Users, groups, permissions in Linux

How to manage permissions and ownership for users, groups, and all others to resources such as directories and files.

Introduction

Managing access to resources is a fundamental task for sysadmins. This responsibility consists of three components: identities, resources, and permissions. This article covers several user, group, and file management commands to control access to resources. The article uses a "How do I...?" format, and it assumes you have a few resources to work with. Specifically, I cover the following topics:

- Creating directories and files
- Creating users and groups
- Assigning permissions to users and groups
- Managing ownership and associated groups
- Setting permissions with absolute and symbolic modes

Setting up a lab_field

I've been in IT for about 11 years, and more than half of that time was spent as a technical trainer. That means that the things that I write are usually structured as some sort of lab or other hands-on opportunity. It's just how I cover material. With that in mind, I'll assume you have a couple of identities and resources to experiment with as you read the rest of this article. You can use the following commands to set up a lab_field. It's best to do this on a virtual machine rather than your personal Linux box, but these tasks are relatively harmless.

To start with, create two new users and two new groups to work with. Note that you do not need to configure passwords for the users in this exercise, as you won't log on with those accounts.

#useradd user1

#useradd user2

#groupadd groupA

#groupadd groupB

Note: You would use the passwd user1 command to set the user's password.

In your home directory, create a new directory named lab_field.

#mkdir lab_field

Change into the ~/lab_field directory by using the cd command. You are ready to work with the commands and concepts below.

NB: When you've completed the article and learned the techniques I've covered, delete the two user accounts, the groups, and the lab_field directory. Use rm -fR /lab_field, userdel user1, and groupdel groupA to remove the resources.

Creating directories and files

Use the mkdir command to create directories. The touch command is one of many ways to create files.

How do I create a directory named mydata?

#mkdir mydata

How do I create a directory path (a series of directories that don't yet exist)?

#mkdir -p mydata/data/2021data

Note: The goal here is to create the 2021data directory, but the given path's data directory does not yet exist. The -p option creates parent directories as needed to complete the path.

How do I create a file named file1?

#touch file1

How do I create several files at once?

#touch file1 file2 file3 file4

Managing Ownership and groups

In the lab_field directory, display the current owner and group associated with the mydata directory and the files.

How do I display permission, owners, and groups?

#ls -l

The *ls -l* command displays directory contents in long format. The long format contains both permissions and ownership. You can see that the user account that created the mydata also owns the resources in it. The group association is also that user's primary group.

How do I change the user/owner associated with file1?

#chown user2 file1

How do I change the group associated with file1?

#chown :groupA file1

How do I change the owner and group at the same time for file2?

#chown user2:groupA file2

There is a specific chgrp command, but I prefer only to memorize one command (chown) and apply it to both functions (user and group associations) rather than chown for the user and then have to recall chgrp for the group.

So how do I use chgrp?

#chgrp groupB file1

How do I change the user/group for a directory and all of its contents?

chown -R user1:groupA mydata

The above task provides a recursive configuration. Technically, recursive commands are repeated on each specified object. Effectively, recursive means "this and everything in it." In the above example, you are configuring the related user/group for the mydata directory and everything in it. Without the -R option, you would only affect the mydata directory itself, but not its contents.

Managing permissions

The change mode or chmod command sets permissions. The syntax is straight-forward:

chmod permissions resource-name
Here are two examples of manipulating permissions for file2:

#chmod 740 file2

#chmod u=rwx,g=r,o-rwx file2

But wait! Those appear to be radically different examples (they're not, actually). What are all those letters and numbers?

We need to discuss absolute mode and symbolic mode.

Using absolute mode

Absolute mode is one of two ways of specifying permissions. I've seen this mode referred to as octal or numeric mode, but the term I learned was absolute. That term also makes the most sense to me because it's an absolute statement of the desired permissions. I always told my students that this seemed like the most complex of the two modes but is actually the simplest. Usually, they agreed.

Each access level (read, write, execute) has an octal value:

Access level	Octal value
Read	4
Write	2
Execute	1

Each identity (user, group, others) has a position:

Identity	Position
User	First or left-most
Group	Middle
Others	Last or right-most

The absolute mode syntax states the desired permissions from left to right.

How do I grant the user (owner) read, write, and execute, the group read-only, and all others no access to file2 by using absolute mode?

#chmod 740 file2

The three permissions values are associated with identities:

ugo
740

The 7 is assigned to the user and is the sum of 4+2+1 or read+write+execute (full access)
The 4 is assigned to the group and is the sum of 4+0+0 (read-only)
The 0 is assigned to others and is the sum of 0+0+0 (no access)
In this example, the user has rwx, the group has r only, and all others have no access to file2.

Let's look at one more example.

How do I grant the user (owner) read and write, the group read-only, and all others read-only to file2?

#chmod 644 file2

The user has 6 (read and write)
The group has 4 (read-only)
All others have 4 (read-only)
I find this easier because there are no calculations involved. I'm not concerned with adding or subtracting specific permissions based on the current settings. Instead, I say, "set the permissions to be this," and that's the end result I get. It's an absolute statement.

How do I set permissions for the mydata directory and all of its contents by using absolute mode?

#chmod -R 744 mydata

How do I use symbolic mode?

Symbolic mode uses more symbols, but the symbols are simpler to understand. That's attractive to sysadmins that are new to standard Linux permissions.

Each access level has a symbol:

Access level	Symbol
Read	r
Write	w
Execute	x

Each identity has a symbol:

Identity	Symbol
User	u
Group	g
Others	o

There are also operators to manipulate the permissions:

Task	Operator
Grant a level of access	+
Remove a level of access	-
Set a level of access	=

The general chmod command syntax is the same: *command permissions directory/file*

Here is an example:

How do I remove the read permissions from others for file2 by using symbolic mode?

#chmod o-r file2

This example removes (-) the read (r) permission from others (o) for file2.

Here's another simple example:

How do I grant the read and write permissions to the group for file2?

#chmod g+rw file2

This one gives (+) read and write (rw) to the group (g) for file2.

How do I set permissions for a directory and all of its contents by using symbolic mode?

#chmod -R u=rwx,g+rw,o-rwx mydata