Table of Contents

$ chgrp

Beginner

Change group ownership of files

chgrp changes the group ownership of files and directories. It is equivalent to the group-only part of chown (:group syntax), but regular users can use chgrp to change group to any group they belong to.

chgrp is commonly used for shared project directories where multiple users need access throug...

Options & Flags

`-R`	Recursive
`-v`	Verbose
`--reference`	Copy group from another file
`-f`	Suppress error messages

Practical Examples

Example: Change group

```
$ chgrp developers project/
```

Changes the group of the project directory.

Example: Recursive group change

```
$ sudo chgrp -R www-data /var/www/html/
```

Changes group for all files in web directory.

Example: Shared directory setup

```
$ sudo chgrp -R devteam /shared/ && sudo chmod -R g+rws /shared/
```

Sets up a shared directory: group ownership and setgid for new files.

Example: Copy group

```
$ chgrp --reference=index.php upload.php
```

Copies the group ownership from one file to another.

Example: Verbose output

```
$ chgrp -Rv developers src/
changed group of 'src/main.py' to developers
```

Shows each file as its group is changed.

Tips & Best Practices

Pro Tip: Regular users can use chgrp: Unlike chown, regular users can chgrp to any group they belong to (check with groups command).

Note: setgid for shared directories: After chgrp, set setgid: chmod g+s dir/. New files automatically inherit the directory group.

Warning: chown does both: chown :groupname file does the same thing. chgrp is mainly useful when non-root users need to change groups.

$ chmod

Beginner

Change file and directory permissions

chmod (change mode) modifies file and directory permissions. It controls who can read, write, and execute files using either symbolic notation (u+x, g-w) or numeric/octal notation (755, 644).

Linux permissions are organized in three levels: owner (u), group (g), and others (o). Each level has th...

Options & Flags

`+x`	Add execute permission
`-R`	Recursive (apply to all files in directory)
`u+x`	Add execute for owner only
`g+w`	Add write for group
`o-rwx`	Remove all permissions for others
`a+r`	Add read for all (owner, group, others)
`--reference`	Copy permissions from another file

Practical Examples

Example: Make script executable

```
$ chmod +x deploy.sh
```

Adds execute permission for all users.

Example: Standard file permissions

```
$ chmod 644 config.php
```

Owner: read-write (6). Group: read (4). Others: read (4).

Example: Standard directory permissions

```
$ chmod 755 /var/www/html/
```

Owner: full (7). Group: read-execute (5). Others: read-execute (5).

Example: Private file

```
$ chmod 600 ~/.ssh/id_rsa
```

Only owner can read and write. No access for anyone else. Required for SSH keys.

Example: Recursive web directory

```
$ find /var/www -type d -exec chmod 755 {} \; && find /var/www -type f -exec chmod 644 {} \;
```

Sets directories to 755 and files to 644 - standard web server permissions.

Tips & Best Practices

Pro Tip: Common permission numbers: 644 = files (rw-r--r--). 755 = directories/scripts (rwxr-xr-x). 600 = private files (rw-------). 700 = private directories.

Warning: Never chmod 777: chmod 777 gives everyone full access including write. This is a security risk. Use specific permissions instead.

Note: setuid, setgid, sticky bit: Special bits: setuid (4xxx) runs as file owner. setgid (2xxx) inherits group. Sticky bit (1xxx) prevents deletion by non-owners (/tmp).

$ chown

Beginner

Change file owner and group

chown (change owner) changes the user and/or group ownership of files and directories. File ownership determines which permission set (user, group, other) applies to each user accessing the file.

chown requires root privileges (or ownership of the file on some systems). It can change the owner, ...

Options & Flags

`user:group`	Change both owner and group
`user`	Change owner only
`:group`	Change group only
`-R`	Recursive
`--reference`	Copy ownership from another file
`-v`	Verbose - show changes

Practical Examples

Example: Change owner and group

```
$ sudo chown www-data:www-data /var/www/html/
```

Sets both owner and group to www-data (web server).

Example: Recursive ownership

```
$ sudo chown -R deploy:deploy /opt/myapp/
```

Changes ownership of entire application directory.

Example: Change group only

```
$ sudo chown :developers /shared/project/
```

Changes only the group, leaving owner unchanged.

Example: Fix home directory

```
$ sudo chown -R user:user /home/user/
```

Fixes ownership after accidentally changing home directory permissions.

Example: Web deployment

```
$ sudo chown -R www-data:www-data /var/www/html/ && sudo find /var/www/html/ -type d -exec chmod 755 {} \; && sudo find /var/www/h
```

Complete web deployment: set ownership and permissions.

Tips & Best Practices

Warning: Requires root: Only root can change file ownership. Use sudo chown. Regular users can only change the group to a group they belong to.

Pro Tip: Use user:group syntax: chown user:group file changes both at once. This is more efficient than separate chown and chgrp commands.

Note: Numeric IDs: chown 1000:33 file works with numeric UIDs and GIDs. Useful when username does not exist on the current system.

$ getfacl

Advanced

Get file access control lists

getfacl displays Access Control Lists (ACLs) for files and directories. ACLs provide fine-grained permissions beyond the basic owner/group/other model, allowing permissions for specific users and groups.

getfacl shows both the standard permissions and any extended ACL entries. ACLs are useful wh...

Options & Flags

`-R`	Recursive
`-t`	Tabular format
`-p`	No path stripping
`-d`	Show default ACLs only

Practical Examples

Example: View file ACLs

```
$ getfacl document.txt
# file: document.txt
# owner: admin
# group: staff
user::rw-
```

Shows all permissions including ACL entries.

Example: View directory ACLs

```
$ getfacl /shared/project/
```

Shows directory permissions and default ACLs for new files.

Example: Default ACLs

```
$ getfacl -d /shared/
```

Shows only the default ACL entries that new files will inherit.

Example: Recursive ACL listing

```
$ getfacl -R /var/www/ > acl_backup.txt
```

Saves all ACLs recursively for backup.

Example: Check for ACLs

```
$ ls -la | grep "+"
-rw-rw-r--+ 1 admin staff 1234 file.txt
```

The + sign in ls -l output indicates ACLs are set.

Tips & Best Practices

Pro Tip: + in ls output: ls -l shows + at the end of permissions (drwxrwxr-x+) when ACLs are set. Use getfacl to see the details.

Note: Backup and restore ACLs: getfacl -R /dir > backup.acl saves all ACLs. setfacl --restore=backup.acl restores them.

Warning: ACLs and cp/rsync: Regular cp does not preserve ACLs. Use cp -a or rsync -A to preserve ACL entries during copy.

$ setfacl

Advanced

Set file access control lists for fine-grained permissions

setfacl sets Access Control Lists (ACLs) on files and directories. ACLs provide fine-grained permissions beyond the basic owner/group/other model, allowing specific permissions for individual users and groups.

setfacl is invaluable when you need to grant access to specific users without changing...

Options & Flags

`-m`	Modify (set) ACL entry
`-x`	Remove ACL entry
`-b`	Remove all ACLs
`-d`	Set default ACL (for directories)
`-R`	Recursive
`--restore`	Restore ACLs from backup

Practical Examples

Example: Grant user access

```
$ setfacl -m u:alice:rw document.txt
```

Gives alice read-write access to the file, regardless of group membership.

Example: Grant group access

```
$ setfacl -m g:contractors:rx /opt/app/
```

Gives the contractors group read-execute access.

Example: Set default ACL

```
$ setfacl -dm u:alice:rwx /shared/project/
```

New files in this directory will automatically inherit alice rwx access.

Example: Remove user ACL

```
$ setfacl -x u:alice file.txt
```

Removes the ACL entry for alice.

Example: Remove all ACLs

```
$ setfacl -b file.txt
```

Strips all ACL entries, reverting to standard permissions only.

Tips & Best Practices

Pro Tip: Default ACLs for inheritance: setfacl -dm sets default ACLs on directories. New files automatically inherit these permissions. Essential for shared directories.

Warning: Mask limits effective permissions: The ACL mask limits the maximum permissions for named users and groups. Check with getfacl if permissions seem wrong.

Note: ACL syntax: u:user:perms for user ACL. g:group:perms for group ACL. o::perms for others. d: prefix for default ACLs.

$ umask

Intermediate

Set default file creation permissions

umask sets the default permission mask for newly created files and directories. It determines which permissions are NOT set when creating new files. The umask is subtracted from the maximum permissions.

Default maximum is 666 for files (no execute) and 777 for directories. With umask 022: new fi...

Options & Flags

`(no args)`	Show current umask
`NNN`	Set numeric umask
`-S`	Show symbolic representation
`-p`	Show in re-usable format

Practical Examples

Example: Show current umask

```
$ umask
0022
```

Displays the current permission mask.

Example: Show symbolic

```
$ umask -S
u=rwx,g=rx,o=rx
```

Shows umask as symbolic permissions for clarity.

Example: Set restrictive umask

```
$ umask 077
```

New files: 600 (owner only). New directories: 700. Very private.

Example: Set shared umask

```
$ umask 002
```

New files: 664. New directories: 775. Group can write.

Example: Make permanent

```
$ echo "umask 027" >> ~/.bashrc
```

Sets umask 027 permanently: files 640, directories 750.

Tips & Best Practices

Pro Tip: Common umasks: 022 = default (files 644, dirs 755). 077 = private (files 600, dirs 700). 002 = shared (files 664, dirs 775).

Note: umask is subtracted: umask removes permissions. umask 022: from 666 (files) removes group-write and other-write = 644.

Warning: Does not change existing files: umask only affects NEW files. Use chmod to change permissions on existing files.