

ESSENTIAL COMMANDS

Covers
Fedora Linux

LINUX

POCKET GUIDE



O'REILLY®

DANIEL J. BARRETT

(users), device type (30a), inode number (99492), number of hard links (1), and timestamps of the file's most recent access, modification, and status change. Filesystem information looks like:

```
$ stat -f myfile
File: "myfile"
  ID: bffff358 ffffffff Namelen: 255      Type: EXT2
Blocks: Total: 2016068   Free: 876122   Available:
773709   Size: 4096
Inodes: Total: 1026144   Free: 912372
```

and includes the filename (*myfile*), filesystem ID (bffff358 ffffffff), maximum length of a filename for that filesystem (255 bytes), filesystem type (EXT2), the counts of total, free, and available blocks in the filesystem (2016068, 876122, and 773709, respectively), block size for the filesystem (4096), and the counts of total and free inodes (1026144 and 912372, respectively).

The `-t` option presents the same data but on a single line, without headings. This is handy for processing by shell scripts or other programs.

```
$ stat -t myfile
myfile 1264 8 81a4 500 500 30a 99492 1 44 1e 1062130572
1059016181 1059016308
$ stat -tf myfile
myfile bffff358 ffffffff 255 ef53 2016068 875984 773571
4096 1026144 912372
```

Useful options

- `-l` Follow symbolic links and report on the file they point to.
- `-f` Report on the filesystem containing the file, not the file itself.
- `-t` Terse mode: print information on a single line.

wc [*options*] [*files*]

coreutils

/usr/bin *stdin* *stdout* *-file* *--opt* *--help* *--version*

The `wc` (word count) program prints a count of bytes, words, and lines in (presumably) a text file.

```
$ wc myfile
 24      62     428 myfile
```

This file has 24 lines, 62 whitespace-delimited words, and 428 bytes.

Useful options

- l Print the line count only.
- w Print the word count only.
- c Print the byte (character) count only.
- L Locate the longest line in each file and print its length in bytes.

du [options] [files|directories]

coreutils

/usr/bin stdin stdout -file --opt --help --version

The `du` (disk usage) command measures the disk space occupied by files or directories. By default, it measures the current directory and all its subdirectories, printing totals in blocks for each, with a grand total at the bottom.

```
$ du
8 ./Notes
36 ./Mail
340 ./Files/mine
40 ./Files/bob
416 ./Files
216 ./PC
2404 .
```

However, it can also measure the size of files:

```
$ du myfile myfile2
4 ./myfile
16 ./myfile2
```

Useful options

- b -k -m Measure usage in bytes (-b), kilobytes (-k), or megabytes (-m).
- B N Display sizes in blocks that you define, where 1 block = N bytes. (Default = 1024)
- h -H Print “human readable” output, and choose the most appropriate unit for each size. For example, if two directories are of size 1 gigabyte or 25 kilobytes, respectively, `du -h` prints 1G and 25K. The `-h` option uses powers of 1024 whereas `-H` uses powers of 1000.
- c Print a total in the last line. This is the default behavior when measuring a directory, but for measuring individual files, provide `-c` if you want a total.
- L Following symbolic links and measure the files they point to.

-s Print only the total size.

file [options] files

file

/usr/bin stdin stdout -file --opt --help --version

The file command reports the type of a file:

```
$ file /etc/hosts /usr/bin/who letter.doc
/etc/hosts:      ASCII text
/usr/bin/who:    ELF 32-bit LSB executable, Intel 80386 ...
letter.doc:      Microsoft Office Document
```

Unlike some other operating systems, Linux does not keep track of file types, so the output is an educated guess based on the file content and other factors.

Useful options

- b Omit filenames (left column of output).
- i Print MIME types for the file, such as “text/plain” or “audio/mpeg”, instead of the usual output.
- f *name_file* Read filenames, one per line, from the given *name_file* (and report their types), and afterward process filenames on the command line as usual.
- L Follow symbolic links, reporting the type of the destination file instead of the link.
- z If a file is compressed (see “File Compression and Packaging” on page 82), examine the uncompressed contents to decide the file type, instead of reporting “compressed data.”

touch [options] files

coreutils

/bin stdin stdout -file --opt --help --version

The touch command changes two timestamps associated with a file: its modification time (when the file’s data was last changed) and its access time (when the file was last read).

```
$ touch myfile
```

You can set these timestamps to arbitrary values, e.g.:

```
$ touch -d "November 18 1975" myfile
```

If a given file doesn’t exist, touch creates it, a handy way to create empty files.

Useful options

- a Change the access time only.
- m Change the modification time only.
- c If the file doesn't exist, don't create it (normally, touch creates it).
- d *timestamp* Set the file's timestamp(s). A tremendous number of timestamp formats are acceptable, from "12/28/2001 3pm" to "28-May" (the current year is assumed, and a time of midnight) to "next tuesday 13:59" to "0" (midnight today). Experiment and check your work with stat. Full documentation is available from info touch.
- t *timestamp* A less intelligent way to set the file's *timestamp*, using the format `[[CC]YY]MMDDhhmm[.ss]`, where *CC* is the two-digit century, *YY* is the two-digit year, *MM* is the 2-digit month, *DD* is the two-digit day, *hh* is the two-digit hour, *mm* is the two-digit minute, and *ss* is the two-digit second. For example, `-t 20030812150047` represents August 12, 2003, at 15:00:47.

chown [*options*] *user_spec files*

coreutils

`/bin` `stdin` `stdout` `-file` `--opt` `--help` `--version`

The `chown` (change owner) command sets the ownership of files and directories.

```
$ chown smith myfile myfile2 mydir
```

The *user_spec* parameter may be any of these possibilities:

- A username (or numeric user ID), to set the owner
- A username (or numeric user ID), optionally followed by a colon and a group name (or numeric group ID), to set the owner and group
- A username (or numeric user ID) followed by a colon, to set the owner *and* to set the group to the invoking user's login group
- A group name (or numeric group ID) preceded by a colon, to set the group only
- `--reference=file` to set the same owner and group as another given file

Useful options

- `--dereference` Follow symbolic links and operate on the files they point to.

-R Recursively change the ownership within a directory hierarchy.

chgrp [*options*] *group_spec files* coreutils

/bin *stdin* *stdout* -file --opt --help --version

The `chgrp` (change group) command sets the group ownership of files and directories.

```
$ chgrp smith myfile myfile2 mydir
```

The *group_spec* parameter may be any of these possibilities:

- A group name or numeric group ID
- `--reference=file`, to set the same group ownership as another given file

See “Working with Groups” on page 119 for more information on groups.

Useful options

`--dereference` Follow symbolic links and operate on the files they point to.

-R Recursively change the ownership within a directory hierarchy.

chmod [*options*] *permissions files* coreutils

/bin *stdin* *stdout* -file --opt --help --version

The `chmod` (change mode) command sets access permissions for files and directories. Not every file should be available to everyone (this isn’t Windows 95, y’know), and `chmod` is the tool for ensuring this. Typical permissions are read, write, and execute, and they may be limited to the file owner, the file’s group owner, and/or other users. The permissions argument can take three different forms:

- `--reference=file`, to set the same permissions as another given file
- An octal number, up to four digits long, that specifies the file’s *absolute* permissions in bits. The rightmost digit is special (described later) and the second, third, and fourth represent the file’s owner, the file’s group, and all users. See Figure 3 for an example, displaying the meaning of mode 0640.

- One or more strings specifying *absolute or relative* permissions (i.e., relative to the file's existing permissions) to be applied, separated by commas.

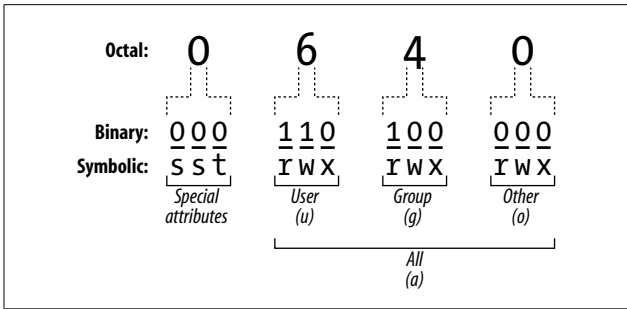


Figure 3. File permission bits explained

In the third form, each string consists of three parts: an optional *scope*, a *command*, and *permissions*.

Scope (optional)

u for user, g for group, o for other users not in the group, a for all users. The default is a.

Command

+ to add permissions, - to remove permissions, = to set absolute permissions, ignoring existing ones

Permissions

r for read, w for write/modify, x for execute (for directories, this is permission to cd into the directory), X for conditional execute (explained later), u to duplicate the user permissions, g to duplicate the group permissions, o to duplicate the “other users” permissions, s for setuid or setgid, and t for the sticky bit.

For example, ug+rw would add read and write permission for the user and the group, a-x (or just -x) would remove execute permission for everyone, and u=r would first remove all existing permissions and then make the file readable only by its owner. You can combine these strings by separating them with commas, such as ug+rw,a-x.

Attribute	Meaning
j	Journaled data (ext3 filesystems only).
s	Secure deletion: if deleted, this file's data is overwritten with zeroes.
S	Synchronous update: changes are written to disk immediately, as if you had typed <code>sync</code> after saving (see "Disks and Filesystems" on page 91).
u	Undeleteable: file cannot be deleted (undeleteable).

Useful options

-R Recursively process directories.

lsattr [options] [files]

e2fsprogs

/usr/bin *stdin* *stdout* -file --opt --help --version

If you set extended attributes with `chattr`, you can view them with `lsattr` (list attributes). The output uses the same letters as `chattr`; for example, this file is immutable and undeleteable:

```
$ lsattr myfile
-u--i--- myfile
```

Useful options

-R Recursively process directories.

-a List all files, including those whose names begin with a dot.

-d If listing a directory, do not list its contents, just the directory itself.

With no files specified, `lsattr` prints the attributes of all files in the current directory.