NAME

git-check-ignore - Debug gitignore / exclude files

SYNOPSIS

```
git check-ignore [<options>] <pathname>...
git check-ignore [<options>] --stdin
```

DESCRIPTION

For each pathname given via the command-line or from a file via --stdin, check whether the file is excluded by .gitignore (or other input files to the exclude mechanism) and output the path if it is excluded.

By default, tracked files are not shown at all since they are not subject to exclude rules; but see '--no-index'.

OPTIONS

-q, --quiet
Don't output anything, just set exit status. This is only valid with a single pathname.

-v, --verbose
Instead of printing the paths that are excluded, for each path that matches an exclude pattern, print the exclude pattern together with the path. (Matching an exclude pattern usually means the path is excluded, but if the pattern begins with ! then it is a negated pattern and matching it means the path is NOT excluded.)

For precedence rules within and between exclude sources, see gitignore(5).

--stdin
Read pathnames from the standard input, one per line, instead of from the command-line.

-z
The output format is modified to be machine-parsable (see below). If --stdin is also given, input paths are separated with a NUL character instead of a linefeed character.

-n, --non-matching
Show given paths which don't match any pattern. This only makes sense when --verbose is enabled, otherwise it would not be possible to distinguish between paths which match a pattern and those which don't.

--no-index
Don't look in the index when undertaking the checks. This can be used to debug why a path became tracked by e.g. git add . and was not ignored by the rules as expected by the user or when developing patterns including negation to match a path previously added with git add -f.

OUTPUT

By default, any of the given pathnames which match an ignore pattern will be output, one per line. If no pattern matches a given path, nothing will be output for that path; this means that path will not be ignored.

If --verbose is specified, the output is a series of lines of the form:

<source> <COLON> <linenum> <COLON> <pattern> <HT> <pathname>

<pathname> is the path of a file being queried, <pattern> is the matching pattern, <source> is the pattern's source file, and <linenum> is the line number of the pattern within that source. If the pattern

contained a ! prefix or / suffix, it will be preserved in the output. <source> will be an absolute path when referring to the file configured by core.excludesFile, or relative to the repository root when referring to .git/info/exclude or a per-directory exclude file.

If -z is specified, the pathnames in the output are delimited by the null character; if --verbose is also specified then null characters are also used instead of colons and hard tabs:

<source> <NULL> <linenum> <NULL> <pattern> <NULL> <pathname> <NULL>

If -n or --non-matching are specified, non-matching pathnames will also be output, in which case all fields in each output record except for <pathname> will be empty. This can be useful when running non-interactively, so that files can be incrementally streamed to STDIN of a long-running check-ignore process, and for each of these files, STDOUT will indicate whether that file matched a pattern or not. (Without this option, it would be impossible to tell whether the absence of output for a given file meant that it didn't match any pattern, or that the output hadn't been generated yet.)

Buffering happens as documented under the GIT_FLUSH option in git(1). The caller is responsible for avoiding deadlocks caused by overfilling an input buffer or reading from an empty output buffer.

EXIT STATUS

0
: One or more of the provided paths is ignored.

1
: None of the provided paths are ignored.

128
: A fatal error was encountered.

SEE ALSO

gitignore(5) git-config(1) git-ls-files(1)

GIT

Part of the git(1) suite