chmod and chown File permissions

START HERE

`ls -l <file>`
Show mode and owner

`chmod u+w,go-w <file>`
Adjust by class

`chmod +x <file>`
Make executable

`chmod -R <mode> <dir>`
Apply recursively

`chmod 644 <file>`
Owner writes, others read

`chown <user>:<group> <file>`
Change owner and group

`chmod 755 <dir>`
Typical dir or script

`chown -R <user> <dir>`
Own a whole tree

Special bits setuid 4, setgid 2, sticky 1
Owner (u) first digit
Group (g) second digit
Others (o) third digit

CONCEPTS

Mode digits are sums per class: r=4, w=2, x=1. 640 means owner rw-, group r--, others ---.

Octal digit

Digit	Bits	Meaning
7	rwx	read, write, execute
6	rw-	read, write
5	r-x	read, execute
4	r--	read only
0	---	nothing

On directories, r lists names, w creates or deletes entries, x lets you enter and access items by name.

chown normally needs root. chmod works only for the file owner or root.

OCTAL MODES

`chmod 644 <file>`
rw for owner, r for rest

`chmod 755 <file>`
rwx for owner, rx for rest

`chmod 600 <file>`
Private to owner
ssh refuses private keys readable by others.

`chmod 700 <dir>`
Private directory

`chmod 664 <file>`
Group-writable file

`chmod 775 <dir>`
Group-writable directory

`chmod 00755 <dir>`
Clear setuid/setgid too
On directories, 755 and 0755 keep setgid/setuid; 5 digits or = clears them. Files: 0755 clears.

SYMBOLIC MODES

`chmod +x <file>`
Add execute (per umask)
Without a class, + honours umask bits; use a+x to force all.

`chmod a+x <file>`
Add execute for everyone

`chmod u+x <file>`
Add execute for owner

`chmod g+w <file>`
Let group write

SYMBOLIC MODES (cont.)

`chmod o-rwx <file>`
Remove all access for others

`chmod u=rw,go=r <file>`
Set classes exactly

`chmod g=u <file>`
Copy owner perms to group

`chmod u+x,g-w <file>`
Several changes, comma-separated

`chmod a-x <file>`
Remove execute for all

RECURSIVE CHANGES

`chmod -R <mode> <dir>`
Recurse into a directory

`chmod -R a+rX <dir>`
Read all, x only on dirs
Capital X adds execute only to directories and files that already have an execute bit.

`chmod -R u=rwX,go=rX <dir>`
Sane tree permissions

`find <dir> -type d -exec chmod 755 {} +`
Dirs only

`find <dir> -type f -exec chmod 644 {} +`
Files only

`chmod -R -P <mode> <dir>`
Skip all symlinks, even top one
Default is -H: a symlink given on the command line is followed.

`chmod -L -R <mode> <dir>`
Follow symlinks to directories
chmod never changes a symlink itself on Linux; it acts on the target.

`chmod --preserve-root -R <mode> /`
Refuse to recurse on /

SPECIAL BITS

`chmod u+s <file>`
Setuid: run as file owner

`chmod g+s <dir>`
Setgid dir: new files keep group

`chmod +t <dir>`
Sticky: only owners may delete

`chmod 4755 <file>`
Setuid via octal

`chmod 2775 <dir>`
Setgid shared directory

SPECIAL BITS (cont.)

`chmod 1777 <dir>`
Sticky world-writable (like /tmp)

`chmod u-s,g-s <file>`
Clear setuid and setgid

Setuid is ignored on shell scripts by Linux. Setgid on a directory makes new entries inherit its group.

ls -l symbols

Shown	Meaning
`s in owner x slot`	setuid with execute
`S`	setuid without execute
`s in group x slot`	setgid with execute
`S in group x slot`	setgid without execute
`t`	sticky with other-execute
`T`	sticky without other-execute

READING AND COPYING MODES

`ls -l <path>`
Long listing with mode

`ls -ld <dir>`
Mode of the dir itself

`stat -c '%a %U:%G %n' <file>`
Octal mode and owner

`chmod --reference=<ref> <file>`
Copy mode from another file

`chmod -c <mode> <file>`
Report only real changes

`chmod -v <mode> <file>`
Report every file

`chmod -f <mode> <file>`
Suppress error messages

CHOWN

`chown <user> <file>`
Change owner

`chown <user>:<group> <file>`
Change owner and group

`chown :<group> <file>`
Change group only

`chown <user>: <file>`
Owner and their login group

`chown -R <user>:<group> <dir>`
Recursive ownership

CHOWN (cont.)

`chown --reference=<ref> <file>`
Copy owner from another file

`chown --from=<user> <new> <file>`
Only if current owner matches

`chown -h <user> <link>`
Change the symlink itself
By default chown follows the symlink and changes its target.

`chgrp <group> <file>`
Change group (no chown needed)

`sudo chown -R $USER: <dir>`
Reclaim files made by root

UMASK

`umask`
Show current mask

`umask -S`
Show mask symbolically

`umask 022`
New files 644, dirs 755

`umask 077`
New files private to you

`umask u=rwx,g=rx,o=`
Set mask symbolically

umask removes bits from 666 for files and 777 for dirs. It only affects files created later by that shell and its children.

RECIPES

Web content tree

```
find /srv/www -type d -exec
chmod 755 {} +
find /srv/www -type f -exec
chmod 644 {} +
chown -R www-data:www-data /srv/
www
```

Shared team directory

```
sudo chgrp -R team /srv/shared
sudo chmod -R g+rwX /srv/shared
sudo chmod g+s /srv/shared
# new files now inherit group
team
```

Lock down SSH keys

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/id_ed25519
~/.ssh/authorized_keys
chmod 644 ~/.ssh/id_ed25519.pub
```

Fix files left by sudo or Docker

```
ls -l <dir> | head
sudo chown -R "$(id -un):$(id -
gn)" <dir>
```