chown example: recursive update

chown command

chown is a basic Unix command, super useful and very powerful. I have provided a **chown example** earlier, but would like to show another common way it's used.

chown: update ownership recursively

chown command uses -R option to apply changes recursively.

For example, if I have a **/Users/greys/unixtutorial** directory with the following layout:

```
file1
file2
dir1/file3
```

… then it’s very easy to show the difference between chown and chown -R commands.

Let’s go to the **/Users/greys/unixtutorial** directory:

```
greys@maverick:~ $ cd /Users/greys/unixtutorial
greys@maverick:~/unixtutorial $
```

Now, let’s look at the files with the **ls command**:

```
greys@maverick:~/unixtutorial $ ls -al *
 -rw-r--r--  1 greys  staff  0  8 Oct 22:55 file1
 -rw-r--r--  1 greys  staff  0  8 Oct 22:55 file2
dir1:
 total 0
 drwxr-xr-x  3 greys  staff   96  8 Oct 22:55 .
 drwxr-xr-x  5 greys  staff  160  8 Oct 22:55 ..
 -rw-r--r--  1 greys  staff    0  8 Oct 22:55 file3
```

Everything belongs to my own user, **greys** and my primary group: **staff**.

Time to change ownership of everything in the current directory:

```
greys@maverick:~/unixtutorial $ sudo chown root:wheel *
```

Checking files with **ls** again, I can see that immediate contents of the /Users/greys/unixtutorial directory (where I was at the time of running chown) got updated ownership: file1 and file2, along with dir1, now belong to root:wheel.

But **file3** was in a **dir1** subdirectory, so it stayed intact and still belongs to greys:staff:

```
greys@maverick:~/unixtutorial $ ls -al *

-rw-r--r--  1 root  wheel  0  8 Oct 22:55 file1

-rw-r--r--  1 root  wheel  0  8 Oct 22:55 file2
dir1:
 total 0
 drwxr-xr-x  3 root   wheel   96  8 Oct 22:55 .
 drwxr-xr-x  5 root   wheel  160  8 Oct 22:55 ..
 -rw-r--r--  1 greys  staff    0  8 Oct 22:55 file3
```

That's because without the -R (recursive) option, **chown** will only inspect and update files in the current directory, but not in any of its subdirectories (or their subdirectories, and so on).

Running the same command with -R option does the trick:

```
greys@maverick:~/unixtutorial $ sudo chown -R root:wheel *
greys@maverick:~/unixtutorial $ ls -al *
-rw-r--r--  1 root  wheel  0  8 Oct 22:55 file1
-rw-r--r--  1 root  wheel  0  8 Oct 22:55 file2
dir1:
total 0
drwxr-xr-x  3 root  wheel   96  8 Oct 22:55 .
drwxr-xr-x  5 root  wheel  160  8 Oct 22:55 ..
```

```
-rw-r--r--  1 root  wheel    0  8 Oct 22:55 file3
```

Hope you learned something new, come back for more!

See Also

- **Unix Basic Commands**
- **ls**
- chown command
- chmod vs chown
- Running chown with sudo

How To Change Ownership of Files and Directories in Unix

I've just been asked a question about changing the ownership of files from one Unix user to another, and thought it probably makes sense to have a quick post on it.

File ownership in Unix

Just to give you a quick reminder, I'd like to confirm that every single file in Unix belongs to some user and some group. There simply isn't a way to create a file without assigning ownership. I've briefly touched the topic of confirming file ownership in Unix before, so today I will simply build on that and show you how to change ownership of files.

Here’s a setup for today: I have created a temporary directory with a few files and made myself the owner of all the files:

```
ubuntu$ ls -al /home/greys/example/
total 12
drwxr-xr-x  3 greys admin 4096 Feb  9 03:55 .
drwxr-xr-x 13 greys greys 4096 Feb  9 03:54 ..
drwxr-xr-x  2 greys admin 4096 Feb  9 03:55 dir1
-rw-r--r--  1 greys admin    0 Feb  9 03:54 file1
-rw-r--r--  1 greys admin    0 Feb  9 03:55 file2
```

As you can see from this listing, the owner (third field in each line) is my username – **greys**. The next field is a Unix group of each file’s owner – **admin** in my example.

Changing owner of a file in Unix

Changing file ownership means only updating the association between a Unix user and a file, and nothing else. When you’re changing the owner of a file, no data contained in a file is changed.

To change the owner of a file, you need to use the **chown command** (easy enough to remember: CHange OWNer – **chown**), with the following syntax:

```
ubuntu$ chown nobody file1
```

In this command, **nobody** is the username of the new owner for a list of files. In my example, the only file we’d like to change ownership for is **file1**.

It is important to realize that you can only change file ownership as a super-user (root). Any regular Unix user cannot change the ownership of any file, and I’d like to explain why.

Indeed, some people are surprised: if I’m the owner of a given file, why can’t I change the ownership for it? That’s because transferring the ownership will mean some other Unix user will become the owner of the file(s) in question. So changing

ownership is like making a decision not only for yourself, but for the new owner of the files.This is only something a super-user – special administrative account in Unix – can do.

The same logic applies to other people not being able to become owners of your files, even if they're willing to assume the new responsibilities of owning files. They cannot revoke your ownership, because each Unix user is only allowed to make decisions and take actions on his/her own behalf.

That's why you will probably see an error like this if you attempt to change ownership of a file as your own regular Unix user:

```
ubuntu$ id
uid=1000(greys) gid=113(admin) groups=33(www-data),113(admin)
ubuntu$ chown nobody file1
chown: changing ownership of `file1': Operation not permitted
```

But if we become root:

```
ubuntu$ sudo -i
[sudo] password for greys:
ubuntu#
```

… we'll have no problem changing owners for any files:

```
ubuntu# cd /home/greys/example
ubuntu# chown nobody file1
ubuntu# ls -l file1
-rw-r--r-- 1 nobody admin 0 Feb  9 03:54 file1
```

Changing owner for multiple files

If you're going to change owner of a few files, this can easily be done using either a full list of files or a mask.

First, here's an example of updating ownership for a specified list of files (and as you can see, directories as well):

```
ubuntu# chown nobody file2 dir1
ubuntu# ls -al
```

```
total 12
drwxr-xr-x  3 greys  admin 4096 Feb  9 03:55 .
drwxr-xr-x 13 greys  greys 4096 Feb  9 03:54 ..
drwxr-xr-x  2 nobody admin 4096 Feb  9 03:55 dir1
-rw-r--r--  1 nobody admin    0 Feb  9 03:54 file1
-rw-r--r--  1 nobody admin    0 Feb  9 03:55 file2
```

IMPORTANT: here's one thing which is often forgotten: when you're changing an owner of a directory, this DOES NOT automatically change owner of all the files which already exist in this directory. So, if we check the file3 in dir1 after the example above, we can see that even though dir1 now belongs to user nobody, file3 in it still belongs to me:

```
ubuntu# ls -l dir1/file3
-rw-r--r-- 1 greys admin 0 Feb  9 03:55 dir1/file3
```

If your intention is to change ownership of all the files and directories of a certain location in your filesystem, you need to use a **-R** option of the **chown command**, which means recursive ownership change:

```
ubuntu# chown -R nobody dir1
ubuntu# ls -l dir1/file3
-rw-r--r-- 1 nobody admin 0 Feb  9 03:55 dir1/file3
```

And just to further demonstrate this, I'm going to change owner of all the files and directories in **/home/greys/example** directory back to my own username, **greys**:

```
ubuntu# chown -R greys /home/greys/example/
ubuntu# ls -l /home/greys/example/
total 4
drwxr-xr-x 2 greys admin 4096 Feb  9 03:55 dir1
-rw-r--r-- 1 greys admin    0 Feb  9 03:54 file1
-rw-r--r-- 1 greys admin    0 Feb  9 03:55 file2
```

Changing group ownership for a file

Similar to the chown command, there's a command specifically helping you with changing not the owner (user) of a file.

IMPORANT: unlike chown command, chgrp can be used by non-privileged (regular) users of a system. So you don't have to be root if you want to change a group ownership for some of your files, provided that you're changing the ownership to a group you're a member of.

For example, I'm a member of quite a few groups on one of my Ubuntu servers:

```
ubuntu$ id greys
uid=1000(greys)                              gid=1000(greys)
groups=1000(greys),4(adm),20(dialout),24(cdrom),46(plugdev),11
4(lpadmin),115(sambashare),116(admin)
```

Now, if I create a new file, it will by default belong to my primary group (called **greys**, just like my username):

```
ubuntu$ touch file
ubuntu$ ls -al file
-rw-r--r-- 1 greys greys 0 2012-09-20 10:48 file
```

I can now change group ownership of this file, in this case to a group admin, which I'm also part of.

```
ubuntu$ chgrp admin file
```

and this is just to confirm that the change actualyl happened:

```
ubuntu$ ls -al file
-rw-r--r-- 1 greys admin 0 2012-09-20 10:48 file
```

That's it for today, good luck with changing file owners on your Unix system!

Recommended books:

See also:

- [Finding the owner of a file in Unix]

- **Find files which belong to a given Unix user**
- **What to do when numeric user id is shown instead of username**