# grep Search text lines

## CONCEPTS

grep selects whole lines. Add -o to print only the matched part.

**Input** files or stdin
↓ pattern
**Match** per line
↓ options
**Select** -v, -m, -w, -x
↓ format
**Output** -n, -o, -c, -l

Quote patterns in single quotes so the shell leaves *, $ and | alone.

Exit status: 0 = a line matched, 1 = none, 2 = error. Handy in if.

## MATCHING

`grep "<pattern>" <file>`
Lines matching a pattern

`grep -i "<pattern>" <file>`
Ignore case

`grep -v "<pattern>" <file>`
Invert: lines that do not match

`grep -w "<word>" <file>`
Match whole words only

`grep -x "<line>" <file>`
Match whole lines only

`grep -F "<text>" <file>`
Fixed string, no regex

`grep -E "<a>|<b>" <file>`
Extended regex, alternation

`grep -e <a> -e <b> <file>`
Several patterns

`grep -f <patterns> <file>`
Read patterns from a file

`grep -P "<regex>" <file>`
Perl-style regex

`grep -m <n> "<pattern>" <file>`
Stop after n matching lines

`grep -e "-v" <file>`
Search for a pattern starting with -

## OUTPUT

`grep -n "<pattern>" <file>`
Prefix line numbers

## OUTPUT (cont.)

`grep -c "<pattern>" <file>`
Count matching lines

`grep -l "<pattern>" <files>`
List files with a match

`grep -L "<pattern>" <files>`
List files without a match

`grep -o "<regex>" <file>`
Print only the matched text

`grep -q "<pattern>" <file>`
Silent, exit status only

`grep -H "<pattern>" <file>`
Always show file name

`grep -h "<pattern>" <files>`
Never show file name

`grep --color=always "<pattern>" <file>`
Force highlighting

## CONTEXT

`grep -A <n> "<pattern>" <file>`
n lines after each match

`grep -B <n> "<pattern>" <file>`
n lines before each match

`grep -C <n> "<pattern>" <file>`
n lines on both sides

### Output prefixes

| Sign | Meaning |
|---|---|
| `file:12:` | matching line |
| `file-12-` | context line |
| `--` | gap between groups |

## FILES & RECURSION

`grep -r "<pattern>" <dir>`
Search a directory tree

`grep -R "<pattern>" <dir>`
Recurse and follow all symlinks

`grep -r --include="*.py" "<p>" <dir>`
Only files matching a glob

`grep -r --exclude="*.min.js" "<p>" .`
Skip files matching a glob

`grep -r --exclude-dir=<dir> "<p>" .`
Skip a directory

`grep -I "<pattern>" <files>`
Skip binary files

## FILES & RECURSION (cont.)

git grep is faster inside a repository and skips ignored files.

## REGEX SYNTAX

### Basic building blocks

| Syntax | Matches |
|---|---|
| `.` | any one character |
| `*` | zero or more of previous |
| `^ $` | start / end of line |
| `[abc] [^abc]` | one of / none of |
| `[a-z]` | a range |
| `[[:digit:]]` | POSIX class |
| `\< \>` | word start / end |
| `\b` | word boundary |

### ERE (-E) additions

| Syntax | Matches |
|---|---|
| `a\|b` | a or b |
| `( )` | group |
| `+` | one or more |
| `?` | zero or one |
| `{n,m}` | n to m repeats |
| `\1` | back-reference to group 1 |

**BRE vs ERE**
In plain grep (BRE) write \| \+ \? \( \) \{ \} for the special forms. -E drops the backslashes.

`grep -P "\d+" <file>`
PCRE shorthand classes

`grep -oP "<before>\K<value>" <file>`
Print only text after a marker

## RECIPES

### Find a symbol in source

```
grep -rnw --include="*.c"
"<name>" src/
# add --exclude-dir=build to skip
output
```

### Filter a process list

```
ps aux | grep "[n]ginx"
# [n] stops grep from matching
itself
```

### Count unique matches

```
grep -oE "[0-9]+\.[0-9]+\.[0-9]+
\.[0-9]+" access.log |
  sort | uniq -c | sort -rn |
head
```

### Use the result in a script

```
if grep -q "^<user>:" /etc/
passwd; then
  echo "user exists"
fi
```