# Universal Ctags Source code indexer

**START HERE**

`ctags -R`
Index the tree into ./tags

`ctags -R src/ lib/`
Index chosen directories

`ctags -R --exclude=node_modules .`
Skip a directory

`ctags --list-languages`
Supported languages

`ctags -x <file>`
Print tags, no tags file

`ctags -a <file>`
Append to existing tags

`ctags -R --fields=+n`
Add line numbers

`ctags -e -R`
Write Emacs TAGS

**Source files** many languages → `ctags -R` → **tags file** name, file, location → `editor reads` → **Jump** Ctrl+] in Vim

## CONCEPTS

ctags scans source files and writes a sorted index (the tags file) mapping each name to its file and location. Editors read it to jump to definitions without a language server.

Re-run ctags after larger edits: the index is a snapshot and does not update itself.

Universal Ctags (ctags) is the maintained successor of Exuberant Ctags; some systems ship the old one under the same name. Check ctags --version.

## GENERATE TAGS

`ctags -R`
Index current directory recursively

`ctags -R <dir>...`
Index only these directories

`ctags <file>...`
Index specific files

`ctags -f <tagfile> -R`
Write to another file name

`ctags -f - <file>`
Write tags to stdout

`ctags -a <file>`
Append instead of overwrite

`ctags -L <list>`
Read file names from a list

`find . -name '*.c' | ctags -L -`
Read file names from stdin

`ctags --tag-relative=yes`
Store paths relative to tags file

`ctags --sort=no`
Keep tags unsorted
*Vim's binary search expects sorted tags; use foldcase with 'tagcase'.*

`ctags --sort=foldcase`
Sort ignoring case

`ctags --totals`
Print counts when finished

`ctags -V`
Verbose: show what is read

## SELECT FILES AND LANGUAGES

`ctags --exclude=<pattern>`
Skip matching files or dirs

`ctags --languages=<list>`
Only parse these languages

## SELECT FILES AND LANGUAGES (cont.)

`ctags --languages=-<lang>`
Disable one language

`ctags --languages=+<lang>`
Enable one more language

`ctags --exclude=@<file>`
Read exclude patterns from a file

`ctags --exclude-exception=<pattern>`
Re-include an excluded path

`ctags --list-languages`
List supported languages

`ctags --list-maps`
Show extension to language map

`ctags --map-<lang>=+<ext>`
Add an extension to a language

`ctags --langmap=<lang>:<ext>`
Set the extensions of a language

`ctags --print-language <file>`
Show the parser chosen for a file

`ctags -G`
Guess language from file content

`ctags --links=no`
Do not follow symlinks

## TAG CONTENT

`ctags --list-kinds=<lang>`
List kinds of a language

`ctags --kinds-<lang>=+<kinds>`
Add kinds, e.g. --kinds-c=+p

`ctags --kinds-<lang>=-<kinds>`
Drop kinds

`ctags --fields=+n`
Add line numbers

`ctags --fields=+S`
Add function signatures

`ctags --fields=+l`
Add source language

`ctags --fields=+K`
Show kind as a long name

`ctags --fields=*`
Enable all fields

`ctags --extras=+q`
Add qualified names (Class.member)

`ctags --extras=+f`
Add a tag per file name

## TAG CONTENT (cont.)

`ctags --extras=+r`
Add reference tags (includes)

`ctags --list-fields`
List available fields

`ctags --list-extras`
List available extras

`ctags -n`
Locate by line number, not pattern

`ctags --pattern-length-limit=0`
Never truncate search patterns

## OUTPUT FORMATS

`ctags -x <file>`
Cross-reference table on stdout

`ctags --output-format=json`
One JSON object per tag
*Needs a build with libjansson; check ctags --list-features.*

`ctags -e`
Emacs etags format (TAGS)

`ctags --output-format=u-ctags`
Default format

`ctags --list-output-formats`
List supported formats

`ctags --list-features`
List compiled-in features

`ctags --version`
Show the version

## CONFIG FILES

`ctags --options=<path>`
Read options from file or directory

`ctags --options=NONE`
Ignore preloaded config files
*Must be the first option.*

`ctags --options-maybe=<path>`
Like --options, no error if missing

`~/.ctags.d/*.ctags`
Per-user options, one per line

`.ctags.d/*.ctags`
Per-project options

`ctags --regex-<lang>=/<re>/<name>/<kind>/`
Add a regex tag rule

`ctags --langdef=<name>`
Define a new language

## CONFIG FILES (cont.)

`ctags -I <ident>`
Ignore a C macro identifier

`ctags -D <macro>=<def>`
Define a C preprocessor macro

## USE IN VIM

Vim looks for ./tags then the parent directories when 'tags' contains ./tags;, so run ctags -R at the project root.

`Ctrl+]`
Jump to tag under cursor

`Ctrl+T`
Jump back up the tag stack

`:tag <name>`
Jump to a tag by name

`g]`
List matches for tag under cursor

`:tselect <name>`
Choose among matching tags

`:tnext / :tprev`
Next or previous matching tag

`Ctrl+W ]`
Open tag in a split

`:tags`
Show the tag stack

`vim -t <name>`
Start Vim at a tag

`:set tags=./tags;,tags`
Search parent directories for tags

## RECIPES

**Project index, skipping noise**

```
ctags -R \
  --exclude=.git --
exclude=node_modules \
  --exclude=build --
exclude='*.min.js' .
```

**Per-project config (.ctags.d/main.ctags)**

```
# one option per line
--recurse=yes
--exclude=.git
--exclude=node_modules
--languages=-JSON
```

**Find definitions from the shell**

```
ctags -R --fields=+n .
grep -P '^main\t' tags
ctags -x --kinds-c=f src/main.c
```

**Tags of git-tracked files only**

```
git ls-files | ctags -L - --
fields=+nS
# fast index of just what git
tracks
```