8.9.2 Managing File Permissions

Click one of the buttons to take you to that part of the video.

Managing Permissions 0:04-0:22

In this demonstration, we're going to spend some time learning about how you manage permissions that are assigned to files in the Linux file system. These permissions need to be managed very carefully to ensure that you grant the appropriate level of access to your users. Basically, a user should have just enough access to do his or her job and no more.

ls Command 0:21-8:29

Let's begin by using the 'ls' command to view the contents of my user's /home directory, the rtracy user. Within this directory, you see that I have a folder created called RandD. This is used to store research and development files. For our purposes today we're going to assume that my rtracy user account is a member of the RandD team, and we'll be storing files in this folder. In fact, other users will also collaborate with me on this RandD project and will also need access to the files in this folder.

Let's begin by creating a file--a new document file--within RandD, but let's do it as the root user. Switch to my root user account. I'll use the 'vi' command to create a new document, and add some text to it '/home/rtracy/RandD' and let's name the file 'widget_design_specs.odt'. This will be an Open Office word processing file. We'll just insert some regular old text in here for now. Save our changes, 'exit' the editor, and now we've created this file in the RandD folder. Let's use the 'ls -l' command to view the contents of the home/rtracy/RandD folder. Because we used the ls -l option here, were able to view the ownership of the file that we created, as well as the mode that's been assigned to that file.

Two things should stand out to you right away. First, look at the ownership of this file. Because I was logged in as root when I created that file, the owning user is root and the owning group is the root group, not the rtracy user. This file ownership will have a dramatic impact on the level of access that my rtracy user has, because of the mode that's been automatically assigned to this file. The first three characters in the mode describe the permissions that are assigned to the owning user--in this case, the root user. Root has read and write access to this file. The next three characters in the mode specify the permissions that are assigned to the group that owns this file--in this case, the root group. Members of the root group had read access to this file. The permissions that are assigned to all other authenticated users are specified by the last three characters in the mode. Any user who is not root and is not a member of the root group will have these permissions assigned, which is just basically read access again.

With that in mind, what level of access will my rtracy user have to this file? Let's find out. I'm going to enter 'exit' to switch back to my rtracy user account. Let's run a word processing application here. I am currently logged in to my graphical environment as my rtracy user. Let's see what level of access I have to that file in this word processor as rtracy. Open and we want to go into RandD and let's open up the file we just created. Okay, I am able to open the file, so I have read level access. I can see the contents, but notice up here something's different in the title bar of the document. It says, "read-only". Because my rtracy user is not the root user and because the rtracy user is not a member of the root group, I received the permissions that are assigned to other authenticated users, which was just read. I have read-only access to the file. I can't come in here and make changes to the file and save them. Go ahead and 'exit' out of editor.

The first thing we probably should do is change ownership of this file. In order to do that, I do need to be logged in as my root user. Remember that the root user is the only one who can change the user that owns the file. Let's go ahead and switch into the '/home/rtracy/RandD' directory. If we do an 'ls -l' again, we can see the mode of the file that we're working with. In this scenario, we want to change both the user that owns the file and the group. If I use the 'tail' command here to view the /etc/root file, you can see that I have a group down here named the RandD group. Both my rtracy and my ksanders users are both members of the RandD group. Essentially, we have an RandD team going in the organization, and my user and this other user are both users of that team, and hence are members of the RandD group. Both of these users are going to need read/write access to this file as we collaborate on this new project.

What we want to do is change the owner of this file to rtracy and change the owning group to the RandD group. We can do that with the 'chown' command. We'll specify rtracy as the owner and RandD as the group. The name of the file is 'widget_design_specs.odt'. Now we use the 'ls -l' command, and you can see the ownership has now changed. rtracy is the user that owns the file, and the RandD group is the owning group. These permissions are the ones that are assigned to the owning group. Users who are members of the RandD group will receive read permissions and all other authenticated users on the system will also receive read permissions.

Let's test this out. Go back into my editor, my word processing application. As my rtracy user, I will open the file again. Notice that the read-only text that was up here in the title bar before is now gone. I can make changes to this file without any problems. I can hit save. It does warn me that this file was created using a text editor, essentially, and so it just contains text. Do we want to convert it over to actual Open Office format? Yeah, we want to do that. We'll hit save to overwrite it, yes. Now we've converted this file essentially from a text file into a real word processing file. I was able to do that because the rtracy user has read and write permissions to that file.

However, we still have a problem. Remember, we said that the ksanders and the rtracy user are both members of the team that will be working on this project, and both users will need read/write access to these files. Well, we've got a problem because ksanders doesn't have read/write access. Remember, ksanders is a member of the RandD group. And RandD, as the owning group, only had read access to the file. We can't make ksanders an owner of the file too, because you can only have one user that owns a file. If we were to assign ksanders the file, then rtracy would no longer be the owner of the file and would lose access. That option won't work.

What we can do, however, is modify the mode to grant the write permission to the RandD group. That way, any user who is a member of RandD will receive the write permission. We can do that with the chmod command. We'll enter 'chmod' and then we'll specify 'g' for group plus 'add' and the permission that we want to add--'write'. Then we specify the name whose mode we want to modify. We should see in the mode that the permissions assigned to the owning group have changed. Members of the owning group have write permissions as well as read permissions. Essentially, any user who is a member of the RandD group has the exact permissions as the user that owns the file, which is rtracy. We've effectively granted ksanders the appropriate level of access to this file. Remember that we said over here that any other user who logs in to the system who is not rtracy and who is not a member of the RandD group will be assigned these permissions. If I were to log in to this system as a user named Fred or something like that, I would still be able to see this file because I have read permissions to that file.

Numerical Permission 8:28-10:38

What if this a confidential document? What if this is a secret project that we're working on that we don't necessarily want getting out? We don't want our competitors to find out about our new widget design; therefore, we want to constrain who can actually see this file. In this case, we probably want to lock things down, because basically anyone on this system can see this file. We probably don't want that. We need to remove this read permission that's assigned to other authenticated users. Let's do that with the 'chmod' command again. But this time we're going to use a different syntax with chmod. Previously, we just assigned the group the write permission using g plus w. We can do the same thing here to specify that the others lose the read permission, and we would use the minus sign instead of the plus sign to do that. However, you can also represent permissions numerically. Remember, read is assigned a value of 4, write a value of 2, and execute is a value of 1. Therefore, we want to preserve the read/write permissions assigned to the user. Read is 4, write is 2, so we would first specify a '6'.

Remember, the first number in the 'chmod' command refers to the permissions assigned to the user. We specify '6' to ensure that rtracy has read/write. Likewise, we want the group--the owning group--to have read and write, so we specify another '6'. That's four plus two. For others, which is the third digit in the 'chmod' command, we need to change things. If we don't want others to have any permissions at all, we specify '0', because by specifying 0, we specify no permissions at all be assigned to other authenticated users. Then we specify the name of the file to modify.

Let's use the 'ls' command to view the mode. Things are looking better. User has read/write, group still has read/write, but notice what happened to other authenticated users. No permissions at all. They will not be able to open this file; they will not be able to view it. Therefore, the security of this file has increased significantly. Basically, we've locked it down to the just the rtracy user and any user who is a member of the RandD group.

Summary 10:38-10:43

That's it for the demonstration. In this demo, we learned about using the chmod command at the Linux shell prompt to manage the permissions that are assigned to files.