1. How do Linux file permissions (r, w, x) work for files vs directories? Give an example using ls -l.

For regular files:

i. r (read) → You can view the file's contents (cat, less).
ii. w (write) → You can modify or delete the file.
iii. x (execute) → You can run the file as a program/script.

Example:

ls -l file.sh

- → regular file, rwx (owner) → can read, write, execute, r-x (group) → can read and execute, but not write, r-- (others) → can only read.

For directories:

i. r (read) → You can list the names of files in the directory (ls).
ii. w (write) → You can create, delete, or rename files in the directory.
iii. x (execute/search) → You can enter the directory (cd) and access its contents (open files if you know the name).

Example:

ls -ld mydir

d → directory, rwx (owner) → can list, create/delete files, and enter, r-x (group) → can list and enter, but not create/delete, --- (others) → no access at all.

2. Explain octal notation for permissions and what the umask command does. Give one calculation example.

Linux file permissions are represented with 3 bits per category (user, group, others).

Each permission (r, w, x) has a numeric value:

- r = 4
- w = 2
- x = 1

Example:

-rwxr-xr--

chmod 754 file.txt

- Owner: rwx = 7
- Group: r-x = 5
- Others: r-- = 4

So in octal notation, this is: **754**.

The umask command in Linux is used to set default permissions for files or directories the user creates.

- **File →** The full permission set for a file is 666 (read/write permission for all)
- **Directory →** The full permission set for a directory is 777 (read/write/execute)
- When we make a new directory, the permissions will be calculated as (full permissions for directory) - (umask value) i.e. 777 - 543 = 234
- When we make a new file, the permission will be given out similarly but with a slight change as follows: (full permissions for file) - (umask value) i.e. 666-543 = 123

Example Calculation

If umask is 022:

1. **New File**
   - Max = 666
   - Subtract 022 → 644 (rw-r--r--)
2. **New Directory**
   - Max = 777
   - Subtract 022 → 755 (rwxr-xr-x)

3. What is the difference between the root user and a normal user? Why is root considered dangerous?

Root User

i. The superuser account in Linux (username: root).
ii. Has UID 0.
iii. Can do anything on the system:
- Read/write any file (even other users').
- Install/remove software.
- Kill any process.
- Change system settings.
- Format disks, edit /etc/passwd, etc.

Normal User

i. Created for daily use (e.g., ahmed, john).
ii. Each has its own UID > 0.
iii. Permissions are restricted:
- Can only access files they own (unless world-readable).
- Cannot bind to ports < 1024 without special privileges.
- Cannot directly install system-wide software.
- Cannot modify system-critical files.

Root is dangerous because:

1. **No safeguards**: One wrong command can break the system.
2. Example: rm -rf / (deletes everything).
3. **Security risks**: If malware gains root access, the attacker owns the whole system.
4. **Bypasses permissions**: Root can read private files of all users (loss of privacy).

5. **System stability**: Misconfigured root actions (e.g., editing /etc/fstab) can prevent booting.