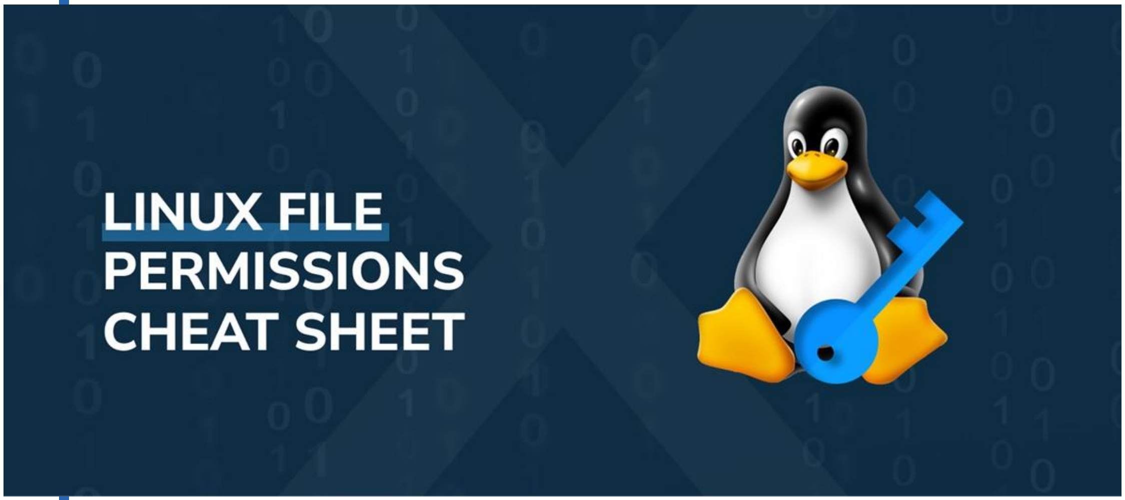
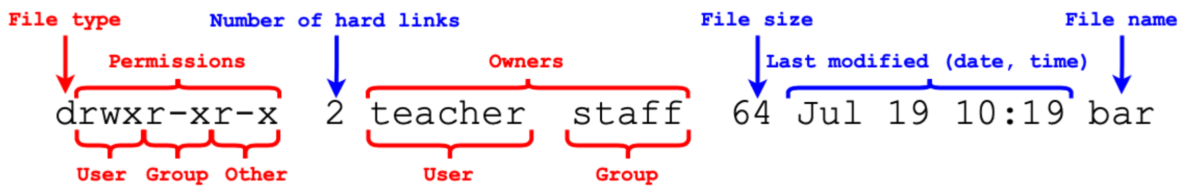
Linux File Permissions Cheat Sheet



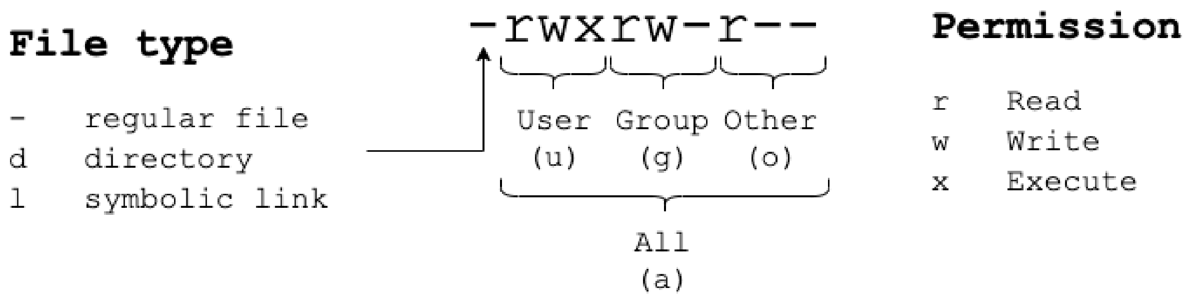
Permissions

The following commands display file/directory permissions:

Command	Description
`ls -l foo.sh`	Check permissions of file `foo.sh`
`ls -ld bar`	Check permissions of directory `bar`



Permissions, scope and file details upon executing `ls -l` or `ls -ld`



Permissions in symbolic notation

The permissions on files and directories span four scopes:

Scope	Symbol	Description
User	`u`	The owner of the file or directory
Group	`g`	The group of users to who can access the file or directory
Other	`o`	Other users (world)
All	`a`	All users

File Permissions

Permission type	Symbol	If a file has this permission, you can:	If a directory has this permission, you can:
Read	`r`	Open and view file contents (`cat, head, tail`)	Read directory contents (`ls, du`)
Write	`w`	Edit, delete or rename file (`vi`)	Edit, delete or rename directory and files within it; create files within it (`touch`)
Execute	`x`	Execute the file	Enter the directory (`cd`); without `x`, the directory's `r` and `w` permissions are useless
None	`-`	Do nothing	Do nothing

Permission-Related Commands

Command	Description
`chmod permission foo`	Change the permissions of a file or directory `foo` according to a `permission` in symbolic or octal notation format. Examples:
`chmod +x foo`	Grant execute permissions to all users to `foo` using symbolic notation.
`chmod 777 foo`	Grant read, write and execute permissions to all users to `foo` using octal notation.
`chown user2 foo`	Change the owner of `foo` to `user2`.
`chgrp group2 foo`	Change the group to which `foo` belongs to `group2`.
`umask`	Get a four-digit `subtrahend`. Recall in subtraction: `minuend - subtrahend = difference` If the `minuend` is 777, the `difference` is your default directory permissions; if it's 666, the `difference` is your default file permissions.
`su / sudo / sudo -i`	Invoke superuser privileges.
`id`	Find your user id and group id.
`groups`	Find all groups to which you belong.

If you run a command beyond the permissions granted, you get errors such as “Permission denied” or “Operation not permitted”.

Changing Permissions

There are two methods to represent permissions on the command line. The first argument of the `chmod` command admits both representations.

Method	Format of permission	Examples	Non-`chmod` application

Symbolic notation	A short text string consisting of one character of `[u/g/o/a]`, one of the assignment symbols `[+/-/=]` and at least one of `[r/w/x]`. If you omit `u/g/o/a`, the default is `a`.	`u+r`
		`g-wx`
		`o=rx`
		`+x` (i.e., `a+x`)
			`ls -l` and `ls -ld` command outputs, e.g. `-rwxrw-r--x`
			
			Here, `-` denotes the absence, not the removal, of a permission.

Octal notation	three-digit octal number ranging from 000 to 777	`774`
		`640`
			Computing default permissions with `umask`

Symbolic Notation

This notation is used in the `ls -l` and `ls -ld` command outputs, and it uses a combination of `u/g/o/a` (denoting the scope), `+/-/=`, and `r/w/x` to change permissions. If you omit `u/g/o/a`, the default is `a`.

The notation `+/-/=` refers to granting/removing/setting various permissions.

Here are some examples of `chmod` usage with symbolic notation. You may change more than one permission at a time, joining symbolic notations with a comma (,) as shown in the fourth example below.

Command in symbolic notation	Change in user (`u`) permissions	Change in group (`g`) permissions	Change in world (`o`) permissions
`chmod +x foo`	✓ Execute	✓ Execute	✓ Execute
`chmod a=x foo`	☐ Read ☐ Write ✓ Execute	☐ Read ☐ Write ✓ Execute	☐ Read ☐ Write ✓ Execute
`chmod u-w foo`	☐ Write	(No change)	(No change)
`chmod u+wx,g-x,o=rx foo`	✓ Write ✓ Execute	☐ Execute	✓ Read ☐ Write ✓ Execute

Octal Notation

This notation is a three-digit number, in which each digit represents permissions as the sum of four addends 4, 2, and 1 corresponding to the read (`r`), write (`w`) and execute (`x`) permissions respectively.

- The first digit applies to the user (owner) (`u`).
- The second digit applies to the group (`g`).
- The third digit applies to the world (other users) (`o`).

Octal digit	Permission(s) granted	Symbolic
`0`	None	`[u/g/o]-rwx`
`1`	Execute permission only	`[u/g/o]=x`
`2`	Write permission only	`[u/g/o]=w`
`3`	Write and execute permissions only: 2 + 1 = 3	`[u/g/o]=wx`
`4`	Read permission only	`[u/g/o]=r`
`5`	Read and execute permissions only: 4 + 1 = 5	`[u/g/o]=rx`
`6`	Read and write permissions only: 4 + 2 = 6	`[u/g/o]=rw`
`7`	All permissions: 4 + 2 + 1 = 7	`[u/g/o]=rwx`

Here are some examples of `chmod` usage with octal notation:

Command in octal notation	Change in user (`u`) permissions	Change in group (`g`) permissions	Change in world (`o`) permissions
`chmod 777 foo`	✓ Read ✓ Write ✓ Execute	✓ Read ✓ Write ✓ Execute	✓ Read ✓ Write ✓ Execute
`chmod 501 foo`	✓ Read ☐ Write ✓ Execute	☐ Read ☐ Write ☐ Execute	☐ Read ☐ Write ✓ Execute
`chmod 365 foo`	☐ Read ✓ Write ✓ Execute	✓ Read ✓ Write ☐ Execute	✓ Read ☐ Write ✓ Execute
`chmod 177 foo`	☐ Read ☐ Write ✓ Execute	✓ Read ✓ Write ✓ Execute	✓ Read ✓ Write ✓ Execute

Conversion Between Symbolic and Octal Notations

To visualize octal notation, let ↔ map symbolic notation to binary numbers (0 = permission denied, 1 = permission granted), and let ⇔ convert between the binary and octal numeric system. You have:

- `r` $\leftrightarrow 100_2 \Leftrightarrow 4_8$,
- `w` $\leftrightarrow 010_2 \Leftrightarrow 2_8$, and
- `x` $\leftrightarrow 001_2 \Leftrightarrow 1_8$.

Therefore, each combination of `r`, `w`, and `x` corresponds to the unique sum of their numerical representations, such as full `rwx` permissions ↔ $111\ 111\ 111_2 \Leftrightarrow 777_8$, as follows:

Symbolic notation (`ls -l`)	Binary representation	Octal notation
`rwxr-xr-x`	`111 101 101`	`755`
`rw-r--r--`	`110 100 100`	`644`
`rwx------`	`111 000 000`	`700`
`r-xr-xr-x`	`101 101 101`	`555`

Default Permissions

Apart from being an alternative to symbolic notation, octal notation has a special use case with the `umask` command.

To check what permissions you have as the current user, use the `umask` command to get a four-digit number which, if subtracted from 0777, gives your default permissions for creating a directory and, if subtracted from 0666, gives your default permissions for creating a file.

Usage:

Command	Description
umask	Find your default user and group permissions when you create a new file or directory

Examples:

umask output	Default directory permissions	Default file permissions
0002	Octal: `777 – 2 = 775` Symbolic: `rwxrwxr-x`	Octal: `666 – 2 = 664` Symbolic: `rw-rw-r--`
0022	Octal: `777 – 22 = 755` Symbolic: `rwxr-xr-x`	Octal: `666 – 22 = 644` Symbolic: `rw-r--r--`
0314	Octal: `777 – 314 = 463` Symbolic: `r--rw-wx`	Octal: `666 – 314 = 352` Symbolic: `-wxr-x-w-`

Changing Ownership

Before changing the ownership of any file or directory, you need to know how your computer identifies users and groups. Two useful commands are `id` and `groups`.

Usage:

Command	Description
`id`	Find your user id (`uid`) and your group id (`gid`)
`groups`	Find the group(s) your user belongs to

Example:

`id` output	Description
`uid=501(teacher) gid=20(staff) groups=20(staff),12(everyone),61(localaccounts)`	Your user id (`uid`) is 501. Your group id (`gid`) is 20.

	Your user belongs to three groups: `staff`, `everyone` and `localaccounts`.
`groups` output	**Description**
`staff everyone localaccounts`	Your user belongs to three groups: `staff`, `everyone` and `localaccounts`.

Superuser

Most Linux distributions contain a program which lets you access the terminal as the superuser (or root user). This program helps experienced users perform system administration tasks.

The two ways to invoke this program are the commands `su` (short for substitute user) to open up a dedicated root shell and `sudo` to execute commands appended to it inline. In both cases, you will need to enter the superuser's password to proceed with the task you intend to perform.

Modern distributions don't set the superuser password, so in that situation, use the `sudo -i` command to enter the root shell.

The shell symbol changes from `$` to `#` in the root shell. It is a reminder that with great power comes great responsibility. To quit the root shell, use the `exit` command.

Command (includes shell symbol)	Description of command	Output prompt and (new) shell symbol

`$ su`	Invoke superuser shell	`Password:`
		`#`

`$ sudo some_command`	Invoke superuser privilege in running `some_command`	`Password:`
		`$`

`$ sudo -i`	Invoke superuser shell if `su` is disabled	`Password:`
		`#`

Use these superuser commands with care.

Changing File Ownership

If you have superuser privileges, you may change the (user) owner of a file or directory by using the `chown` command. If you know the `uid` of the new owner, you may replace `user2` below with the corresponding `uid` as well.

Command	Description
`sudo chown user2 foo`	Transfer user ownership of `foo` to `user2`
`sudo chown 102 foo`	Transfer user ownership of `foo` to the user with `uid=102`

Changing Group Ownership

If you’re the owner of a file or directory, you may change the group ownership of a file or directory by using the `chgrp` command.

Command	Description
`chgrp group2 foo`	Transfer the ownership of file/directory `foo` to group `group2`
`chgrp 2 foo`	Transfer the ownership of file/directory `foo` to group with `gid=2`
`sudo chown user2:group2 foo`	(Superuser privileges required) Change the user and group ownership simultaneously to `user2` and `group2` respectively