Permissions & Ownership (Core)

Linux Commands Course · Section 3

IDSchool

Why Permissions Matter

Permissions protect your system and data from unauthorized changes.

Each file and directory has:

- **Owner** – the user who owns it.
- **Group** – users sharing the same project/team.
- **Others** – everyone else.

Each of these can have **read**, **write**, or **execute** rights.

Viewing Permissions — ls -l

List files with detailed information:

```
ls -l
```

Example output:

```
-rwxr-xr-- 1 student staff  1024 Oct 22 10:30 script.sh
```

Parts of the first field (-rwxr-xr--):

Symbol	Meaning
-	file type (-=file, d=directory, l=symlink)
r	read permission
w	write permission
x	execute permission
-	permission missing

Grouped as: **user / group / others** → rwx r-x r--.

Basic Permission Categories

Entity	Description	Example
User (u)	The file owner	rwx
Group (g)	Members of the file’s group	r-x
Others (o)	Everyone else	r--

Example string breakdown:

```
-rwxr-xr--
 ↑   ↑   ↑
 u   g   o
```

Each group has three bits — total 9 permission bits.

Changing Permissions — chmod (symbolic)

Change permissions using symbolic notation.

Format:

```
chmod [who][operator][permission] file
```

Examples:

```
chmod u+x script.sh     # give user execute permission
chmod g-w file.txt      # remove group write permission
chmod o+r notes.txt     # allow others to read
chmod a-rwx test.log    # remove all access for everyone
```

Who: u (user), g (group), o (others), a (all)
Operators: + (add), - (remove), = (set exactly)

Changing Permissions — chmod (octal)

Each permission is represented by a **number**:

Permission	Binary	Value
read (r)	100	4
write (w)	010	2
execute (x)	001	1

Sum them per group → **rwx = 7**, **r-x = 5**, **r-- = 4**.

Example:

```
chmod 755 script.sh
```

Breakdown:

User	Group	Others	Mode
rwx	r-x	r-x	755

Common Octal Modes

Mode	Meaning	Use case
644	rw-r--r--	normal text files
600	rw------	private files
755	rwx-r-x-r-x	executable scripts, public dirs
700	rwx------	private scripts
777	rwx-rwx-rwx	full access (dangerous)

Avoid 777 unless in temporary environments.

Changing Ownership — chown

Set the file’s owner (and optionally group).

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

Change both owner and group:

```
sudo chown alice:developers project/
```

Recursively change everything inside a directory:

```
sudo chown -R alice:developers /var/www/
```

Only root or file owners can change ownership.

Changing Group Ownership — chgrp

Change only the group part of ownership.

```
sudo chgrp staff report.txt
```

Useful when multiple users share access via a group.

Default Permissions — umask

umask defines what permissions **new files** start with.

Display your current mask:

```
umask
```

Example output: 0022 → means remove write for *group* and *others*.

Base defaults:

- Files start as 666 (rw-rw-rw-)
- Directories start as 777 (rwxrwxrwx)

So, 666 - 022 = 644 → normal default for new files.

Special Permission Bits

In addition to basic read/write/execute, three **special bits** exist:

Bit	Applies to	Symbol	Purpose
setuid	Executable files	s in user field	Run as file’s owner
setgid	Executables / directories	s in group field	Run as file’s group; new files inherit group
sticky bit	Directories	t in others field	Only owner can delete own files

setuid Example

If a binary has setuid bit, it runs as its owner (often root).

```
ls -l /usr/bin/passwd
```

You’ll see:

```
-rwsr-xr-x 1 root root ...
```

The `s` in the user part means it runs with owner privileges.

---

setgid Example

For executables:

- setgid means they run with the group of the file.

For directories:

- Files created inside inherit the directory’s group.

```
chmod g+s shared_dir
```

This helps in group collaboration environments.

Sticky Bit Example

Used on shared directories like `/tmp` to prevent deletion by others.

```
ls -ld /tmp
```

Output:

```
drwxrwxrwt ...
```

The final `t` means sticky bit is set — only file owners can delete their own files.

Checking Special Bits (Numeric Form)

Special bits occupy a **fourth digit** before the usual 3-digit mode.

Bit	Octal	Combined example
setuid	4	4755
setgid	2	2755
sticky	1	1755

Example:

```
chmod 1777 /shared/tmp
```

makes it world-writable but protected by sticky bit.

Recap

- View: `ls -l`
- Change: `chmod` (symbolic or octal)
- Ownership: `chown`, `chgrp`
- Defaults: `umask`
- Special bits: `setuid`, `setgid`, `sticky`

Permissions define *who* can read, write, or execute — the backbone of Linux security.
