

Sidebar

Linux · grep Cheat Sheet

Searching text, at a glance. The options were run on GNU grep 3.0 (the one on Linux and in Git Bash). For the regex syntax itself, see [Learning Regular Expressions](#); for editing what you find, [SED — A Course for Beginners](#) and [AWK](#).

BASICS

- `grep PAT file` Print lines that match
- `grep PAT f1 f2` Several files (output gets the file name)
- `cmd | grep PAT` Filter another command's output
- `-i` Ignore case
- `-v` Invert: lines that do NOT match
- `-w` Whole words only
- `-x` Whole line must match
- `-F` Fixed string: no regex (a . is just a dot)
- `-e A -e B` Several patterns (A or B)
- `-f file` Patterns from a file, one per line

OUTPUT CONTROL

- `-n` Show line numbers
- `-c` Count matching lines (not matches)
- `-o` Print only the matching part
- `-l / -L` Only names of files with / without a match
- `-m N` Stop after N matching lines
- `-H / -h` Always / never show the file name
- `-b` Byte offset of each match
- `-q` Quiet: print nothing, just set the exit code
- `-s` Hide errors about missing or unreadable files
- `--color=auto` Highlight the match

CONTEXT

- A N** N lines After each match
- B N** N lines Before each match
- C N** N lines of Context either side
- Groups of matches are separated by a -- line
- no-group-separator** Leave that separator out

FILES & FOLDERS

- r** Recurse into folders (no path: current folder)
- R** As -r, but also follows symbolic links
- include='*.py'** Only search files matching this name
- exclude='*.log'** Skip files matching this name
- exclude-dir=.git** Skip a folder (repeat the option for more)
- I** Skip binary files
- a** Treat binary files as text
- Z** End file names with a NUL (for xargs -0)

BASIC REGEX (DEFAULT)

- ^ \$** Start / end of line
- .** Any one character
- *** Zero or more of the last item
- [abc] [^abc]** One of / none of these characters
- [a-z]** A range
- \< \> \b** Start / end of a word / either edge
- \(ab\) \1** Group, and a back-reference to it
- \{2,4\}** Between 2 and 4 of the last item
- \+ \? \|** One or more / optional / or (GNU grep)

EXTENDED REGEX (-E)

- + ? |** One or more / optional / or, no backslashes
- (ab|cd)** Group with alternatives
- {2,4}** Between 2 and 4
- E '^(error|warn)'** Lines starting with either word
- E 'colou?r'** color or colour
- E '[0-9]{3}-[0-9]{4}'** A phone-style number

CHARACTER CLASSES

<code>[:digit:]</code>	0-9
<code>[:alpha:]</code>	Letters
<code>[:alnum:]</code>	Letters and digits
<code>[:upper:]</code> <code>[:lower:]</code>	Upper / lower case letters
<code>[:space:]</code>	Space, tab, newline
<code>[:punct:]</code>	Punctuation

PERL REGEX (-P, GNU ONLY)

<code>\d \s \w</code>	Digit / space / word character
<code>*? +?</code>	Lazy: as little as possible
<code>(?=x) (?!x)</code>	Followed by / not followed by x
<code>(?<=x) (?<!x)</code>	Preceded by / not preceded by x
<code>\K</code>	Forget what was matched so far
<code>-oP 'id=\K\d+'</code>	Print just the number after id=

PIPELINES & RECIPES

<code>ps aux grep [n]ginx</code>	Find a process without matching grep itself
<code>grep -rn TODO --include='*.js' .</code>	Find TODOs in JavaScript files
<code>grep -rILZ PAT . xargs -0 sed -i ...</code>	Replace across files: add -- exclude-dir=.git, or -r edits .git too
<code>grep -Fxf a b</code>	Lines of file b that are also in file a
<code>grep -vFxf a b</code>	Lines of file b that are NOT in file a
<code>grep -c ' ' file</code>	Count all lines in a file
<code>grep -o PAT f sort uniq -c</code>	Count how often each match occurs
<code>grep -v '^#' f grep -v '^\$'</code>	Hide comments and blank lines

EXIT STATUS

<code>0</code>	At least one line matched
<code>1</code>	No lines matched (not an error)
<code>2</code>	A real error (bad file, bad pattern)
<code>if grep -q PAT f; then ...</code>	Test for a match in a script

THE #1 TRAP: QUOTE YOUR PATTERN

Always put the pattern in single quotes. Unquoted, the shell gets there first: `grep error|warn log pipes` grep into a command called `warn`, and `*` or `$` are expanded before grep ever sees them. And in basic regex a bare `|`, `+`, `?`, `(` or `{` is just a literal character: add `-E` if you meant it as an operator. Also, `-P` is not in every grep (macOS/BSD grep often lacks it).