NAME

gitignore - Specifies intentionally untracked files to ignore

SYNOPSIS

$XDG_CONFIG_HOME/git/ignore, $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):

- read from the command line for those commands that support them.

- read from a **.gitignore** file in the same directory as the path, or in any parent directory (up to the top-level of the working tree), with patterns in the higher level files 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.

- read from **$GIT_DIR/info/exclude**.

- 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.

- 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.

- 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.

- 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

- blank line matches no files, so it can serve as a separator for readability.

- line starting with # serves as a comment. Put a backslash ("\") in front of the first hash for patterns that begin with a hash.

- spaces are ignored unless they are quoted with backslash ("\").

- optional prefix "**!**" which negates the pattern; any matching file excluded by a previous pattern will become included again. It is not possible to re-include a file if a parent directory of that file is excluded. Git doesn't list excluded directories for performance reasons, so any patterns on contained files have no effect, no matter where they are defined. Put a backslash ("\") in front of the first "**!**" for patterns that begin with a literal "**!**", for example, "**\!important!.txt**".

- slash "/" is used as the directory separator. Separators may occur at the beginning, middle or end of the **.gitignore** search pattern.

- there is a separator at the beginning or middle (or both) of the pattern, then the pattern is relative to the directory level of the particular **.gitignore** file itself. Otherwise the pattern may also match at any level below the **.gitignore** level.

- there is a separator at the end of the pattern then the pattern will only match directories, otherwise the pattern can match both files and directories.

- example, a pattern **doc/frotz/** matches **doc/frotz** directory, but not **a/doc/frotz** directory; however **frotz/** matches **frotz** and **a/frotz** that is a directory (all paths are relative from the **.gitignore** file).

- asterisk "*****" matches anything except a slash. The character "**?**" matches any one character except "/". The range notation, e.g. **[a-zA-Z]**, can be used to match one of the characters in a range. See fnmatch(3) and the FNM_PATHNAME flag for a more detailed description.

Two consecutive asterisks ("******") in patterns matched against full pathname may have special meaning:

- leading "******" followed by a slash means match in all directories. For example, "****/foo**" matches file or directory "**foo**" anywhere, the same as pattern "**foo**". "****/foo/bar**" matches file or directory "**bar**" anywhere that is directly under directory "**foo**".

- trailing "**/****" matches everything inside. For example, "**abc/****" matches all files inside directory "**abc**", relative to the location of the **.gitignore** file, with infinite depth.

- slash followed by two consecutive asterisks then a slash matches zero or more directories. For example, "**a/**/b**" matches "**a/b**", "**a/x/b**", "**a/x/y/b**" and so on.

- consecutive asterisks are considered regular asterisks and will match according to the previous rules.

CONFIGURATION

The optional configuration variable **core.excludesFile** indicates a path to a file containing patterns of file names to exclude, similar to **$GIT_DIR/info/exclude**. Patterns in the exclude file are used in addition to those in **$GIT_DIR/info/exclude**.

NOTES

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

To stop tracking a file that is currently tracked, use *git rm --cached* to remove the file from the index. The filename can then be added to the **.gitignore** file to stop the file from being reintroduced in later commits.

Git does not follow symbolic links when accessing a **.gitignore** file in the working tree. This keeps

behavior consistent when the file is accessed from the index or a tree versus from the filesystem.

EXAMPLES

- pattern **hello.*** matches any file or directory whose name begins with **hello.**. If one wants to restrict this only to the directory and not in its subdirectories, one can prepend the pattern with a slash, i.e. **/hello.***; the pattern now matches **hello.txt**, **hello.c** but not **a/hello.java**.

- pattern **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)

- pattern **doc/frotz** and **/doc/frotz** have the same effect in any **.gitignore** file. In other words, a leading slash is not relevant if there is already a middle slash in the pattern.

- pattern **foo/***, matches **foo/test.json** (a regular file), **foo/bar** (a directory), but it does not match **foo/bar/hello.c** (a regular file), as the asterisk in the pattern does not match **bar/hello.c** which has a slash in it.

```
$ 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**.

Example to exclude everything except a specific directory **foo/bar** (note the **/*** - without the slash, the wildcard would also exclude everything within **foo/bar**):

```
$ cat .gitignore
# exclude everything except directory foo/bar
/*
!/foo
/foo/*
!/foo/bar
```

SEE ALSO

git-rm(1), **gitrepository-layout**(5), **git-check-ignore**(1)

GIT

Part of the **git**(1) suite