1. **What is the significance of the setuid and setgid bits in file permissions?**

The setuid and setgid bits in file permissions allow executables and directories to run with elevated privileges.

- **setuid (Set User ID)**: When set on an **executable file**, it runs with the **file owner's** privileges.
    - Example: /usr/bin/passwd runs as **root** to modify system passwords.
    - Set: chmod u+s filename | Remove: chmod u-s filename
- **setgid (Set Group ID)**:
    - On an **executable file**, it runs with the **file group's** privileges.
    - On a **directory**, new files inherit the **directory's group** instead of the creator's default group.
    - Set: chmod g+s filename/directory | Remove: chmod g-s filename/directory

2. **How can you restrict file access to the file owner only?**

To restrict file access to the owner only:

Linux/macOS:

```
chmod 600 filename #Read & Write for owner only
```

```
chmod 700 filename #Read, Write & Execute for owner only
```

Windows (PowerShell):

```
icacls filename /inheritance:r /grant %username%:F
```

This removes all other permissions and grants full control to the owner.

3. **How do you remove write permissions for a group on a file using chmod?**

```
chmod g-w filename
```

Removes write permissions for the group.

4. **How can you check the number of hard links associated with a file?**

Use the following command to check the number of **hard links** to a file:

```
ls -l filename
```

The **second column** in the output shows the hard link count.

Alternatively, use:

```
stat filename
```

Look for **"Links"** in the output.

5. **What is the purpose of the sticky bit in directory permissions?**

The **sticky bit** (t) prevents users from deleting others' files in a shared **directory**.
Set it:
```
chmod +t directory_name
```
Check:
```
ls -ld directory_name
```
Seen as t (e.g., drwxrwxrwt).

6. **How do you recursively change the permissions of all files and subdirectories within a directory?**

Use the -R flag with chmod to **recursively** change permissions for all files and subdirectories:

```
chmod -R mode directory_name
```

Examples:

- Set **read, write, execute** for owner, and read for others:
- chmod -R 744 directory_name
- Remove **write** permission for **group** and **others**:
- chmod -R go-w directory_name

7. **How can you create a file and assign specific permissions at the same time?**

You can create a file and set specific permissions in one command using touch and chmod:

```
touch filename && chmod 640 filename
```

- touch filename → Creates the file.
- chmod 640 filename → Sets **read & write for owner, read for group, no access for others**.

Alternatively, use install:

```
install -m 640 /dev/null filename
```

This creates a file with the specified permissions in one step.

8. **What command is used to view the effective permissions of a symbolic link?**

To view the **effective permissions** of a **symbolic link**, use:

```
ls -l symlink_name
```

This shows the symlink itself and its target.

To check the **permissions of the target file**, use:

```
  ls -lL symlink_name
```

9. **How do you check and modify default permissions for newly created files in Linux?**

   Check Default Permissions:
   Use the umask command:
   umask
   This displays the current permission mask (e.g., 0022).
   Modify Default Permissions:
   Set a new umask value (e.g., 0077 for owner-only access):
   umask 0077
   This affects new files and directories created in the current session.
   To make it permanent, add the umask command to ~/.bashrc or ~/.profile.

10. **What is the command to change the ownership of a directory and all its contents?**

Use the chown command with the -R flag to recursively change ownership of a directory and all its contents:

chown -R new_owner:new_group directory_name

Examples:

- Change owner to user1 and group to group1:
- chown -R user1:group1 mydir
- Change only the owner:
- chown -R user1 mydir