



Linux Ubuntu 26.04 Manual Pages on command 'git-apply.1'

\$ man git-apply.1

GIT-APPLY(1)

Git Manual

GIT-APPLY(1)

NAME

git-apply - Apply a patch to files and/or to the index

SYNOPSIS

```
git apply [--stat] [--numstat] [--summary] [--check]
          [--index | --intent-to-add] [--3way] [--ours | --theirs | --union]
          [--apply] [--no-add] [--build-fake-ancestor=<file>] [-R | --reverse]
          [--allow-binary-replacement | --binary] [--reject] [-z]
          [-p<n>] [-C<n>] [--inaccurate-eof] [--recount] [--cached]
          [--ignore-space-change | --ignore-whitespace]
          [--whitespace=(nowarn|warn|fix|error|error-all)]
          [--exclude=<path>] [--include=<path>] [--directory=<root>]
          [--verbose | --quiet] [--unsafe-paths] [--allow-empty] [<patch>...]
```

DESCRIPTION

Reads the supplied diff output (i.e. "a patch") and applies it to files. When running from a subdirectory in a repository, patched paths outside the directory are ignored. With the --index option, the patch is also applied to the index, and with the --cached option, the patch is only applied to the index. Without these options, the command applies the patch only to files, and does not require them to be in a Git repository.

This command applies the patch but does not create a commit. Use git-am(1) to create commits from patches generated by git-format-patch(1) and/or received by email.

OPTIONS

<patch>...

The files to read the patch from. - can be used to read from the standard input.

--stat

Instead of applying the patch, output diffstat for the input. Turns off "apply".

--numstat

Similar to --stat, but shows the number of added and deleted lines in decimal notation and the pathname without abbreviation, to make it more machine friendly. For binary files, outputs two - instead of saying 0 0. Turns off "apply".

--summary

Instead of applying the patch, output a condensed summary of information obtained from git diff extended headers, such as creations, renames, and mode changes. Turns off "apply".

--check

Instead of applying the patch, see if the patch is applicable to the current working tree and/or the index file and detects errors. Turns off "apply".

--index

Apply the patch to both the index and the working tree (or merely check that it would apply cleanly to both if --check is in effect). Note that --index expects index entries and working tree copies for relevant paths to be identical (their contents and metadata such as file mode must match), and will raise an error if they are not, even if the patch would apply cleanly to both the index and the working tree in isolation.

--cached

Apply the patch to just the index, without touching the working tree. If --check is in effect, merely check that it would apply cleanly to the index entry.

-N, --intent-to-add

When applying the patch only to the working tree, mark new files to be added to the index later (see --intent-to-add option in git-add(1)). This option is ignored if --index or --cached are used, and has no effect outside a Git repository. Note that --index could be implied by other options such as --3way.

-3, --3way

Attempt 3-way merge if the patch records the identity of blobs it is supposed to apply to and we have those blobs available locally, possibly leaving the conflict markers in the files in the working tree for the user to resolve. This option implies the --index option unless the --cached option is used, and is incompatible with the --reject option.

When used with the `--cached` option, any conflicts are left at higher stages in the cache.

`--ours`, `--theirs`, `--union`

Instead of leaving conflicts in the file, resolve conflicts favouring our (or their or both) side of the lines. Requires `--3way`.

`--build-fake-ancestor=<file>`

Newer git diff output has embedded index information for each blob to help identify the original version that the patch applies to. When this flag is given, and if the original versions of the blobs are available locally, builds a temporary index containing those blobs.

When a pure mode change is encountered (which has no index information), the information is read from the current index instead.

`-R`, `--reverse`

Apply the patch in reverse.

`--reject`

For atomicity, git apply by default fails the whole patch and does not touch the working tree when some of the hunks do not apply. This option makes it apply the parts of the patch that are applicable, and leave the rejected hunks in corresponding `*.rej` files.

`-Z`

When `--numstat` has been given, do not munge pathnames, but use a NUL-terminated machine-readable format.

Without this option, pathnames with "unusual" characters are quoted as explained for the configuration variable `core.quotePath` (see `git-config(1)`).

`-p<n>`

Remove `<n>` leading path components (separated by slashes) from traditional diff paths.

E.g., with `-p2`, a patch against `a/dir/file` will be applied directly to file. The default is 1.

`-C<n>`

Ensure at least `<n>` lines of surrounding context match before and after each change.

When fewer lines of surrounding context exist they all must match. By default no context is ever ignored.

`--unidiff-zero`

By default, git apply expects that the patch being applied is a unified diff with at

least one line of context. This provides good safety measures, but breaks down when applying a diff generated with `--unified=0`. To bypass these checks use `--unidiff-zero`.

Note, for the reasons stated above, the usage of context-free patches is discouraged.

`--apply`

If you use any of the options marked "Turns off apply" above, git apply reads and outputs the requested information without actually applying the patch. Give this flag after those flags to also apply the patch.

`--no-add`

When applying a patch, ignore additions made by the patch. This can be used to extract the common part between two files by first running diff on them and applying the result with this option, which would apply the deletion part but not the addition part.

`--allow-binary-replacement, --binary`

Historically we did not allow binary patch application without an explicit permission from the user, and this flag was the way to do so. Currently, we always allow binary patch application, so this is a no-op.

`--exclude=<path-pattern>`

Don't apply changes to files matching the given path pattern. This can be useful when importing patchsets, where you want to exclude certain files or directories.

`--include=<path-pattern>`

Apply changes to files matching the given path pattern. This can be useful when importing patchsets, where you want to include certain files or directories.

When `--exclude` and `--include` patterns are used, they are examined in the order they appear on the command line, and the first match determines if a patch to each path is used. A patch to a path that does not match any include/exclude pattern is used by default if there is no include pattern on the command line, and ignored if there is any include pattern.

`--ignore-space-change, --ignore-whitespace`

When applying a patch, ignore changes in whitespace in context lines if necessary. Context lines will preserve their whitespace, and they will not undergo whitespace fixing regardless of the value of the `--whitespace` option. New lines will still be fixed, though.

`--whitespace=<action>`

When applying a patch, detect a new or modified line that has whitespace errors. What

are considered whitespace errors is controlled by `core.whitespace` configuration. By default, trailing whitespaces (including lines that solely consist of whitespaces) and a space character that is immediately followed by a tab character inside the initial indent of the line are considered whitespace errors.

By default, the command outputs warning messages but applies the patch. When `git-apply` is used for statistics and not applying a patch, it defaults to `nowarn`.

You can use different `<action>` values to control this behavior:

- ? `nowarn` turns off the trailing whitespace warning.
- ? `warn` outputs warnings for a few such errors, but applies the patch as-is (default).
- ? `fix` outputs warnings for a few such errors, and applies the patch after fixing them (strip is a synonym ? the tool used to consider only trailing whitespace characters as errors, and the fix involved stripping them, but modern Gits do more).
- ? `error` outputs warnings for a few such errors, and refuses to apply the patch.
- ? `error-all` is similar to `error` but shows all errors.

`--inaccurate-eof`

Under certain circumstances, some versions of `diff` do not correctly detect a missing new-line at the end of the file. As a result, patches created by such `diff` programs do not record incomplete lines correctly. This option adds support for applying such patches by working around this bug.

`-v, --verbose`

Report progress to `stderr`. By default, only a message about the current patch being applied will be printed. This option will cause additional information to be reported.

`-q, --quiet`

Suppress `stderr` output. Messages about patch status and progress will not be printed.

`--recount`

Do not trust the line counts in the hunk headers, but infer them by inspecting the patch (e.g. after editing the patch without adjusting the hunk headers appropriately).

`--directory=<root>`

Prepend `<root>` to all filenames. If a `"-p"` argument was also passed, it is applied before prepending the new root.

For example, a patch that talks about updating `a/git-gui.sh` to `b/git-gui.sh` can be applied to the file in the working tree `modules/git-gui/git-gui.sh` by running `git apply`

`--directory=modules/git-gui.`

--unsafe-paths

By default, a patch that affects outside the working area (either a Git controlled working tree, or the current working directory when "git apply" is used as a replacement of GNU patch) is rejected as a mistake (or a mischief).

When git apply is used as a "better GNU patch", the user can pass the --unsafe-paths option to override this safety check. This option has no effect when --index or --cached is in use.

--allow-empty

Don't return an error for patches containing no diff. This includes empty patches and patches with commit text only.

CONFIGURATION

Everything below this line in this section is selectively included from the git-config(1) documentation. The content is the same as what's found there:

apply.ignoreWhitespace

When set to change, tells git apply to ignore changes in whitespace, in the same way as the --ignore-space-change option. When set to one of: no, none, never, false, it tells git apply to respect all whitespace differences. See git-apply(1).

apply.whitespace

Tells git apply how to handle whitespace, in the same way as the --whitespace option.

See git-apply(1).

SUBMODULES

If the patch contains any changes to submodules then git apply treats these changes as follows.

If --index is specified (explicitly or implicitly), then the submodule commits must match the index exactly for the patch to apply. If any of the submodules are checked-out, then these check-outs are completely ignored, i.e., they are not required to be up to date or clean and they are not updated.

If --index is not specified, then the submodule commits in the patch are ignored and only the absence or presence of the corresponding subdirectory is checked and (if possible) updated.

SEE ALSO

git-am(1).

Part of the git(1) suite

Git 2.53.0

03/02/2026

GIT-APPLY(1)