NAME

git-clean − Remove untracked files from the working tree

SYNOPSIS

git clean [−d] [−f] [−i] [−n] [−q] [−e <pattern>] [−x | −X] [−−] <path>...

DESCRIPTION

Cleans the working tree by recursively removing files that are not under version control, starting from the current directory.

Normally, only files unknown to Git are removed, but if the **−x** option is specified, ignored files are also removed. This can, for example, be useful to remove all build products.

If any optional **<path>...** arguments are given, only those paths are affected.

OPTIONS

−d
: Normally, when no <path> is specified, git clean will not recurse into untracked directories to avoid removing too much. Specify −d to have it recurse into such directories as well. If any paths are specified, −d is irrelevant; all untracked files matching the specified paths (with exceptions for nested git directories mentioned under **−−force**) will be removed.

−f, −−force
: If the Git configuration variable clean.requireForce is not set to false, *git clean* will refuse to delete files or directories unless given −f or −i. Git will refuse to modify untracked nested git repositories (directories with a .git subdirectory) unless a second −f is given.

−i, −−interactive
: Show what would be done and clean files interactively. See “Interactive mode” for details.

−n, −−dry−run
: Don’t actually remove anything, just show what would be done.

−q, −−quiet
: Be quiet, only report errors, but not the files that are successfully removed.

−e <pattern>, −−exclude=<pattern>
: Use the given exclude pattern in addition to the standard ignore rules (see **gitignore**(5)).

−x
: Don’t use the standard ignore rules (see **gitignore**(5)), but still use the ignore rules given with **−e** options from the command line. This allows removing all untracked files, including build products. This can be used (possibly in conjunction with *git restore* or *git reset*) to create a pristine working directory to test a clean build.

−X
: Remove only files ignored by Git. This may be useful to rebuild everything from scratch, but keep manually created files.

INTERACTIVE MODE

When the command enters the interactive mode, it shows the files and directories to be cleaned, and goes into its interactive command loop.

The command loop shows the list of subcommands available, and gives a prompt "What now> ". In general, when the prompt ends with a single >, you can pick only one of the choices given and type return, like this:

```
*** Commands ***
   1: clean                2: filter by pattern    3: select by numbers
   4: ask each             5: quit                 6: help
What now> 1
```

You also could say **c** or **clean** above as long as the choice is unique.

The main command loop has 6 subcommands.

clean
: Start cleaning files and directories, and then quit.

filter by pattern
: This shows the files and directories to be deleted and issues an "Input ignore patterns>>" prompt. You can input space−separated patterns to exclude files and directories from deletion. E.g. "*.c *.h" will excludes files end with ".c" and ".h" from deletion. When you are satisfied with the filtered result, press ENTER (empty) back to the main menu.

select by numbers
: This shows the files and directories to be deleted and issues an "Select items to delete>>" prompt. When the prompt ends with double >> like this, you can make more than one selection, concatenated with whitespace or comma. Also you can say ranges. E.g. "2−5 7,9" to choose 2,3,4,5,7,9 from the list. If the second number in a range is omitted, all remaining items are selected. E.g. "7−" to choose 7,8,9 from the list. You can say * to choose everything. Also when you are satisfied with the filtered result, press ENTER (empty) back to the main menu.

ask each
: This will start to clean, and you must confirm one by one in order to delete items. Please note that this action is not as efficient as the above two actions.

quit
: This lets you quit without do cleaning.

help
: Show brief usage of interactive git−clean.

SEE ALSO

gitignore(5)

GIT

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