File Permissions

File permissions are critical for system security and access control. This handout covers how Linux handles file permissions, numerical and symbolic methods to set permissions, and permissions specific to directories.

---

1. Why File Permissions?

File permissions control who can:

- **Read** (view contents)
- **Write** (modify or delete)
- **Execute** (run as a program or enter a directory)

Permissions are assigned separately for:

- **Owner**
- **Group**
- **Others** (everyone else)

---

2. Permission Types

Permission	Symbol	Numeric	Files	Directories
Read	`r`	`4`	View file contents	List directory contents
Write	`w`	`2`	Modify/delete file	Create/delete files in directory
Execute	`x`	`1`	Execute file as program	Enter/traverse directory

---

3. Viewing Permissions

Use `ls -al` to view permissions:

```
ls -al filename.txt
```

Example output:

```
-rwxr-xr-- 1 alice alice 4096 Mar 20 16:05 filename.txt
```

- First character: `-` for a file, `d` for directory
- Next three (`rwx`) → Owner permissions
- Next three (`r-x`) → Group permissions
- Last three (`r--`) → Others permissions

---

4. Setting Permissions (Symbolic)

Use `chmod` to change permissions symbolically.

Syntax:

```
chmod [u|g|o][+|-][r|w|x] filename
```

Examples:

- Add execute for owner:

```
chmod u+x script.sh
```

- Remove write permission for group:

```
chmod g-w file.txt
```

- Add read permission for others:

```
chmod o+r public.txt
```

---

5. Numerical Permissions (chmod):

Numerical permissions simplify setting permissions. Each digit (`0-7`) combines permissions.

- Read (`4`), Write (`2`), Execute (`1`)

Numeric Value	Permissions	Description
7	rwx	Read, Write, Execute
6	rw-	Read, Write
5	r-x	Read, Execute
4	r--	Read-only
3	-wx	Write, Execute
2	-w-	Write-only
1	--x	Execute-only
0	---	No permissions

Example (`chmod 754 file.txt`):

- Owner (`7`) → `rwx` (Read, Write, Execute)
- Group (`5`) → `r-x` (Read, Execute)
- Others (`4`) → `r--` (Read-only)

Usage:

```
chmod 754 filename.txt
```

---

6. Directory Permissions

Directories have special behaviors:

- **Read (`r`)** → List files in the directory
- **Write (`w`)** → Add/remove files (**requires execute** `x`)
- **Execute (`x`)** → Enter directory (`cd`) and access files directly

Examples:

- Give owner full permission, deny others:

```
chmod 700 private_folder
```

- Allow everyone to list files, but no write/execute:

```
chmod 444 public_folder
```

Important: To create/delete files, directory must have `write (w)` **and** `execute (x)` permissions set.

---

7. Changing Ownership (`chown`)

Change file owner/group.

Syntax:

```
chown user:group filename
```

Examples:

- Change owner only:

```
sudo chown alice file.txt
```

- Change owner and group:

```
sudo chown alice:developers file.txt
```

- Recursively change ownership (`R`):

```
sudo chown -R user:group directory
```

- It is **recursively applied to all files and subdirectories** inside a given directory.

---

8. Recursive Permissions

Apply permission changes to directories and all contents recursively.

- Change permissions recursively (`R`):

```
chmod -R 755 website_folder
```

🚩 Important Security Recommendations

- **Do NOT use `chmod 777` carelessly**

  It grants everyone full access, potentially dangerous.

- Prefer restrictive permissions:
  - `755` for directories (owner full, others read/execute).
  - `644` for files (owner read/write, others read-only).
- Regularly review permissions with `ls -al` .

📌 Summary of Key Commands

Command	Description
`ls -al filename`	View detailed permissions
`chmod u+x file`	Add execute permission to owner
`chmod g-w file`	Remove write permission from group
`chmod 755 directory`	Set numeric permissions (e.g., web folder)
`chmod -R 644 directory`	Recursively set numeric permissions for all files
`sudo chown user:group file`	Change file ownership
`sudo chown -R user:group directory`	Change ownership recursively