Operating System Lab 6

User Management

In Linux

Eng. Yazan ALShannik

Part 1: User Management

Objective:

The objective of this lab is to understand and manage Linux file-system permissions, including basic user, group, and other permissions, as well as special permissions like SUID, SGID, and the sticky bit. By learning how to set and modify these permissions using commands like chmod, chown, and chgrp, users will be able to effectively control access and security for files and directories in a multi-user environment.

What is File-System Permissions?

File permissions in Linux are essential for controlling access to files and directories. The Linux file permissions system is both simple and flexible, making it easy to understand and apply while still effectively handling most common permission scenarios.

Categories of Users:

- Owner (User): Every file in Linux is owned by a user, usually the one who created the file. This user has specific permissions that apply only to them.
- Group: Each file is also associated with a single group. By default, this group is usually the primary group of the user who created the file, but it can be changed as needed. Group permissions apply to all users who are members of this group.
- Others: This category includes all other users on the system who are neither the owner of the file nor members of the owning group. Permissions for 'others' apply to everyone else.

Permission Precedence:

Permissions are applied in a specific order of precedence. The most specific permissions override more general ones:

- User (Owner) permissions take precedence over group permissions.
- Group permissions take precedence over others.

For example, if a user has explicit permission to read a file, that permission overrides the group's permission to deny reading.

Permission Types and Their Effects:

- Read (r):
  - **Effect on Files:** The content of the file can be read.
  - **Effect on Directories:** The names of the files within the directory can be listed.
- Write (w):
  - **Effect on Files:** The content of the file can be modified.
  - **Effect on Directories:** New files can be created, and existing files can be deleted within the directory.
- Execute (x):
  - **Effect on Files:** The file can be executed as a program or script.
  - **Effect on Directories:** Users can access the directory and its files (if they know the file names and have the necessary permissions).

Deleting Files:

Any user with write (w) permission to a directory can delete files within it, regardless of the file's ownership or individual permissions.

This behavior can be restricted using a special permission called the sticky bit, which ensures only the file's owner can delete it.

Viewing File and Directory Permissions and Ownership

To see detailed information about file permissions and ownership, you can use the ls command with the –l option.

- Example:

ls –l test

Managing File System Permissions from the Command Line

In Linux, the chmod command is used to modify file and directory permissions. This command stands for "change mode," reflecting its function of altering the mode (permissions) of a file or directory. The chmod command can apply permission changes using two methods: symbolic and numeric.

1- Changing Permissions with the Symbolic Method
The symbolic method allows you to specify permission changes using letters that represent the user types and the permissions to be modified.

- Syntax:

chmod WhoWhatWhich file/directory

Who: Specifies the category of users to which the permission change applies. It can be one or more of the following:

u: User (owner of the file)
g: Group (group associated with the file)
o: Others (everyone else)
a: All (user, group, and others)

What: Indicates what to do with the specified permissions:

+: Add the specified permissions.
-: Remove the specified permissions.
=: Set the permissions exactly as specified, replacing any existing permissions.

Which: Specifies the permissions to be modified:

r: Read permission.
w: Write permission.
x: Execute permission.
X: Special execute permission. It only sets execute if the file is a directory or if it already has execute permission for user, group, or others.

- Examples:

➢ Adding Permissions:

- To add execute permission for the user (owner) on a file named example.txt→ chmod u+x example.txt
- To add execute Permission for Everyone on file2 →

➢ Removing Permissions:

- To remove write permission for the group on a file named example.txt → chmod g-w example.txt.
- To remove read and Write Permissions for Group and Others on file1→chmod go-rw file1

➢ Setting Exact Permissions:

To set read and execute permissions for all users on a file →chmod a=rx example.txt

2- Changing Permissions with the Numeric Method

The numeric method of using chmod represents file permissions with octal numbers. This method uses three (or sometimes four) digits to define permissions for the user (owner), group, and others. Each digit is a sum of numbers that correspond to specific permissions.

- Understanding the Numeric Values:

4: Read (r) permission
2: Write (w) permission
1: Execute (x) permission

To set permissions using the numeric method, you add these values together for each category (user, group, others):

- A value of 7 (4 + 2 + 1) means read, write, and execute permissions are granted.
- A value of 6 (4 + 2) means read and write permissions are granted.
- A value of 5 (4 + 1) means read and execute permissions are granted.
- A value of 0 means no permissions are granted.

- Example Calculation:

1- Consider the permission set -rwxr-x---, what is the numeric representation of these permissions

- User: rwx = 4 (read) + 2 (write) + 1 (execute) = 7.
- Group: r-x = 4 (read) + 0 (write) + 1 (execute) = 5.
- Others: --- = 0 (no permissions).

The numeric representation of these permissions is 750.

- User (6): rw- = 4 (read) + 2 (write) = 6
- Group (4): r-- = 4 (read) = 4
- Others (0): --- = 0 (no permissions)

2- If the numeric permission is **640**, what is the symbolic representation is -rw-r-----

- Examples:

Set Read and Write Permissions for User, Read for Group and Others → chmod 644 samplefile
Set Full Permissions for User, Read and Execute for Group, No Permission for Others →chmod 750 sampledir

Changing File and Directory User or Group Ownership

In Linux, every file and directory has an associated user (owner) and group ownership. By default, a newly created file is owned by the user who creates it, and its group ownership is set to the user's primary group. In most cases, the primary group is a private group that includes only the user. However, there are scenarios where changing the user or group ownership of a file is necessary, especially for collaboration and access control.

Changing Ownership with chown Command

chown (change owner) is the command used to change the user and/or group ownership of files and directories.

- Changing File User Ownership:

  - Only the root user can change the user that owns a file.

  - To grant ownership of a file named test_file to a user named student:

- Syntax:

chown student test_file

- Changing Directory Ownership Recursively:

  - The -R option with chown is used to recursively change the ownership of a directory and all its contents (files and subdirectories).

  - To grant ownership of test_dir and everything within it to student:

- Syntax:

chown -R student test_dir

- Changing Group Ownership:

To change the group ownership of a directory named test_dir to a group named admins, use the **:** before the group name

- Syntax:

chown :admins test_dir

- Changing Both User and Group Ownership:

  - To simultaneously change both the user and the group ownership of a directory, use the owner:group syntax.

  - To change the ownership of test_dir to the user visitor and the group guests:

- Syntax:

chown visitor:guests test_dir

- Changing Group Ownership with chgrp Command

  - The **chgrp (change group)** command is an alternative to chown specifically for changing group ownership. It functions similarly to chown but is limited to group changes and does not require the **(:)** before the group name.

  - To change the group ownership of a directory named test_dir to admins:

- Syntax:

chgrp admins test_dir

Best Wishes