gitignore(5) Manual Page

NAME

gitignore - Specifies intentionally untracked files to ignore

SYNOPSIS

$GIT_DIR/info/exclude, .gitignore

DESCRIPTION

A `gitignore` file specifies intentionally untracked files that git should ignore. Files already tracked by git are not affected; see the NOTES below for details.

Each line in a `gitignore` file specifies a pattern. When deciding whether to ignore a path, git normally checks `gitignore` patterns from multiple sources, with the following order of precedence, from highest to lowest (within one level of precedence, the last matching pattern decides the outcome):

- Patterns read from the command line for those commands that support them.
- Patterns read from a `.gitignore` file in the same directory as the path, or in any parent directory, with patterns in the higher level files (up to the toplevel of the work tree) being overridden by those in lower level files down to the directory containing the file. These patterns match relative to the location of the `.gitignore` file. A project normally includes such `.gitignore` files in its repository, containing patterns for files generated as part of the project build.
- Patterns read from `$GIT_DIR/info/exclude`.
- Patterns read from the file specified by the configuration variable `core.excludesfile`.

Which file to place a pattern in depends on how the pattern is meant to be used.

- Patterns which should be version-controlled and distributed to other repositories via clone (i.e., files that all developers will want to ignore) should go into a `.gitignore` file.
- Patterns which are specific to a particular repository but which do not need to be shared with other related repositories (e.g.,

auxiliary files that live inside the repository but are specific to one user's workflow) should go into the $GIT_DIR/info/exclude file.

- Patterns which a user wants git to ignore in all situations (e.g., backup or temporary files generated by the user's editor of choice) generally go into a file specified by core.excludesfile in the user's ~/.gitconfig. Its default value is $XDG_CONFIG_HOME/git/ignore. If $XDG_CONFIG_HOME is either not set or empty, $HOME/.config/git/ignore is used instead.

The underlying git plumbing tools, such as git ls-files and git read-tree, read gitignore patterns specified by command-line options, or from files specified by command-line options. Higher-level git tools, such as git status and git add, use patterns from the sources specified above.

PATTERN FORMAT

- A blank line matches no files, so it can serve as a separator for readability.
- A line starting with # serves as a comment. Put a backslash ("\") in front of the first hash for patterns that begin with a hash.
- An optional prefix "!" which negates the pattern; any matching file excluded by a previous pattern will become included again. If a negated pattern matches, this will override lower precedence patterns sources. Put a backslash ("\") in front of the first "!" for patterns that begin with a literal "!", for example, "\!important!.txt".
- If the pattern ends with a slash, it is removed for the purpose of the following description, but it would only find a match with a directory. In other words, foo/ will match a directory foo and paths underneath it, but will not match a regular file or a symbolic link foo (this is consistent with the way how pathspec works in general in git).
- If the pattern does not contain a slash /, git treats it as a shell glob pattern and checks for a match against the pathname relative to the location of the .gitignore file (relative to the toplevel of the work tree if not from a .gitignore file).
- Otherwise, git treats the pattern as a shell glob suitable for consumption by fnmatch(3) with the FNM_PATHNAME flag: wildcards in the pattern will not match a / in the pathname. For example, "Documentation/*.html" matches "Documentation/git.html" but not "Documentation/ppc/ppc.html" or "tools/perf/Documentation/perf.html".
- A leading slash matches the beginning of the pathname. For example, "/*.c" matches "cat-file.c" but not "mozilla-sha1/sha1.c".

NOTES

The purpose of gitignore files is to ensure that certain files not tracked by git remain untracked.

To ignore uncommitted changes in a file that is already tracked, use `git update-index --assume-unchanged`.

To stop tracking a file that is currently tracked, use `git rm --cached`.

EXAMPLES

```
$ git status
[...]
# Untracked files:
[...]
#       Documentation/foo.html
#       Documentation/gitignore.html
#       file.o
#       lib.a
#       src/internal.o
[...]
$ cat .git/info/exclude
# ignore objects and archives, anywhere in the tree.
*.[oa]
$ cat Documentation/.gitignore
# ignore generated html files,
*.html
# except foo.html which is maintained by hand
!foo.html
$ git status
[...]
# Untracked files:
[...]
#       Documentation/foo.html
[...]
```

Another example:

```
$ cat .gitignore
vmlinux*
$ ls arch/foo/kernel/vm*
arch/foo/kernel/vmlinux.lds.S
$ echo '!/vmlinux*' >arch/foo/kernel/.gitignore
```

The second .gitignore prevents git from ignoring `arch/foo/kernel/vmlinux.lds.S`.

SEE ALSO

git-rm(1), git-update-index(1), gitrepository-layout(5)

GIT

Part of the git(1) suite

Last updated 2013-02-15 18:10:50 UTC