NAME

gitcli − Git command line interface and conventions

SYNOPSIS

gitcli

DESCRIPTION

This manual describes the convention used throughout Git CLI.

Many commands take revisions (most often "commits", but sometimes "tree−ish", depending on the context and command) and paths as their arguments. Here are the rules:

- Revisions come first and then paths. E.g. in git diff v1.0 v2.0 arch/x86 include/asm−x86, v1.0 and v2.0 are revisions and arch/x86 and include/asm−x86 are paths.
- When an argument can be misunderstood as either a revision or a path, they can be disambiguated by placing −− between them. E.g. git diff −− HEAD is, "I have a file called HEAD in my work tree. Please show changes between the version I staged in the index and what I have in the work tree for that file". not "show difference between the HEAD commit and the work tree as a whole". You can say git diff HEAD −− to ask for the latter.
- Without disambiguating −−, Git makes a reasonable guess, but errors out and asking you to disambiguate when ambiguous. E.g. if you have a file called HEAD in your work tree, git diff HEAD is ambiguous, and you have to say either git diff HEAD −− or git diff −− HEAD to disambiguate.

  When writing a script that is expected to handle random user−input, it is a good practice to make it explicit which arguments are which by placing disambiguating −− at appropriate places.
- Many commands allow wildcards in paths, but you need to protect them from getting globbed by the shell. These two mean different things:

  ```
  $ git checkout −− *.c
  $ git checkout −− \*.c
  ```

  The former lets your shell expand the fileglob, and you are asking the dot−C files in your working tree to be overwritten with the version in the index. The latter passes the *.c to Git, and you are asking the paths in the index that match the pattern to be checked out to your working tree. After running git add hello.c; rm hello.c, you will *not* see hello.c in your working tree with the former, but with the latter you will.

Here are the rules regarding the "flags" that you should follow when you are scripting Git:

- it’s preferred to use the non dashed form of Git commands, which means that you should prefer git foo to git−foo.
- splitting short options to separate words (prefer git foo −a −b to git foo −ab, the latter may not even work).
- when a command line option takes an argument, use the *sticked* form. In other words, write git foo −oArg instead of git foo −o Arg for short options, and git foo −−long−opt=Arg instead of git foo −−long−opt Arg for long options. An option that takes optional option−argument must be written in the *sticked* form.
- when you give a revision parameter to a command, make sure the parameter is not ambiguous with a name of a file in the work tree. E.g. do not write git log −1 HEAD but write git log −1 HEAD −−; the former will not work if you happen to have a file called HEAD in the work tree.
- many commands allow a long option "−−option" to be abbreviated only to their unique prefix (e.g. if there is no other option whose name begins with "opt", you may be able to spell "−−opt" to invoke the "−−option" flag), but you should fully spell them out when writing your scripts; later

versions of Git may introduce a new option whose name shares the same prefix, e.g. "−−optimize", to make a short prefix that used to be unique no longer unique.

ENHANCED OPTION PARSER

From the Git 1.5.4 series and further, many Git commands (not all of them at the time of the writing though) come with an enhanced option parser.

Here is a list of the facilities provided by this option parser.

Magic Options

Commands which have the enhanced option parser activated all understand a couple of magic command line options:

−h
: gives a pretty printed usage of the command.

```
$ git describe −h
usage: git describe [options] <committish>*
  or: git describe [options] −−dirty

    −−contains          find the tag that comes after the commit
    −−debug             debug search strategy on stderr
    −−all               use any ref
    −−tags              use any tag, even unannotated
    −−long              always use long format
    −−abbrev[=<n>]      use <n> digits to display SHA−1s
```

−−help−all
: Some Git commands take options that are only used for plumbing or that are deprecated, and such options are hidden from the default usage. This option gives the full list of options.

Negating options

Options with long option names can be negated by prefixing −−no−. For example, git branch has the option −−track which is *on* by default. You can use −−no−track to override that behaviour. The same goes for −−color and −−no−color.

Aggregating short options

Commands that support the enhanced option parser allow you to aggregate short options. This means that you can for example use git rm −rf or git clean −fdx.

Abbreviating long options

Commands that support the enhanced option parser accepts unique prefix of a long option as if it is fully spelled out, but use this with a caution. For example, git commit −−amen behaves as if you typed git commit −−amend, but that is true only until a later version of Git introduces another option that shares the same prefix, e.g ‘git commit −−amenity" option.

Separating argument from the option

You can write the mandatory option parameter to an option as a separate word on the command line. That means that all the following uses work:

```
$ git foo −−long−opt=Arg
$ git foo −−long−opt Arg
$ git foo −oArg
$ git foo −o Arg
```

However, this is **NOT** allowed for switches with an optional value, where the *sticked* form must be used:

```
$ git describe --abbrev HEAD     # correct
$ git describe --abbrev=10 HEAD  # correct
$ git describe --abbrev 10 HEAD  # NOT WHAT YOU MEANT
```

NOTES ON FREQUENTLY CONFUSED OPTIONS

Many commands that can work on files in the working tree and/or in the index can take --cached and/or --index options. Sometimes people incorrectly think that, because the index was originally called cache, these two are synonyms. They are **not** — these two options mean very different things.

- The --cached option is used to ask a command that usually works on files in the working tree to **only** work with the index. For example, git grep, when used without a commit to specify from which commit to look for strings in, usually works on files in the working tree, but with the --cached option, it looks for strings in the index.
- The --index option is used to ask a command that usually works on files in the working tree to **also** affect the index. For example, git stash apply usually merges changes recorded in a stash to the working tree, but with the --index option, it also merges changes to the index as well.

git apply command can be used with --cached and --index (but not at the same time). Usually the command only affects the files in the working tree, but with --index, it patches both the files and their index entries, and with --cached, it modifies only the index entries.

See also **http://marc.info/?l=git&m=116563135620359** and **http://marc.info/?l=git&m=119150393620273** for further information.

GIT

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