chmod Command Cheatsheet

A concise reference for the Linux/Unix chmod command, covering file permissions, symbolic and numeric modes, special permissions, and practical examples.

File Permissions & Basics

Understanding File Permissions

File permissions in Unix-like systems control who can read, write, or execute a file or directory. They are shown using `ls -l`.

Permissions are set for three categories:

1. **Owner (u):** The user who owns the file/directory.
2. **Group (g):** The group associated with the file/directory.
3. **Others (o):** Everyone else on the system.

The permissions themselves are:

- **r:** Read permission (view file contents, list directory contents).
- **w:** Write permission (modify file, add/remove files in a directory).
- **x:** Execute permission (run a script/binary, access directory contents).

Directory execute (`x`) permission is crucial. It allows users to traverse into the directory and access files/subdirectories within it, even if they can't list its contents (`r`) or modify it (`w`).

Permissions are displayed as a 10-character string:

`-rwxr-xr-x`

- 1st char: File type (`-`=file, `d`=directory, `l`=symlink, etc.)
- Next 3: Owner permissions (rwx)
- Next 3: Group permissions (r-x)
- Last 3: Others permissions (r-x)

`chmod` is the command used to change these permissions.

Symbolic Mode Syntax

Who	Target user(s): • `u`: owner • `g`: group • `o`: others • `a`: all (u, g, o)
Operator	Action: • `+`: Add permission • `-`: Remove permission • `=`: Set permission exactly
Permissions	Permission type: • `r`: Read • `w`: Write • `x`: Execute

Format	`who operator permissions file(s)`
	
	Combine multiple changes with commas:
	`u+rwx,g+rx,o+rx file.txt`

Examples	• `u+x`: Add execute for owner • `go-w`: Remove write for group and others • `a=rw`: Set read/write for all (removes execute if present) • `+r`: Add read for all (a is default if 'who' is omitted)
Recursive	Use `-R` to apply changes recursively to directories and their contents.

Symbolic & Numeric Modes

Symbolic Mode Examples

Make owner executable	`chmod u+x myscript.sh`
Remove write for group/others	`chmod go-w myfile.txt`
Set owner to rwx, group/others to r-x	`chmod u=rwx,go=rx mydir` (Equivalent to `chmod 755 mydir`)
Add read for all	`chmod +r shared_file.txt`
Remove execute from all	`chmod -x all_files`
Add read/write for group	`chmod g+rw config.conf`
Recursively make scripts executable for owner	`chmod -R u+x scripts/`
Set permissions to owner rwx, group r-, others -	`chmod u=rwx,g=r,o= mysecretfile` (Equivalent to `chmod 740 mysecretfile`)

Numeric Mode (Octal)

Permissions can be represented as a 3-digit octal number. Each digit corresponds to the permissions for owner, group, and others, respectively.

Each permission has a numeric value:

- `r` (Read) = 4
- `w` (Write) = 2
- `x` (Execute) = 1
- `-` (No permission) = 0

Sum the values for each permission type to get the digit for that category (Owner, Group, Others).

rwx	r–	rw-	r-x	-wx	–x	-w-	–r
7	4	6	5	3	1	2	4

Example: `rwxr-xr-x`

- Owner: `rwx` = 4 + 2 + 1 = 7
- Group: `r-x` = 4 + 0 + 1 = 5
- Others: `r-x` = 4 + 0 + 1 = 5

Numeric mode: `755`

The command format is:

`chmod NNN file(s)`

Where NNN is the 3-digit octal number.

Numeric mode is often quicker for setting exact permission sets.

Numeric Mode & Special Bits

Numeric Mode Examples

Owner rwx, Group rwx, Others rwx	`777`
	 `chmod 777 public_file.txt`
	
	 (Caution: Generally avoid unless necessary!)

Owner rwx, Group r-x, Others r-x	`755`
	 `chmod 755 my_directory`
	
	 (Common for directories)

Owner rw-, Group r--, Others r--	`644`
	 `chmod 644 report.pdf`
	
	 (Common for regular files)

Owner rwx, Group ---, Others ---	`700`
	 `chmod 700 private_script.sh`
	
	 (Only owner can read/write/execute)

Owner rw-, Group rw-, Others rw-	`666`
	 `chmod 666 temp_file`
	
	 (All can read/write, no execute. Caution advised.)

Owner r--, Group r--, Others r--	`444` `chmod 444 readonly_file.txt` (All can read, no write/execute)

Special Permissions (SetUID, SetGID, Sticky)

Special permissions add capabilities beyond basic rwx. They are represented by a leading digit in numeric mode (4-digit octal) or special characters in symbolic mode (`s` , `t`).

SetUID (SUID): (4000)

- **On files:** Runs the file with the permissions of the **owner**, not the user executing it. (e.g., `passwd` command)
- **In `ls -l`:** Shown as `s` in owner's execute position (`rwsr-xr-x`). If owner doesn't have `x` , it's `S` (`rwSr-xr-x`).
- **Numeric:** Add 4 to the leading digit (e.g., `chmod 4755 file` = SetUID + 755).

SetGID (SGID): (2000)

- **On files:** Runs the file with the permissions of the **group**, not the user executing it.
- **On directories:** Newly created files/subdirectories within this directory inherit the **group** of the directory, not the primary group of the user creating them. Useful for shared work areas.
- **In `ls -l`:** Shown as `s` in group's execute position (`rwxrwsr-x`). If group doesn't have `x` , it's `S` (`rwxrwsr-x`).
- **Numeric:** Add 2 to the leading digit (e.g., `chmod 2755 dir` = SetGID + 755).

Sticky Bit: (1000)

- **On directories:** Restricts file deletion or renaming within the directory to the file's owner, the directory's owner, or the root user, even if users have write permission. (e.g., `/tmp` directory)
- **On files:** (Historically used, less common now) Keep executable image in swap space.
- **In `ls -l`:** Shown as `t` in others' execute position (`rwxrwxrwt`). If others don't have `x` , it's `T` (`rwxrwxrwT`).
- **Numeric:** Add 1 to the leading digit (e.g., `chmod 1777 dir` = Sticky + 777).

Special permissions are often used for security or group collaboration. Use with caution, especially SetUID/SetGID on files.

Advanced Usage & Tips

Useful chmod Options

-R	Recursively change permissions of directories and their contents. `chmod -R 755 my_project/`
-v, --verbose	Output a diagnostic for every file processed. `chmod -v u+x script.sh`
-c, --changes	Like verbose but only report when a change is made. `chmod -c 644 report.txt`
-f, --silent, --quiet	Suppress most error messages.
--reference=RFILE	Use permissions from RFILE instead of specifying mode. `chmod --reference=template.txt new_file.txt`
a+X	Special mode: Grant execute (`x`) permission only if the file is a directory or if it already has execute permission for some user. `# Make directories executable, leave files as is (unless already executable)` `find . -exec chmod a+X {} \;`

Default Permissions (`umask`):

New files/directories get default permissions based on the system's `umask` setting. Check with `umask` command. It's an octal mask *removed* from a base permission (usually 666 for files, 777 for directories).

Example: `umask 022` means new files are `666-022=644`, directories are `777-022=755`.

Files vs. Directories:

- **Files:** Need read (`r`) to view content, write (`w`) to modify, execute (`x`) to run as a program/script.
- **Directories:** Need read (`r`) to list contents, write (`w`) to create/delete files, execute (`x`) to traverse into it (`cd`), access files within, or use `ls -l` inside.

Common Settings:

- Regular files: Often `644` (owner rw, group r, others r) or `600` (owner rw only).
- Executable scripts/binaries: Often `755` (owner rwx, group rx, others rx) or `700` (owner rwx only).
- Directories: Often `755` (owner rwx, group rx, others rx).

Be Cautious with `777` or `666`:

Granting write permission to 'others' (`o+w` or the last digit being 6 or 7) can pose a security risk, allowing any user to modify or delete files.

Use `find` for Complex Recursive Changes:

`find` can select files based on type (`-type f` for files, `-type d` for directories) and then apply `chmod` using `-exec`. This is useful for setting different permissions for files and directories recursively.

```
# Set 644 for files and 755 for directories recursively
find mydir -type f -exec chmod 644 {} \;
find mydir -type d -exec chmod 755 {} \;
```

Test First:

Especially when using `-R`, it's wise to test the command on a copy of the data or use `-v` or `-c` to see what changes are being made before applying it to critical data.

Documentation:

For full details and advanced options, consult the manual page:

```
man chmod
```