MANAGING FILE AND DIRECTORY PERMISSIONS

Unit-III

SCS281: Linux and Shell Programming

Mapped Course Outcomes (CO): CO4

Er. Anu Arora, Assistant Professor, GNA University

MANAGING FILE AND DIRECTORY PERMISSIONS

In Linux, managing file and directory permissions is crucial for system security and ensuring users have appropriate access levels.

Permissions help control who can read, modify, or execute files and directories.

The three main commands for managing permissions and ownership in Linux are:

1. **chmod** - change file permissions
2. **chown** - change file owner and group
3. **chgrp** - change group ownership of files and directories

TYPES OF PERMISSIONS

- **Read (r):** Like regular files, this permission allows you to read the contents of the directory. However, that means that you can view the contents (or files) stored within the directory. This permission is required to have things like the ls command work.
- **Write (w):** As with regular files, this allows someone to modify the contents of the directory. When you are changing the contents of the directory, you are either adding files to the directory or removing files from the directory. As such, you must have write permission on a directory to move (mv) or remove (rm) files from it. You also need write permission to create new files (using touch or a file-redirect operator) or copy (cp) files into the directory.
- **Execute (x):** This permission is very different on directories compared to files. Essentially, you can think of it as providing access to the directory. Having execute permission on a directory authorizes you to look at extended information on files in the directory (using ls -l, for instance) but also allows you to change your working directory (using cd) or pass through this directory on your way to a subdirectory underneath.

MANAGING PERMISSIONS

Permissions are assigned to three classes of users:

- **User (u)** - the owner of the file.
- **Group (g)** - members of the file's group.
- **Others (o)** - all other users.

Permissions can be represented either symbolically or numerically.

Symbolic Mode: In symbolic mode, permissions are represented as:

- r - read
- w - write
- x - execute

CHMOD – CHANGE FILE AND DIRECTORY PERMISSIONS

The chmod command is used to change the permission of a file or directory.

Read (r) - permission to read a file or list a directory's contents.

Write (w) - permission to modify a file or directory.

Execute (x) - permission to execute a file or access a directory.

CHMOD-MANAGING PERMISSIONS

Each user type (user, group, others) has specific permissions.

The syntax is:

chmod [who][operator][permissions] file

- **who**: u (user), g (group), o (others), a (all)
- **operator**: + (add permission), - (remove permission), = (set exact permission)
- **permissions**: r, w, x

MANAGING PERMISSIONS- EXAMPLES

 # Add execute permission for the user

chmod u+x filename

 # Remove write permission for the group

chmod g-w filename

 # Set read and write permissions for all (user, group, others)

chmod a=rw filename

NUMERIC (OCTAL) MODE

When Linux file permissions are represented by numbers, it's called numeric mode.

In numeric mode, a three-digit value represents specific file permissions (for example, 744.) These are called octal values.

The first digit is for owner permissions, the second digit is for group permissions, and the third is for other users.

The syntax is:

chmod [mode] file

The mode is represented by three digits: one for the **user**, one for the **group**, and one for **others**.

NUMERIC (OCTAL) MODE

In numeric mode, each permission is represented by a digit:

- 4 - read
- 2 - write
- 1 - execute

Add these numbers to set combined permissions:

- 7 (4+2+1) - read, write, execute
- 6 (4+2) - read, write
- 5 (4+1) - read, execute
- 0 - no permissions

 # Set full permissions for the user, read and execute for the group, and read-only for others

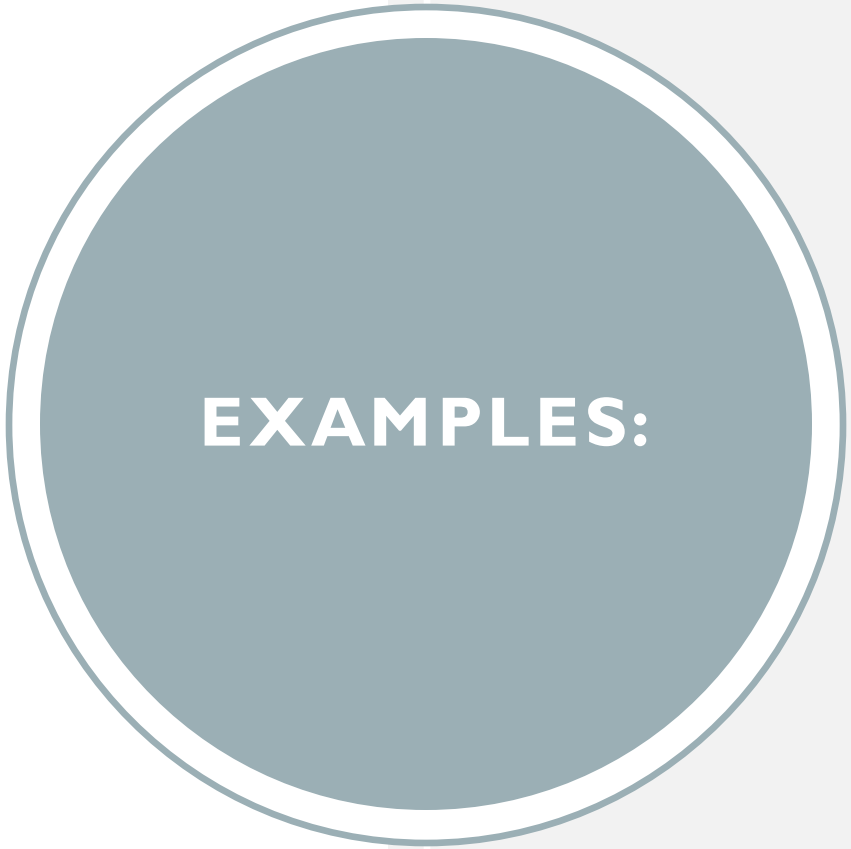
chmod 754 filename

 # Set read and write permissions for everyone

chmod 666 filename

 # No permissions for anyone

chmod 000 filename



CHOWN – CHANGE FILE OWNER AND GROUP

The chown command changes the ownership of a file or directory. Ownership is crucial because only the owner or a superuser can modify the file's permissions.

The syntax for chown is:

chown [OPTIONS] [user][:group] file

- **user**: The username or user ID of the new owner.
- **group**: The group name or group ID. If you specify only the user, the file's user ownership changes.
- If you specify only the group (using : or . before the group), only the group ownership changes.
- If you use both user: group, both user and group ownership are changed.

EXAMPLES

 # Change the owner of a file to 'username'

chown username filename

 # Change both the user and group ownership

chown username:groupname filename

 # Change only the group ownership

chown :groupname filename

COMMON OPTIONS FOR CHOWN

-R: Recursively change ownership for all files and directories within a directory.

Recursively change owner and group for a directory and its contents

chown -R username:groupname /path/to/directory

CHGRP – CHANGE GROUP OWNERSHIP

The chgrp command changes the group ownership of a file or directory. It is similar to chown but is specifically for group ownership.

The syntax is:

chgrp [OPTIONS] group file

group: The name of the new group.

Examples:

Change the group ownership of a file

chgrp groupname filename

COMMON OPTIONS FOR CHGRP

-R: Recursively change group ownership for all files and directories within a directory.

Recursively change group for a directory and its contents

chgrp -R groupname /path/to/directory

UNDERSTANDING PERMISSIONS EXAMPLE

Suppose we have a file with these permissions:

-rwxr-xr--

Breaking it down:

- **User (Owner)**: rwx – read, write, and execute.
- **Group**: r-x – read and execute only.
- **Others**: r-- – read-only.

To replicate these permissions using chmod:

chmod 754 filename

Command	Usage Example	Description
chmod	chmod 755 file	Change file permissions
chown	chown user:group file	Change file owner and group
chgrp	chgrp group file	Change group ownership
Recursive	chown -R user:group dir	Apply change recursively to directory

SUMMARY TABLE

SPECIAL FILE PERMISSIONS

Special permissions are available for files and directories and provide additional privileges over the standard permission sets.

- SUID is the special permission for the user access level and always executes as the user who owns the file, no matter who is passing the command.
- SGID allows a file to be executed as the group owner of the file; a file created in the directory has its group ownership set to the directory owner. This is helpful for directories used collaboratively among different members of a group because all members can access and execute new files.