Linux File Permissions and How to Change Them

In Linux, file permissions are crucial for system security and proper access control. Permissions define who can read, write, or execute a file or directory. Every file and directory has three types of owners and three types of permissions.

Types of Users:

1. **Owner (u):** The user who creates the file.
2. **Group (g):** Users belonging to the same group.
3. **Others (o):** Everyone else on the system.

Permission Types:

- **Read (r):** Ability to view file contents or list directory.
- **Write (w):** Ability to modify file or add/remove files in a directory.
- **Execute (x):** Ability to run a file (if script/binary) or enter a directory.

Viewing Permissions:

Use **ls -l** command:
Example: -rwxr-xr-- 1 user group 1234 file.sh
Breakdown:
- **-** → Regular file
- **rwx** → Owner has read, write, execute
- **r-x** → Group has read, execute
- **r--** → Others have read only

Changing Permissions with chmod:

Symbolic Mode:
- chmod u+x file.sh → Add execute for owner
- chmod g-w file.sh → Remove write for group
- chmod o=r file.sh → Set read-only for others

Numeric Mode: (r=4, w=2, x=1)
- chmod 755 file.sh → Owner: rwx (7), Group: r-x (5), Others: r-x (5)
- chmod 644 file.txt → Owner: rw- (6), Group: r-- (4), Others: r-- (4)

Changing Ownership:

chown: Change file owner.
- chown user1 file.txt
- chown user1:group1 file.txt

chgrp: Change group ownership.
- chgrp developers file.txt

Special Permissions:

SetUID (s): File runs with owner's privileges. Example: /usr/bin/passwd
SetGID (s): Files created in directory inherit group. Example: shared project dirs
Sticky Bit (t): In directories, users can delete only their own files. Example: /tmp

Permission Mapping Table:

Permission	Binary	Value
r--	100	4
-w-	010	2
--x	001	1
rw-	110	6
r-x	101	5
rwx	111	7

Best Practices:

- Use least privilege principle.
- For scripts: prefer 755 instead of 777.
- For config files: use 640 or 600 to protect sensitive data.
- Avoid giving unnecessary write permissions to group/others.