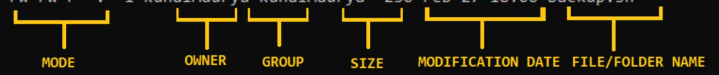
The letters used to represent these permissions are as follows:

- **r**: read permission
- **w**: write permission
- **x**: execute permission
- **-**: no permission

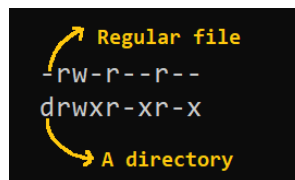
<b>rwX</b>	<b>rwX</b>	<b>rwX</b>
<b>user</b>	<b>group</b>	<b>other</b>

we can find permissions of files and folders using long listing ( `ls -l` ) r by other options on a Linux terminal.

```
[KunalMaurya@Kunal ~]$ ls -ltr
total 16
-rwxrwxrwx. 1 KunalMaurya KunalMaurya 175 Feb 27 11:36 check_disk.sh
-rw-r--r--. 1 KunalMaurya KunalMaurya 26 Feb 27 12:17 test_cron_first.txt
drwxrwxr-x. 92 KunalMaurya KunalMaurya 4096 Feb 27 17:55 day5
drwxrwxr-x. 2 KunalMaurya KunalMaurya 6 Feb 27 18:00 backup
-rw-rw-r--. 1 KunalMaurya KunalMaurya 236 Feb 27 18:06 backup.sh
```



In the output above, `d` represents a directory and `-` represents a regular file.



## Chmod

The `chmod` command is used to change the permissions of a file or directory. It can be used in various ways, depending on whether you want to add or remove permissions, and whether you want to change the permissions for the owner, group, or other users.

**Syntax of `chmod`:**

```
chmod permissions filename
```

COPY

We can change permissions using two modes:

1. **Symbolic mode:** this method uses symbols like `u`, `g`, `o` to represent users, groups, and others. Permissions are represented as `r`, `w`, `x` for read write and execute, respectively. You can modify permissions using `+`, `-` and `=`.
2. **Absolute mode:** this method represents permissions as 3-digit octal numbers ranging from 0-7.

## How to Change Permissions using Symbolic Mode

The table below summarizes the user representation:

USER REPRESENTATION	DESCRIPTION
u	user/owner
g	group
o	other

We can use mathematical operators to add, remove, and assign permissions. The table below shows the summary:

OPERATOR	DESCRIPTION
+	Adds permission to a file or directory
-	Removes the permission
\=	Sets permission if not present before. Also overrides the permissions if set earlier.

#### Example:

Suppose, I have a script and I want to make it executable for owner of the file .

Current file permissions are as follows:

Let's give execute(x) permission to user :

```
chmod u+x mymotd.sh
```

COPY 

#### Output:

## How to Change Permissions using Absolute Mode

The absolute mode uses numbers to represent permissions and mathematical operators to modify them.

The below table shows how we can assign relevant permissions:

PERMISSION	PROVIDE PERMISSION
read	add 4
write	add 2
execute	add 1

Permissions can be revoked using subtraction. The below table shows how you can remove relevant permissions.

PERMISSION	REVOKE PERMISSION
read	subtract 4
write	subtract 2
execute	subtract 1

#### Example:

- Set `read` (add 4) for `user` , `read` (add 4) and `execute` (add 1) for group, and only `execute` (add 1) for others.

```
chmod 451 file-name
```

COPY 

## Chown

The `chown` command is used to change the owner of a file or directory. You need to have root privileges to change the owner of a file that you don't own.

Syntax of `chown` :

```
chown user filename
```

COPY 

Example:

Let's transfer the ownership from user `KunalMaurya` to user `Maurya`.

```
sudo chown Maurya backup.sh
```

COPY 

## Chgrp

The `chgrp` command is used to change the group ownership of a file or directory. You need to have root privileges to change the group ownership of a file that you don't own.

Example:

```
$ chgrp group example.txt
```

COPY 

Output:

- The group ownership of `example.txt` is changed to `group`.

## Access control lists

Access control lists (ACLs) in Linux are a set of permissions that can be applied to a file or directory to grant or restrict access to it for specific users or groups. While traditional Linux file permissions (user, group, and other) only allow for basic access control, ACLs provide more granular control over file access.

ACLs are implemented using a set of rules that are associated with each file or directory. These rules specify which users or groups have what level of access to the file or directory. There are two types of ACLs in Linux:

1. **Basic ACLs:** These are the traditional UNIX-style permissions (read, write, execute) that are associated with each file or directory.
2. **Extended ACLs:** These provide additional permissions beyond the basic permissions. These permissions include things like setting file attributes, setting file ownership, and setting the maximum size of a file.

## getfacl

`getfacl`: This command is used to display the ACLs associated with a file or directory. Here's an example:

## setfacl

`setfacl`: This command is used to modify or remove the ACLs associated with a file or directory.

- `-m`: modify ACLs
- `-x`: remove ACLs
- `-b`: remove all ACLs

Example:

`setfacl -m`: This command is used to modify the ACLs associated with a file or directory. Here's an example:

Thank you for reading! Hope you find this article helpful.

~Kunal



## Subscribe to my newsletter

Read articles from directly inside your inbox.  
Subscribe to the newsletter, and don't miss out.

SUBSCRIBE

Linux

Devops

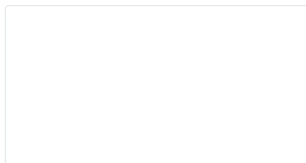
file-permission

acl

access control

### MORE ARTICLES

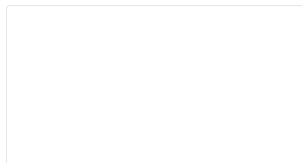
Kunal Maurya



#### Day 5 Task: Advanced Linux Shell Scripting for DevOps Engineers

Memory Commands free: The free command is a utility in Linux that shows the amount of free and used...

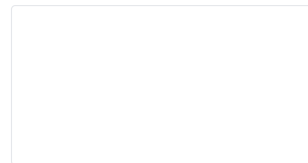
Kunal Maurya



#### Day 4 Task: Basic Linux Shell Scripting for DevOps Engineers.

What is Shell? In Linux architecture, the shell is a command-line interface that provides a way for...

Kunal Maurya



#### Day 3 Task: Basic Linux Commands

Cat Command The cat command is a Unix and Linux utility that stands for "concatenate." It is used to...

[Archive](#) · [Privacy policy](#) · [Terms](#)



**Publish with Hashnode**

Powered by [Hashnode](#) - Home for tech writers and readers