ripgrep Fast recursive search

CONCEPTS

rg skips ignored, hidden and binary files by default. If a match is missing, try rg -uu before blaming the pattern.

Walk tree from . or <path>
↓ skip ignored
Filter globs, types, hidden
↓ skip binary
Match regex per line
↓ format
Print path:line:text
What happens to each file

ripgrep never edits files: --replace only changes what is printed.

SEARCHING

`rg <pattern> [<path>...]`
Search paths recursively (default .)

`rg -e <pat1> -e <pat2>`
Match any of several patterns

`rg -i <pattern>`
Case-insensitive

`rg -S <pattern>`
Smart case: ignore case unless uppercase used

`rg -F <text>`
Fixed string, no regex

`rg -w <word>`
Match whole words only

SEARCHING (cont.)

`rg -v <pattern>`
Show non-matching lines

`rg -m <n> <pattern>`
Stop after n matches per file

FILTERING FILES

`rg -g '<glob>' <pattern>`
Only paths matching the glob

`rg -g '!<glob>' <pattern>`
Exclude paths matching the glob

`rg -t <type> <pattern>`
Only files of a type (py, rust, md...)

`rg -T <type> <pattern>`
Exclude a file type

`rg --type-list`
List known types and their globs

`rg --max-depth <n> <pattern>`
Limit directory depth

`rg --files`
List files that would be searched

`rg --files -g '<glob>'`
Fast file finder by glob

IGNORE RULES

`rg --hidden <pattern>`
Search hidden files and dirs

IGNORE RULES (cont.)

`rg --no-ignore <pattern>`
Ignore .gitignore, .ignore and .rgignore

`rg -u <pattern>`
Stop honouring ignore files

`rg -uu <pattern>`
Also search hidden files

`rg -uuu <pattern>`
Also search binary files

`rg -L <pattern>`
Follow symlinks

`rg --debug <pattern>`
Show why files are skipped

`!<dir>/`
Whitelist an ignored path in a .ignore file

OUTPUT

`rg -n <pattern>`
Show line numbers

`rg --vimgrep <pattern>`
path:line:col format for editors

`rg -o <pattern>`
Print only the matched text

`rg -l <pattern>`
List files with matches

`rg --files-without-match <pattern>`
List files with no match

`rg -c <pattern>`
Count matching lines per file

OUTPUT (cont.)

`rg -A <n> / -B <n> / -C <n>`
Lines after, before, around a match

`rg -M <n> <pattern>`
Truncate lines longer than n columns

`rg --sort path <pattern>`
Sort by path (disables parallelism)

`rg --json <pattern>`
Machine-readable JSON lines

`rg -0 -l <pattern>`
NUL-separate file names for xargs -0

REGEX AND REPLACE

`rg -r '<text>' <pattern>`
Replace matches in the output only

`rg 'a(\w+)' -r 'b$1'`
Use capture group $1 in replacement

`rg -o -r '$1' '<pat>(<grp>)'`
Extract just a capture group

`rg -U '<a>\n<b>'`
Multiline match across lines

`rg -P '<pattern>'`
PCRE2 engine: lookaround, backrefs

CONFIG AND TOOLING

`export RIPGREP_CONFIG_PATH=<file>`
Point rg at a config file

`rg --pre <cmd> <pattern>`
Run a preprocessor per file (pdf, docs)

`rg -z <pattern>`
Search inside compressed files

`rg --version`
Show version and build features

RECIPES

Files that mention a symbol

```
rg -l 'parse_args' -t py
rg -c 'parse_args' -t py |
sort -t: -k2 -nr
```

Persistent defaults

```
# ~/.ripgreprc
--smart-case
--max-columns=150
--glob=!.git/
# in your shell profile
export RIPGREP_CONFIG_PATH=
$HOME/.ripgreprc
```

Preview a refactor

```
rg 'old_name' -r 'new_name'
--passthru <file>
# apply with sed after
review
rg -l 'old_name' -0 | xargs
-0 sed -i 's/old_name/
new_name/g'
```