# Important Bash Commands

(Mostly just copied from manuals)



## grep - Print lines matching a pattern

```
grep [options] PATTERN [FILE...]
grep [options] [-e PATTERN | -f FILE] [FILE...]
```

Grep searches the named input FILEs (or standard input if no files are named, or the file name - is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

## ls - List directory contents

ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

For each operand that names a file of a type other than directory, ls displays its name as well as any requested, associated information. For each operand that names a file of type directory, ls displays the names of files contained within that directory, as well as any requested, associated information.

If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

-l List in long format. In terminal, total sum of file sizes displayed before listing

-a Include directory entries whose names begin with a dot (.).
-d Directories are listed as plain files (not searched recursively).
-i For each file, print the file's file serial number (inode number).

## ps - Process status

ps [-AaCcEefhjlMmrSTvwXx] [-O fmt | -o fmt] [-G gid[,gid...]]
[-g grp[,grp...]] [-u uid[,uid...]] [-p pid[,pid...]]
[-t tty[,tty...]] [-U user[,user...]]
ps [-L]

The ps utility displays a header line, followed by lines containing information about all of your processes that have controlling terminals.

-e Display information about other users' processes, including those without controlling terminals.
-f Display the uid, pid, parent pid, recent CPU usage, process start time, controlling tty, elapsed CPU usage, and the associated command. If the -u option is also used, display the user name rather then the numeric uid.

## sort - Sort lines of text files

sort [OPTION]... [FILE]...

Write sorted concatenation of all FILE(s) to standard output.

-r reverse the result of comparisons
-k --key=POS1[,POS2] start at POS1 and end at POS2
-u filter out duplicate lines

## uniq - Report or filter out repeated lines in a file

uniq [-c | -d | -u] [-i] [-f num] [-s chars] [input_file [output_file]]

The uniq utility reads the specified input_file comparing adjacent lines, and writes a copy of each unique input line to the output_file. If input_file is a single dash (`-') or absent, the standard input is read. If output_file is absent, standard output is used for output. The second and succeeding copies of identical adjacent input lines are not written. Repeated lines in the input will not be detected if they are not adjacent, so it may be necessary to sort the files first.

-c Precede each output line with the **c**ount of the number of times the line occurred in the input, followed by a single space.

## cat - Concatenate and print files

cat [-benstuv] [file ...]

The cat utility reads files sequentially, writing them to the standard output. The file operands are processed in command-line order. If file is a single dash (`-') or absent, cat reads from the standard input. If file is a UNIX tdomain socket, cat connects to it and then reads it until EOF. This complements the UNIX domain binding capability available in inetd(8).

## head - Display first lines of a file

head [-n count | -c bytes] [file ...]

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified. If count is omitted it defaults to 10.

If more than a single file is specified, each file is preceded by a header consisting of the string ``==> XXX <==`' where ``XXX'' is the name of the file.

-n negative n counts back from the last line

## tail - Display the last part of a file

tail [-F | -f | -r] [-q] [-b number | -c number | -n number] [file ...]

The tail utility displays the contents of file or, by default, its standard input, to the standard output.

The display begins at a byte, line or 512-byte block location in the input. Numbers having a leading plus (`+') sign are relative to the beginning of the input, for example, ``-c +2'' starts the display at the

second byte of the input. Numbers having a leading minus (`-') sign or no explicit sign are relative to the end of the input, for example, ``-n 2'' displays the last two lines of the input. The default starting location is ``-n 10'', or the last 10 lines of the input.

## cut - Cut out selected portions of each line of a file

```
cut -b list [-n] [file ...]
cut -c list [file ...]
cut -f list [-d delim] [-s] [file ...]
```

The cut utility cuts out selected portions of each line (as specified by list) from each file and writes them to the standard output. If no file arguments are specified, or a file argument is a single dash (`-'), cut reads from the standard input. The items specified by list can be in terms of column position or in terms of fields delimited by a special character. Column numbering starts from 1.

The list option argument is a comma or whitespace separated set of numbers and/or number ranges. Number ranges consist of a number, a dash (`-'), and a second number and select the fields or columns from the first number to the second, inclusive. Numbers or number ranges may be preceded by a dash, which selects all fields or columns from 1 to the last number. Numbers or number ranges may be followed by a dash, which selects all fields or columns from the last number to the end of the line. Numbers and number ranges may be repeated, overlapping, and in any order. If a field or column is specified multiple times, it will appear only once in the output. It is not an error to select fields or columns not present in the input line.

-f list The list specifies fields, separated in the input by the field delimiter character (see the -d option.) Output fields are separated by a single occurrence of the field delimiter character.

-d delim Use delim as the field delimiter character instead of the tab character.

## wc - Word, line, character, and byte count

```
wc [-clmw] [file ...]
```

The wc utility displays the number of lines, words, and bytes contained in each input file, or standard input (if no file is specified) to the standard output. A line is defined as a string of characters delimited by a <newline> character. Characters beyond the final <newline> character will not be included in the line count.

A word is defined as a string of characters delimited by white space characters. White space characters are the set of characters for which the iswspace(3) function returns true. If more than one input file is specified, a line of cumulative counts for all the files is displayed on a separate line after the output for the last file.

-l The number of lines in each input file is written to the standard output.

## diff - Compare files line by line

diff [OPTION]... FILES

## svn - Subversion command line client tool

svn command [options] [args]

### commit (ci)

commit [PATH...] Send changes from your working copy to the repository.

### add

add PATH... Put files and directories under version control, scheduling them for addition to repository. They will be added in next commit.

### delete (del, remove, rm)

delete PATH... Each item specified by a PATH is scheduled for deletion upon the next commit. Files, and directories that have not been committed, are immediately removed from the working copy unless the --keep-local option is given. PATHs that are, or contain, unversioned or modified items will not be removed unless the --force option is given.
delete URL... Each item specified by a URL is deleted from the repository via an immediate commit.

### move (mv, rename, ren)

move SRC... DST Move and/or rename something in working copy or repository. When moving multiple sources, they will be added as children of DST, which must be a directory.

## update (up)

```
update [PATH...]    Bring changes from the repository into the working copy.
If no revision is given, bring working copy up-to-date with HEAD rev. Else
synchronize working copy to revision given by -r.
For each updated item a line will start with a character reporting the action taken.
These characters have the following meaning:
A  Added, D  Deleted, U  Updated, C  Conflict, G  Merged, E  Existed
```

## info

```
info [TARGET[@REV]...]     Display information about a local or remote item.
Print information about each TARGET (default: '.').
  TARGET may be either a working-copy path or URL.  If specified, REV
  determines in which revision the target is first looked up.
```

## log

```
Show the log messages for a set of revision(s) and/or file(s).
log [PATH]          Print the log messages for a local PATH (default: '.').
     The default revision range is BASE:1.
log URL[@REV] [PATH...]
       Print the log messages for the PATHs (default: '.') under URL.
     If specified, REV determines in which revision the URL is first
     looked up, and the default revision range is REV:1; otherwise,
     the URL is looked up in HEAD, and the default revision range is
     HEAD:1.
```

## status (stat, st)

```
status [PATH...]    Print the status of working copy files and directories.
```

## diff (di)

```
diff [-c M | -r N[:M]] [TARGET[@REV]...]
Display the changes made to TARGETs as they are seen in REV between
     two revisions.  TARGETs may be all working copy paths or all URLs.
     If TARGETs are working copy paths, N defaults to BASE and M to the
     working copy; if URLs, N must be specified and M defaults to HEAD.
     The '-c M' option is equivalent to '-r N:M' where N = M-1.
     Using '-c -M' does the reverse: '-r M:N' where N = M-1.


diff [-r N[:M]] --old=OLD-TGT[@OLDREV] [--new=NEW-TGT[@NEWREV]] [PATH...]
Display the differences between OLD-TGT as it was seen in OLDREV and
     NEW-TGT as it was seen in NEWREV.  PATHs, if given, are relative to
     OLD-TGT and NEW-TGT and restrict the output to differences for those
```

paths.  OLD-TGT and NEW-TGT may be working copy paths or URL[@REV].
NEW-TGT defaults to OLD-TGT if not specified.  -r N makes OLDREV default
to N, -r N:M makes OLDREV default to N and NEWREV default to M.

diff OLD-URL[@OLDREV] NEW-URL[@NEWREV]
Shorthand for 'svn diff --old=OLD-URL[@OLDREV] --new=NEW-URL[@NEWREV]'

Use just 'svn diff' to display local modifications in a working copy.

## chmod - Change file modes or Access Control Lists

chmod [-fv] [-R [-H | -L | -P]] mode file ...
chmod [-fv] [-R [-H | -L | -P]] [-a | +a | =a] ACE file ...
chmod [-fhv] [-R [-H | -L | -P]] [-E] file ...
chmod [-fhv] [-R [-H | -L | -P]] [-C] file ...
chmod [-fhv] [-R [-H | -L | -P]] [-N] file ...

who ::= a | u | g | o
The who symbols ``u'', ``g'', and ``o'' specify the user, group, and
other parts of the mode bits, respectively.  The who symbol ``a'' is
equivalent to ``ugo''.

op ::= + | - | =

perm ::= r | s | t | w | x | X | u | g | o
The perm symbols represent the portions of the mode bits as follows:

| | |
|---|---|
| r | The read bits.6 |
| s | The set-user-ID-on-execution and set-group-ID-on-execution bits. |
| t | The sticky bit. |
| w | The write bits. |
| x | The execute/search bits. |
| X | The execute/search bits if the file is a directory or any of the execute/search bits are set in the original (unmodified) mode.  Operations with the perm symbol ``X'' are only meaningful in conjunction with the op symbol ``+'', and are ignored in all other cases. |
| u | The user permission bits in the original mode of the file. |
| g | The group permission bits in the original mode of the file. |
| o | The other permission bits in the original mode of the file. |

## ln, link - makes links

ln [-Ffhinsv] source_file [target_file]

```
     ln [-Ffhinsv] source_file ... target_dir
     link source_file target_file
The ln utility creates a new directory entry (linked file) which has the
     same modes as the original file.  It is useful for maintaining multiple
     copies of a file in many places at once without using up storage for the
     ``copies''; instead, a link ``points'' to the original copy.  There are
     two types of links; hard links and symbolic links.  How a link
     ``points'' to a file is one of the differences between a hard and sym-
     bolic link.

-s      Create a symbolic link
```

## rm, unlink - remove directory entries

```
     rm [-dfiPRrvW] file ...
     unlink file
The rm utility attempts to remove the non-directory type files specified
     on the command line.  If the permissions of the file do not permit writ-
     ing, and the standard input device is a terminal, the user is prompted
     (on the standard error output) for confirmation.

-r     recursive
-f     Force (no confirmation)
```

## mkdir - make directories

```
       mkdir [-pv] [-m mode] directory_name …
The mkdir utility creates the directories named as operands, in the
     order specified, using mode rwxrwxrwx (0777) as modified by the current
     umask(2).
```

## rmdir - removes directories

```
       rmdir [-p] directory …
The rmdir utility removes the directory entry specified by each
     directory argument, provided it is empty.

     Arguments are processed in the order given.  In order to remove both a
     parent directory and a subdirectory of that parent, the subdirectory
     must be specified first so the parent directory is empty when rmdir
     tries to remove it.
```

## cp - copy files

```
     cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source_file target_file
     cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source_file... target_directory
In the first synopsis form, the cp utility copies the contents of the
     source_file to the target_file.  In the second synopsis form, the con-
```

tents of each named source_file is copied to the destination target_directory. The names of the files themselves are not changed. If cp detects an attempt to copy a file to itself, the copy will fail.

-R If source_file designates a directory, cp copies the directory and the entire subtree connected at that point. If the source_file ends in a /, the contents of the directory are copied rather than the directory itself. This option also causes symbolic links to be copied, rather than indirected through, and for cp to create special files rather than copying them as normal files. Created directories have the same mode as the corresponding source directory, unmodified by the process' umask.

In -R mode, cp will continue copying even if errors are detected.

Note that cp copies hard-linked files as separate files. If you need to preserve hard links, consider using tar(1), cpio(1), or pax(1) instead.

## mv - move files

```
mv [-f | -i | -n] [-v] source target
mv [-f | -i | -n] [-v] source ... directory
```

In its first form, the mv utility renames the file named by the source operand to the destination path named by the target operand. This form is assumed when the last operand does not name an already existing directory.

In its second form, mv moves each file named by a source operand to a destination file in the existing directory named by the directory operand. The destination path for each operand is the pathname produced by the concatenation of the last operand, a slash, and the final pathname component of the named file.

## vim - Vi IMproved, a programmers text editor

```
vim [options] [file ..]
vim [options] -
vim [options] -t tag
vim [options] -q [errorfile]
```

## pico - Nano's ANOther editor, an enhanced free Pico clone

```
nano [OPTIONS] [[+LINE,COLUMN] FILE]...
```

# Shell Commands

## grep [-v] [ $ ^ . * ]

### Synopsis

grep [OPTIONS] PATTERN [FILE...]

### Description

grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

### Examples

**s4258788@moss:~/csse2310/exam$** cat foo
apples
bananas
capsicums
drugs
eggplants
fennels

**s4258788@moss:~/csse2310/exam$** grep an foo
bananas
eggplants

### -v, --invert match

Invert the sense of matching, to select non-matching lines.

### Examples

**s4258788@moss:~/csse2310/exam$** grep -v an foo
apples
capsicums
drugs
fennels

### * (asterisk)

The preceding item will be matched zero or more times

### . (period)

Matches any single (one) character

### ^ (carat)

Matches the empty string at the beginning of a line

### $ (dollar)

Matches the empty string at the end of a line

# ls [-ladi]

## Synopsis

ls [OPTION]... [FILE]...

### Description

List information about the FILEs (the current directory by default). Sort entries

alphabetically if none of -cftuvSUX nor --sort.

## -a, --all

do not ignore entries starting with .

## -i, --inode

print the index number of each file

## -l

use a long listing format

## -d, --directory

list directory entries instead of contents, and do not dereference symbolic links

## Examples

**s4258788@moss:~/csse2310/exam$** ls -a
. .. .bar baz foo mydir myfile

**s4258788@moss:~/csse2310/exam$** ls -l
total 12
-rw-r--r-- 1 s4258788 students 0 Apr 16 15:34 baz
-rw-r--r-- 1 s4258788 students 49 Apr 16 14:58 foo
drwxr-xr-x 2 s4258788 students 4096 Apr 16 15:36 mydir
-rw-r--r-- 1 s4258788 students 14 Apr 16 15:34 myfile

**s4258788@moss:~/csse2310/exam$** ls -li
total 12
33974218 -rw-r--r-- 1 s4258788 students 0 Apr 16 15:34 baz
28434368 -rw-r--r-- 1 s4258788 students 49 Apr 16 14:58 foo
32850991 drwxr-xr-x 2 s4258788 students 4096 Apr 16 15:36 mydir
8965819 -rw-r--r-- 1 s4258788 students 14 Apr 16 15:34 myfile

**s4258788@moss:~/csse2310/exam$** ls -ld
drwxr-xr-x 3 s4258788 students 4096 Apr 16 15:36 .

# ps [-ef]

## Synopsis

ps [options]

### Description

ps displays information about a selection of the active processes.

### -e

Select all processes.  Identical to -A.

### -f

Does full format listing.

## Examples

**s4258788@moss:~/csse2310/exam$** ps
```
  PID TTY          TIME CMD
52410 pts/167  00:00:00 bash
64185 pts/167  00:00:00 ps
```
**s4258788@moss:~/csse2310/exam$** ps -f
```
UID        PID  PPID  C STIME TTY          TIME CMD
s4258788 52410 52409  0 14:45 pts/167  00:00:00 -bash
s4258788 64187 52410 29 15:43 pts/167  00:00:00 ps -f
```
**s4258788@moss:~/csse2310/exam$** ps -e
```
  PID TTY          TIME CMD
    1 ?        00:01:00 init
    2 ?        00:00:02 kthreadd
    3 ?        00:00:01 migration/0
    4 ?        00:00:13 ksoftirqd/0
    5 ?        00:00:00 migration/0
    6 ?        00:00:00 watchdog/0
    7 ?        00:00:06 migration/1
    8 ?        00:00:00 migration/1
    9 ?        00:04:21 ksoftirqd/1
… etc…
65097 ?        00:06:56 tmux
65098 pts/25   00:00:00 bash
65109 pts/167  00:00:00 ps
65110 pts/167  00:00:00 less
65264 ?        00:10:15 tmux
65265 pts/113  00:00:00 bash
```

# sort [-r -k]

## Synopsis

sort [OPTION]... [FILE]...

### Description

Write sorted concatenation of all FILE(s) to standard output.

## -r, --reverse

reverse the result of comparisons

## -k, --key=POS1[,POS2]

start a key at POS1 (origin 1), end it at POS2 (default end of line)

## Examples

**s4258788@moss:~/csse2310/exam$** cat 1sort
apple
guava
banana

**s4258788@moss:~/csse2310/exam$** sort 1sort
apple
banana
guava

**s4258788@moss:~/csse2310/exam$** sort -r 1sort
guava
banana
apple

**s4258788@moss:~/csse2310/exam$** ls -l
total 12
-rw-r--r-- 1 s4258788 students 19 Apr 16 16:21 1sort
-rw-r--r-- 1 s4258788 students  7 Apr 16 16:21 3sort
-rw-r--r-- 1 s4258788 students 63 Apr 16 16:22 6sort

**s4258788@moss:~/csse2310/exam$** ls -l | sort -k 5
total 12
-rw-r--r-- 1 s4258788 students  7 Apr 16 16:21 3sort
-rw-r--r-- 1 s4258788 students 19 Apr 16 16:21 1sort
-rw-r--r-- 1 s4258788 students 63 Apr 16 16:22 6sort

(Sorted by the 5$^{th}$ column)

# uniq [-c]

uniq – report or omit repeated lines

## Synopsis

uniq [OPTION]... [INPUT [OUTPUT]]

### Description

Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).

With no options, matching lines are merged to the first occurrence.

## -c, --count

prefix lines by the number of occurrences

## Examples

**s4258788@moss:~/csse2310/exam$** cat foo
this line is not repeated
this line is repeated 4 times
this line is repeated 4 times
this line is repeated 4 times
this line is repeated 4 times
this line is not
but this line is
but this line is
what about me?
nope, you aren't! but i am
nope, you aren't! but i am

**s4258788@moss:~/csse2310/exam$** uniq foo
this line is not repeated
this line is repeated 4 times
this line is not
but this line is
what about me?
nope, you aren't! but i am

**s4258788@moss:~/csse2310/exam$** uniq -c foo
1 this line is not repeated
4 this line is repeated 4 times
1 this line is not
2 but this line is
1 what about me?
2 nope, you aren't! but i am

# cat

cat – concatenate files and print on the standard output

## Synopsis

cat [OPTION]... [FILE]...

### Description

Concatenate FILE(s), or standard input, to standard output.

## Examples

**s4258788@moss:~/csse2310/exam$** cat
*hello (input in italics)*
hello
*you there?*
you there?
*bye*
bye
*^D (Control-D. 'End of File') (Not shown on console)*

**s4258788@moss:~/csse2310/exam$** cat bar
Hello, World!

**s4258788@moss:~/csse2310/exam$** cat baz
Another file.

**s4258788@moss:~/csse2310/exam$** cat bar baz
Hello, World!
Another file.

**s4258788@moss:~/csse2310/exam$** cat bar - baz
Hello, World!
*this is input*
this is input
*^D*
Another file.

# head [-]

head – output the first part of files

## Synopsis

head [OPTION]... [FILE]...

### Description

Print the first 10 lines of each FILE to standard output.  With more than one FILE,

precede each with a header giving the file name.  With no FILE, or when FILE is  -,

read standard input.

## -n (number)

prints the first “n“ lines instead of the first 10

## Examples

**s4258788@moss:~/csse2310/exam$** head -4 foo
This is a file with more than 10 lines.
I'm not going to include it, you'll just have to trust me on this one.
What? You don't trust me? Well too bad, so sad.
Monkey see, monkey do

**s4258788@moss:~/csse2310/exam$** head bar
I think this file has a few less than ten lines.
Yep! and I can prove it
or can i?

**s4258788@moss:~/csse2310/exam$** head -2 foo bar
==> foo <==
This is a file with more than 10 lines.
I'm not going to include it, you'll just have to trust me on this one.

==> bar <==
I think this file has a few less than ten lines.
Yep! and I can prove it

**s4258788@moss:~/csse2310/exam$** grep and foo | head
and why did he want a face?
well... i can tell you and i won't feel too bad or sad
or mad or mad and sand
like a sand castle
or a jumping castle that is bouncy and fun

# tail[-]

Look at head and reverse it.

# cut [-f -d]

cut - remove sections from each line of files

## Synopsis

cut OPTION... [FILE]...

### Description

Print selected parts of lines from each FILE to standard output.

## -f, --fields=LIST

select only these fields; also print any line that contains no delimiter character, unless the -s option is specified

## -d, --delimiter=DELIM

use DELIM instead of TAB for field delimiter

## Examples

(Not really sure on this one, sorry)

# wc [-l]

wc - print newline, word, and byte counts for each file

## Synopsis

wc [OPTION]... [FILE]...

### Description

Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. With no FILE, or when FILE is -, read standard input.

## -l, --lines

print the newline counts

## Examples

**s4258788@moss:~/csse2310/exam$** wc foo
19 100 469 foo

**s4258788@moss:~/csse2310/exam$** wc foo -l
19 foo

**s4258788@moss:~/csse2310/exam$** wc foo bar baz -l
19 foo
3 bar
1 baz
23 total

# diff

## Synopsis

diff [OPTION]... FILES

## Description

diff - compare files line by line

## Examples

**s4258788@moss:~/csse2310/exam$** cat foo
these files are almost the same
but not quite.
something is different

**s4258788@moss:~/csse2310/exam$** cat bar
These files are almost the same.
But something is a little different.

**s4258788@moss:~/csse2310/exam$** cat baz
These files are completely
and utterly identical
Seriously, one is a direct copy of the other

**s4258788@moss:~/csse2310/exam$** cat faz
These files are completely
and utterly identical
Seriously, one is a direct copy of the other

**s4258788@moss:~/csse2310/exam$** diff foo bar
1,3c1,2
< these files are almost the same
< but not quite.
< something is different
---
> These files are almost the same.
> But something is a little different.

**s4258788@moss:~/csse2310/exam$** diff faz baz
**s4258788@moss:~/csse2310/exam$**

# chmod [all major options]

chmod - change file mode bits

## Synopsis

chmod [OPTION]... MODE[,MODE]... FILE...

## Description

chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

**+** adds em, **-** takes em away!

A combination of the letters **ugoa** controls which users' access to the file will be changed:

**u** - the user who owns it

**g** - other users in the file's group

**o** - other users not in the file's group

**a** - all users

The letters **rwxXst** select file mode bits for the affected users:

**r** - read

**w - write**

**x** - execute (or search for directories)

**X** - x only if the file is a directory or already has execute permission for some user

**s** - set user or group ID on execution

**t** - restricted deletion flag or sticky bit

A numeric mode is from **one** to **four** octal digits **(0-7)**, derived by adding up the bits with values **4**, **2**, and **1**. Omitted digits are assumed to be leading zeros.

**First digit** - Set user ID**(4)**, Set group ID**(2)**, restricted deletion or sticky attributes**(1)**

**Second digit** - Select permissions for user who owns the file. Read**(4)**, write**(2)**, execute**(1)**

**Third digit** - Select permissions for other users in the file's group. Same values ^

**Fourth digit** - Other users not in the file's group. Same values ^

### -v, --verbose
output a diagnostic for every file processed

### -c, --changes
like verbose but report only when a change is made

### -f, --silent, --quiet
suppress most error messages

### -R, --recursive
change files and directories recursively

## Examples
**s4258788@moss:~/csse2310/exam$** ls -l
total 0
-rw-r--r-- 1 s4258788 students 0 Apr 17 11:34 foo

**s4258788@moss:~/csse2310/exam$** chmod a-r foo
**s4258788@moss:~/csse2310/exam$** ls -l
total 0
--w------- 1 s4258788 students 0 Apr 17 11:34 foo

**s4258788@moss:~/csse2310/exam$** chmod u+rwx foo
**s4258788@moss:~/csse2310/exam$** ls -l
total 0
-rwx------ 1 s4258788 students 0 Apr 17 11:34 foo *(executables are shown in green)*

**s4258788@moss:~/csse2310/exam$** chmod g+wr,o+r foo
**s4258788@moss:~/csse2310/exam$** ls -l
total 0
-rwxrw-r-- 1 s4258788 students 0 Apr 17 11:34 foo

**s4258788@moss:~/csse2310/exam$** chmod 777 foo
**s4258788@moss:~/csse2310/exam$** ls -l
total 0
-rwxrwxrwx 1 s4258788 students 0 Apr 17 11:34 foo

**s4258788@moss:~/csse2310/exam$** chmod 0 foo
**s4258788@moss:~/csse2310/exam$** ls -l
total 0
---------- 1 s4258788 students 0 Apr 17 11:34 foo

# ln [-s]

ln - make links between files

## Synopsis

ln [OPTION]... TARGET (2nd form)

ln [OPTION]... TARGET... DIRECTORY (3rd form)

## Description

In the 2nd form, create a link to TARGET in the current directory. In the 3rd and 4th forms, create links to each TARGET in DIRECTORY. Create hard links by default, symbolic links with --symbolic. When creating hard links, each TARGET must exist. Symbolic links can hold arbitrary text; if later resolved, a relative link is interpreted in relation to its parent directory.

## -s, --symbolic

make symbolic links instead of hard links

## Examples

**s4258788@moss:~/csse2310/exam**$ ls
foo mydir *(directories are dark blue)*
**s4258788@moss:~/csse2310/exam$** ls mydir/
bar baz foo nesteddir
**s4258788@moss:~/csse2310/exam$** ls mydir/nesteddir/
myfile
**s4258788@moss:~/csse2310/exam$** cat mydir/baz
this is a file with one line in it

**s4258788@moss:~/csse2310/exam$** ln mydir/baz baz-link
**s4258788@moss:~/csse2310/exam$** cat baz-link
this is a file with one line in it

**s4258788@moss:~/csse2310/exam$** ln -s mydir/nesteddir/ nesteddir-link
**s4258788@moss:~/csse2310/exam**$ ls
baz-link foo mydir nesteddir-link *(symbolic links are cyan)*
**s4258788@moss:~/csse2310/exam$** touch nesteddir-link/mynewfile
**s4258788@moss:~/csse2310/exam$** ls mydir/nesteddir/
myfile mynewfile

**s4258788@moss:~/csse2310/exam$** rm -rf mydir/nesteddir/*
**s4258788@moss:~/csse2310/exam$** ls nesteddir-link/

**s4258788@moss:~/csse2310/exam$** rm -rf mydir/*
**s4258788@moss:~/csse2310/exam$** ls
baz-link foo mydir nesteddir-link *(a symbolic link with a non-existent destination (I removed it!))*

**s4258788@moss:~/csse2310/exam$** cat baz-link
this is a file with one line in it
**s4258788@moss:~/csse2310/exam$** ls mydir/

# rm [-rf]

rm - remove files or directories

## Synopsis

rm [OPTION]... FILE...

## Description

rm removes each specified file. By default, it does not remove directories.

## -f, --force

ignore nonexistent files, never prompt.  Remove all files whether write protected or not.

## -r, -R, --recursive

remove directories and their contents recursively

Examples

**s4258788@moss:~/csse2310/exam$** ls
1file  2file  3file  onedir  twodir
**s4258788@moss:~/csse2310/exam$** ls onedir/
bar  baz  foo  reddir

**s4258788@moss:~/csse2310/exam$** rm 3file
**s4258788@moss:~/csse2310/exam$** ls
1file  2file  onedir  twodir

**s4258788@moss:~/csse2310/exam$** rm onedir
rm: cannot remove `onedir': Is a directory

**s4258788@moss:~/csse2310/exam$** rm -r onedir
**s4258788@moss:~/csse2310/exam$** ls
1file  2file  twodir

**s4258788@moss:~/csse2310/exam$** rm -f twodir/
rm: cannot remove `twodir/': Is a directory

**s4258788@moss:~/csse2310/exam$** dir
1file  2file  twodir

**s4258788@moss:~/csse2310/exam$** chmod a-w 1file 2file
**s4258788@moss:~/csse2310/exam$** rm 1file
rm: remove write-protected regular empty file `1file'? *y*

**s4258788@moss:~/csse2310/exam$** rm -f 2file
**s4258788@moss:~/csse2310/exam$** ls
twodir

# mkdir

mkdir - make directories

## Synopsis

mkdir [OPTION]... DIRECTORY...

## Description

Create the DIRECTORY(ies), if they do not already exist.

## Examples

**s4258788@moss:~/csse2310/exam$** ls
twodir
**s4258788@moss:~/csse2310/exam$** mkdir mydir
**s4258788@moss:~/csse2310/exam$** ls
mydir  twodir

# rmdir

rmdir - remove empty directories

## Synopsis

rmdir [OPTION]... DIRECTORY...

## Description

Remove the DIRECTORY(ies), if they are empty.

Examples
**s4258788@moss:~/csse2310/exam$** ls
twodir
**s4258788@moss:~/csse2310/exam$** ls twodir/
bar  foo

**s4258788@moss:~/csse2310/exam$** rmdir twodir/
rmdir: failed to remove `twodir/': Directory not empty

**s4258788@moss:~/csse2310/exam$** rm -f twodir/*

**s4258788@moss:~/csse2310/exam$** rmdir twodir/

**s4258788@moss:~/csse2310/exam$** ls

**s4258788@moss:~/csse2310/exam$**

# cp [-r]

cp - copy files and directories

## Synopsis

cp [OPTION]... [-T] SOURCE DEST

cp [OPTION]... SOURCE... DIRECTORY

cp [OPTION]... -t DIRECTORY SOURCE...

## Description

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

## -R, -r, --recursive

copy directories recursively

## Examples

**s4258788@moss:~/csse2310/exam$** ls
1dir  2dir  bar  foo
**s4258788@moss:~/csse2310/exam$** cat foo
this file
has two lines
... NOT!

**s4258788@moss:~/csse2310/exam$** cp foo ./2dir
**s4258788@moss:~/csse2310/exam$** ls 2dir/
foo

**s4258788@moss:~/csse2310/exam$** cat 2dir/foo
this file
has two lines
... NOT!

**s4258788@moss:~/csse2310/exam$** ls 1dir/
apple  banana  orange
**s4258788@moss:~/csse2310/exam$** cp -r 1dir/ 2dir/

**s4258788@moss:~/csse2310/exam$** ls 2dir/
1dir  foo

**s4258788@moss:~/csse2310/exam$** ls 2dir/1dir/
apple  banana  orange

# mv

mv - move (rename) files

## Synopsis

mv [OPTION]... [-T] SOURCE DEST

mv [OPTION]... SOURCE... DIRECTORY

mv [OPTION]... -t DIRECTORY SOURCE...

## Description

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

## Examples

**s4258788@moss:~/csse2310/exam$** ls
bar  foo  thatdir  thisdir

**s4258788@moss:~/csse2310/exam$** mv foo foobar
**s4258788@moss:~/csse2310/exam$** ls
bar  foobar  thatdir  thisdir

**s4258788@moss:~/csse2310/exam$** mv foobar thatdir/
**s4258788@moss:~/csse2310/exam**$ ls thatdir/
foobar
**s4258788@moss:~/csse2310/exam$** ls
bar  thatdir  thisdir

**s4258788@moss:~/csse2310/exam$** mv thatdir/foobar ./foo
**s4258788@moss:~/csse2310/exam$** ls
bar  foo  thatdir  thisdir
**s4258788@moss:~/csse2310/exam$** ls thatdir/
**s4258788@moss:~/csse2310/exam$**

An example of the output produced by 'ls -l' is shown below.

| | | | | | | |
|---|---|---|---|---|---|---|
| drwx------ | 2 | richard | staff | 2048 | Jan 2 1997 | private |
| drwxrws--- | 2 | richard | staff | 2048 | Jan 2 1997 | admin |
| -rw-rw---- | 2 | richard | staff | 12040 | Aug 20 1996 | admin/userinfo |
| drwxr-xr-x | 3 | richard | user | 2048 | May 13 09:27 | public |

Field 1: a set of ten permission flags.
Field 2: link count
Field 3: owner of the file
Field 4: associated group for the file
Field 5: size in bytes
Field 6-8: date of last modification (format varies, but always 3 fields)
Field 9: name of file (possibly with path, depending on how ls was called) The permission flags are read as follows (left to right)

| position | Meaning |
|---|---|
| 1 | directory flag, 'd' if a directory, '-' if a normal file, something else occasionally may appear here for special devices. 'l' indicated a link |
| 2,3,4 | read, write, execute permission for User (Owner) of file |
| 5,6,7 | read, write, execute permission for Group |
| 8,9,10 | read, write, execute permission for **Other** (Other meaning NOT in user or group) |
| value | Meaning |
| - | in any position means that flag is not set |
| r | file is readable by owner, group or other. If set on a directory, you can list the directory contents. |
| w | file is writeable. On a directory, write access means you can add or delete files |
| x | file is executable (only for programs and shell scripts - not useful for data files). Execute permission on a directory means you can traverse the directory. |
| s | in the place where 'x' would normally go is called the set-UID or set-groupID flag. |

On an executable program with set-UID or set-groupID, that program runs with the effective permissions of its owner or group.

For a directory, the set-groupID flag means that all files created inside that directory will inherit the group of the directory. Without this flag, a file takes on the primary group of the user creating the file. This property is important to people trying to maintain a directory as group accessible. The subdirectories also inherit the set-groupID property.

## Linux Read mode permissions

- Read access on a file allows you to view file
- Read access on a directory allows you to view directory contents with ls command

## Write mode permissions

- Write access on a file allows you to write to file
- Write access on a directory allows you to remove or add new files

## Execute mode permissions

- Execute access on a file allows to run program or script

Execute access on a directory allows you access file in the directory