|  |  |  |
| --- | --- | --- |
| Name: | Section/Schedule: | Rating: |
| Terminal No. : | Date: |

**Experiment No. 7: File Ownership and Permission Commands**

**Objectives:**

In this experiment, the students are expected:

* Define and explain the permissions and ownership assigned to files and directories
* Assign or set permission to files and directories using the read, write and execute permissions
* Assign or set permission to files and directories given a certain scenario or situation
* Modify ownership and group ownership of files and directories

***Note: Save this file as SURNAME1\_SURNAME2\_expt7.docx***

**DISCUSSION**

**FILE/DIRECTORY OWNERSHIP**

Every directories and files inside the Linux system is owned by a specific user and group. File permissions for each file and directory are defined separately for users, groups, and others (users other than the owner and the members of the group). The description of each type of owners is as follows:

|  |  |
| --- | --- |
| **Owner** | **Description** |
| User | The username of the person who owns the file. By default, the user who creates the file will become its owner. |
| Group | The group that owns or has access to the file. All users who belong into the group will have the same access permissions to the file. This is relevant in situations wherein several users would need access to a common directory or file. Instead of assigning each user to have access to a certain file or directory, a group can be created, and be given access. In this sense, whenever users will be given access to the file or directory, the user will just have to be added to the group that can access it. |
| Other | These are users other than the owner and the members of the group with access to the said file or directory. In other words, if you set a permission for the "other" category, it will affect everyone else by default. |

**FILE/DIRECTORY PERMISSION**

Owners of files and directories in a Linux system should be provided with permissions to define the level of accessibility the owners have for a certain file or directory. There are three types of access permissions on Linux: read, write, and execute. These permissions are defined separately for the file's owner, group and other users.

|  |  |
| --- | --- |
| **Permission** | **Description** |
| Read (r) | For a regular file, the read permission (r) means the file can be opened, viewed and read. For a directory, the read permission means you can display (thru the command *ls*) the contents of the directory. |
| Write (w) | For a regular file, the write permission (w) means you can modify the file (write new data to the file). For the case of a directory, the write permission means you can add, remove, and rename files inside the directory. |
| Execute (x) | For the case of a regular file, the execute permission (x) means you can execute the file as a program or a shell script. For a directory, the execute permission allows you to execute commands in the said directory (e.g. access files in the directory and enter it, with the cd command. However, note that although the execute bit lets you enter the directory, you're not allowed to list its contents, unless you also have the read permissions to that directory). |

**VIEWING FILE PERMISSION**

To view the access permissions of a file, the **ls -l** command. The said command allows the long directory listing to be displayed, as seen in the example below:

$ **ls -l**

drwxr-xr-x 3 user1 users 80 2005-09-20 21:37 dir

-rw-r----- 1 user1 users 8187 2005-09-19 13:35 file

-rwxr-xr-x 1 user1 users 10348 2005-07-17 20:31 otherfile

The output of the ls –l command (as seen above) is interpreted as follows:

* The first column, shows the file type and permissions.
* The second column shows the number of links (directory entries that refer to the file)
* The third column shows the owner of the file
* The fourth column shows the group the file belongs to.
* The other columns show the file's size in bytes, date and time of last modification, and the filename.

The first column is organized into four separate groups, as follows:

* The first group consists of only one character, and it shows the file's type.

For example, ‘d’ means a directory and ‘-‘ means a normal file.

The first character can be any of these:

|  |  |
| --- | --- |
| **d** | **directory** |
| **-** | **regular file** |
| **l** | **symbolic link** |
| **c** | **character device file** |
| **b** | **block device file** |

The next nine characters show the file's permissions, divided into three groups, each consisting of three characters. The first group of three characters shows the read, write, and execute permissions for user (the owner of the file). The next group shows the read, write, and execute permissions for the group owner of the file. Similarly, the last group of three characters shows the permissions for other users.

To summarize, the characters seen for each owner’s permission could be as follows:

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

Given the example, as seen below, the following interpretations can be made:

$ **ls -l**

drwxr-xr-x 3 user1 users 80 2005-09-20 21:37 dir

*Explanation: dir is a directory, owned by user1 with read, write, execute permissions, can be accessed by the members of the group users and by other users with read and execute permission*

-rw-r----- 1 user1 users 8187 2005-09-19 13:35 file

*Explanation: file is a regular file, owned by user1 with read and write permissions, can be accessed by the members of the group users with read only permission while other users won’t be able to access the file.*

**SETTING FILE PERMISSION – SYMBOLIC MODE**

File permission can be set using the **chmod** command. Both the root user and the file's or directory’s owner can set file permissions. **chmod** has two modes, *symbolic* and *numeric*.

chmod, in symbolic mode is used as follows:

*Wipe out all the permissions but add read permission for everybody:*

$ chmod u=r,g=r,o=r file1

After the executing command, the file's permissions would be -r--r--r--

*Referring to the output of the command executed above, add execute permissions for group:*

$ chmod g+x file1

After the executing command, the file's permissions would be -r--r-xr--

*Referring to the output of the command executed above*, *add both write and execute permissions for the file's* *owner. Note that more than one permission can be assign at the same time:*

$ chmod u+wx file1

After the executing command, the file permissions will be -rwxr-xr--

*Referring to the output of the command executed above, remove the execute permission from both the file's owner and group.*

$ chmod u-x,g-x file1

After the executing command, the permissions are -rw-r--r--

**SETTING FILE PERMISSIONS – NUMERIC MODE**

The other mode in which chmod can be used is the numeric mode. In the numeric mode, the file permissions aren't represented by characters. Instead, they are represented by a three-digit octal number.

**4** = read (r)  
**2** = write (w)  
**1** = execute (x)  
**0** = no permission (-)

To set the permission, add up the numbers accordingly. For example, the rwx permissions would be 4+2+1=7, rx would be 4+1=5, and rw would be 4+2=6. Since separate permissions are set for the user (owner), group, and others, a three-digit number representing the permissions of all these groups should be obtained.

Example:  
$ chmod 755 file1

This will change the file1’s permissions to -rwxr-xr-x. The owner will have full access to the file (that is, read, write, and execute permissions (7=4+2+1)), the group will have read and execute permissions (5=4+1), and the others will have the read and execute permissions as well.

$ chmod 640 file1

In this example, file1’s permissions will be set to -rw-r-----. The owner will have read and write permissions (6=4+2), the group will have read permissions only (4), and the others won’t have any access permissions (0).

The numeric mode may not be as straightforward as the symbolic mode, but with the numeric mode, you can more quickly and efficiently set the file permissions. This quick reference for setting file permissions in numeric mode might help:

|  |  |
| --- | --- |
| **Which number?** | |
| **0** | **---** |
| **1** | **--x** |
| **2** | **-w-** |
| **3** | **-wx** |
| **4** | **r--** |
| **5** | **r-x** |
| **6** | **rw-** |
| **7** | **rwx** |

**CHANGING THE OWNER OF A FILE**

File and directory ownership can be modified using the chown command. Take note, however, that the root user or the owner of the file can only do this task.

Set the file's owner:

$ chown *username* filename

After executing this command, the new owner of a file called filename will be the user *username*. The file's group owner will not change.

You can also set the file's group at the same time. If the user name is followed by a dot and a group name, the file's group will be changed as well.

$ chown *username.usergroup* filename

After executing this command, filename's new owner would be user *username* and the group *usergroup*.

You can set the owner of a directory exactly the same way you set the owner of a file:

$ chown *username* dirname

After executing this command, only the owner of the directory will change. The owner of the files inside of the directory won't change.

In order to set the ownership of a directory and all the files in that directory, you'll need the -R option:

$ chown -R *username* dirname

The option -R stands for recursive since this command will recursively change the ownership of directory and its contents. After issuing the example command, the user *username* will be the owner of the directory dirname, as well as the content in that directory.

**CHANGING THE GROUP OWNERSHIP OF A FILE**

In addition to chown, the chgrp command can also be used to change the group ownership of a file or a directory. You must either be the root user or the owner of the file in order to change the group ownership.

Example:

$ chgrp *usergroup* file1

After issuing this command, the file file1 will be owned by the group *usergroup*. Although the file's group has changed to *usergroup*, the file's owner will still be the same.

$ chgrp –R *usergroup* dir1

The option -R stands for recursive since this command will recursively change the group ownership of directory and its contents. After issuing the example command, the group usergroup will be the group owner of the directory dir1, as well as content of that directory..

* **Tasks / Procedure / Instructions:**

**Part I. Define and explain the permissions and ownership assigned to files and directories.**

**Use snipping tool to capture the output.**

* Type the command pwd. What is the output?

Paste your executed command and output below:

What is the meaning of pwd?

Answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* The output of the above command should display your home directory. In case you are NOT inside your home directory, type the command cd ~ to redirect you to your home directory.

Paste your executed command and output below:

* Once inside your home directory, create a directory and name it LabExer7.

Paste your executed command and output below:

* Go inside the LabExer7 directory. Inside this directory, create another directory and name it LabExer7Dir. Also, create an empty file and name it LabExer7File.

Paste your executed command and output below:

* After creating the file and directory as stated above, type the command ls –l. Paste the output of the said command on the space provided below (make sure that you are inside the LabExer7 directory)

Paste your executed command and output below:

* Explain in your own words the ownership and permission of directory LabExer7Dir and file LabExer7file. Refer to the explanation format as seen in the Introduction part.

Answer:

**Part II. Assign or set permission to files and directories using the read, write and execute permissions**

* Referring to the file created above (LabExer7File), set the permission of the said file as follows using symbolic and absolute mode:

User (owner) = read, write, execute

Group = read, write

Others = read only

Paste your executed commands and outputs below:

Symbolic mode:

Absolute mode:

After executing the command above, display the long directory listing of LabExer7. What is the details for LabExer7File?

Paste your executed command and output below:

* Referring to the directory created above (LabExer7Dir), set the permission of the said directory as follows:

User (owner) = read, write, execute

Group = read, write, execute

Others = read and execute

Paste your executed commands and outputs below:

Symbolic mode:

Absolute mode:

After executing the command above, display the long directory listing of LabExer7. What is the details for LabExer7Dir.

Paste your executed commands and outputs below:

**Part III. Assign or set permission to files and directories given a certain scenario or situation.**

* Create a file, and name it financials. Make sure that the said file can only be READ and MODIFIED by owner (student) and can only be READ by the members (localhost) of the group. Other users DON’T HAVE ACCESS to the said file. Write below the answers and series of commands to do the specified tasks:

Using cat command, create file ***financials*** with file content of 123:

Paste your executed commands and outputs below:

Who is the Owner and Group Owner of the file ***financials***:

Answer:

Command to set the permission to the owner, group and other users;

Paste your executed commands and outputs below:

Symbolic mode:

Absolute mode

After executing the command above, display the long directory listing of LabExer7. What is the details for financials.

Paste your executed commands and outputs below:

**Part IV. Modify ownership and group ownership of files and directories**

For this part, the tasks **NEED NOT** be executed from the Linux box. You will be asked to specify the commands to complete the task. Write your answer on the space provided after each task.

* Assuming that LabExer7File is currently owned by root and the group user1. What command will you issue to change the ownership of the said file to user1?

Answer:

* Assuming that financials is currently owned by user1 and the group user1. What single command will you use to change the owner, as well as the group ownership of financials to user2 and group user2?

Answer:

* Assuming that LabExer7File is currently owned by user1 and the group user1. What command will you issue to change the **group ownership** of the said file to user2?

Answer:

* Assuming that LabExer7Dir is currently owned by user1 and the group user1. What command will you issue to change the ownership and the group ownership of the said directory to *user2* and the group *group2*?

Answer:

* Assuming that LabExer7Dir is currently owned by user1 and the group user1. What command will you issue to change the ownership and the group ownership of the said directory and its content to *user2* and the group *group2* ?

Answer: