NAME

git-update-index - Register file contents in the working tree to the index

SYNOPSIS

git update-index
[--add] [--remove | --force-remove] [--replace]
[--refresh] [-q] [--unmerged] [--ignore-missing]
[(--cacheinfo <mode>,<object>,<file>)...]
[--chmod=(+|-)x]
[--[no-]assume-unchanged]
[--[no-]skip-worktree]
[--ignore-submodules]
[--really-refresh] [--unresolve] [--again | -g]
[--info-only] [--index-info]
[-z] [--stdin] [--index-version <n>]
[--verbose]
[--] [<file>...]

DESCRIPTION

Modifies the index or directory cache. Each file mentioned is updated into the index and any *unmerged* or *needs updating* state is cleared.

See also **git-add(1)** for a more user-friendly way to do some of the most common operations on the index.

The way *git update-index* handles files it is told about can be modified using the various options:

OPTIONS

--add
: If a specified file isn't in the index already then it's added. Default behaviour is to ignore new files.

--remove
: If a specified file is in the index but is missing then it's removed. Default behavior is to ignore removed file.

--refresh
: Looks at the current index and checks to see if merges or updates are needed by checking stat() information.

-q
: Quiet. If --refresh finds that the index needs an update, the default behavior is to error out. This option makes *git update-index* continue anyway.

--ignore-submodules
: Do not try to update submodules. This option is only respected when passed before --refresh.

--unmerged
: If --refresh finds unmerged changes in the index, the default behavior is to error out. This option makes *git update-index* continue anyway.

--ignore-missing
: Ignores missing files during a --refresh

--cacheinfo <mode>,<object>,<path>, --cacheinfo <mode> <object> <path>
: Directly insert the specified info into the index. For backward compatibility, you can also give these three arguments as three separate parameters, but new users are encouraged to use a single-parameter form.

--index-info
: Read index information from stdin.

--chmod=(+|-)x
: Set the execute permissions on the updated files.

--[no-]assume-unchanged
: When these flags are specified, the object names recorded for the paths are not updated. Instead, these options set and unset the assume unchanged bit for the paths. When the assume unchanged bit is on, Git stops checking the working tree files for possible modifications, so you need to manually unset the bit to tell Git when you change the working tree file. This is sometimes helpful when working with a big project on a filesystem that has very slow lstat(2) system call (e.g. cifs).

  This option can be also used as a coarse file-level mechanism to ignore uncommitted changes in tracked files (akin to what .gitignore does for untracked files). Git will fail (gracefully) in case it needs to modify this file in the index e.g. when merging in a commit; thus, in case the assumed-untracked file is changed upstream, you will need to handle the situation manually.

--really-refresh
: Like *--refresh*, but checks stat information unconditionally, without regard to the assume unchanged setting.

--[no-]skip-worktree
: When one of these flags is specified, the object name recorded for the paths are not updated. Instead, these options set and unset the skip-worktree bit for the paths. See section Skip-worktree bit below for more information.

-g, --again
: Runs *git update-index* itself on the paths whose index entries are different from those from the HEAD commit.

--unresolve
: Restores the *unmerged* or *needs updating* state of a file during a merge if it was cleared by accident.

--info-only
: Do not create objects in the object database for all <file> arguments that follow this flag; just insert their object IDs into the index.

--force-remove
: Remove the file from the index even when the working directory still has such a file. (Implies --remove.)

--replace
: By default, when a file path exists in the index, *git update-index* refuses an attempt to add path/file. Similarly if a file path/file exists, a file path cannot be added. With --replace flag, existing entries that conflict with the entry being added are automatically removed with warning messages.

--stdin
: Instead of taking list of paths from the command line, read list of paths from the standard input. Paths are separated by LF (i.e. one path per line) by default.

--verbose
: Report what is being added and removed from index.

--index-version <n>
: Write the resulting index out in the named on-disk format version. Supported versions are 2, 3 and 4. The current default version is 2 or 3, depending on whether extra features are used, such as git add -N.

  Version 4 performs a simple pathname compression that reduces index size by 30%-50% on large repositories, which results in faster load time. Version 4 is relatively young (first released in in 1.8.0 in October 2012). Other Git implementations such as JGit and libgit2

may not support it yet.

-z
: Only meaningful with --stdin or --index-info; paths are separated with NUL character instead of LF.

--split-index, --no-split-index
: Enable or disable split index mode. If enabled, the index is split into two files, $GIT_DIR/index and $GIT_DIR/sharedindex.<SHA-1>. Changes are accumulated in $GIT_DIR/index while the shared index file contains all index entries stays unchanged. If split-index mode is already enabled and --split-index is given again, all changes in $GIT_DIR/index are pushed back to the shared index file. This mode is designed for very large indexes that take a signficant amount of time to read or write.

--
: Do not interpret any more arguments as options.

<file>
: Files to act on. Note that files beginning with . are discarded. This includes ./file and dir/./file. If you don't want this, then use cleaner names. The same applies to directories ending / and paths with //

USING --REFRESH

--refresh does not calculate a new sha1 file or bring the index up-to-date for mode/content changes. But what it **does** do is to re-match the stat information of a file with the index, so that you can refresh the index for a file that hasn't been changed but where the stat entry is out of date.

For example, you'd want to do this after doing a *git read-tree*, to link up the stat index details with the proper files.

USING --CACHEINFO OR --INFO-ONLY

--cacheinfo is used to register a file that is not in the current working directory. This is useful for minimum-checkout merging.

To pretend you have a file with mode and sha1 at path, say:

$ git update-index --cacheinfo mode sha1 path

--info-only is used to register files without placing them in the object database. This is useful for status-only repositories.

Both *--cacheinfo* and *--info-only* behave similarly: the index is updated but the object database isn't. *--cacheinfo* is useful when the object is in the database but the file isn't available locally. *--info-only* is useful when the file is available, but you do not wish to update the object database.

USING --INDEX-INFO

--index-info is a more powerful mechanism that lets you feed multiple entry definitions from the standard input, and designed specifically for scripts. It can take inputs of three formats:

1. mode SP sha1 TAB path

   The first format is what git-apply --index-info reports, and used to reconstruct a partial tree that is used for phony merge base tree when falling back on 3-way merge.
2. mode SP type SP sha1 TAB path

   The second format is to stuff *git ls-tree* output into the index file.
3. mode SP sha1 SP stage TAB path

   This format is to put higher order stages into the index file and matches *git ls-files --stage* output.

To place a higher stage entry to the index, the path should first be removed by feeding a mode=0 entry for the path, and then feeding necessary input lines in the third format.

For example, starting with this index:

```
$ git ls-files -s
100644 8a1218a1024a212bb3db30becd860315f9f3ac52 0 frotz
```

you can feed the following input to --index-info:

```
$ git update-index --index-info
0 0000000000000000000000000000000000000000 frotz
100644 8a1218a1024a212bb3db30becd860315f9f3ac52 1 frotz
100755 8a1218a1024a212bb3db30becd860315f9f3ac52 2 frotz
```

The first line of the input feeds 0 as the mode to remove the path; the SHA-1 does not matter as long as it is well formatted. Then the second and third line feeds stage 1 and stage 2 entries for that path. After the above, we would end up with this:

```
$ git ls-files -s
100644 8a1218a1024a212bb3db30becd860315f9f3ac52 1 frotz
100755 8a1218a1024a212bb3db30becd860315f9f3ac52 2 frotz
```

USING "ASSUME UNCHANGED" BIT

Many operations in Git depend on your filesystem to have an efficient lstat(2) implementation, so that st_mtime information for working tree files can be cheaply checked to see if the file contents have changed from the version recorded in the index file. Unfortunately, some filesystems have inefficient lstat(2). If your filesystem is one of them, you can set assume unchanged bit to paths you have not changed to cause Git not to do this check. Note that setting this bit on a path does not mean Git will check the contents of the file to see if it has changed — it makes Git to omit any checking and assume it has **not** changed. When you make changes to working tree files, you have to explicitly tell Git about it by dropping assume unchanged bit, either before or after you modify them.

In order to set assume unchanged bit, use --assume-unchanged option. To unset, use --no-assume-unchanged. To see which files have the assume unchanged bit set, use git ls-files -v (see **git-ls-files(1)**).

The command looks at core.ignorestat configuration variable. When this is true, paths updated with git update-index paths... and paths updated with other Git commands that update both index and working tree (e.g. *git apply --index*, *git checkout-index -u*, and *git read-tree -u*) are automatically marked as assume unchanged. Note that assume unchanged bit is **not** set if git update-index --refresh finds the working tree file matches the index (use git update-index --really-refresh if you want to mark them as assume unchanged).

EXAMPLES

To update and refresh only the files already checked out:

```
$ git checkout-index -n -f -a && git update-index --ignore-missing --refresh
```

On an inefficient filesystem with core.ignorestat set

```
$ git update-index --really-refresh (1)
$ git update-index --no-assume-unchanged foo.c (2)
$ git diff --name-only (3)
$ edit foo.c
$ git diff --name-only (4)
M foo.c
$ git update-index foo.c (5)
$ git diff --name-only (6)
$ edit foo.c
$ git diff --name-only (7)
$ git update-index --no-assume-unchanged foo.c (8)
```

$ git diff --name-only **(9)**
M foo.c

1. forces lstat(2) to set assume unchanged bits for paths that match index.
2. mark the path to be edited.
3. this does lstat(2) and finds index matches the path.
4. this does lstat(2) and finds index does **not** match the path.
5. registering the new version to index sets assume unchanged bit.
6. and it is assumed unchanged.
7. even after you edit it.
8. you can tell about the change after the fact.
9. now it checks with lstat(2) and finds it has been changed.

SKIP-WORKTREE BIT

Skip-worktree bit can be defined in one (long) sentence: When reading an entry, if it is marked as skip-worktree, then Git pretends its working directory version is up to date and read the index version instead.

To elaborate, reading means checking for file existence, reading file attributes or file content. The working directory version may be present or absent. If present, its content may match against the index version or not. Writing is not affected by this bit, content safety is still first priority. Note that Git *can* update working directory file, that is marked skip-worktree, if it is safe to do so (i.e. working directory version matches index version)

Although this bit looks similar to assume-unchanged bit, its goal is different from assume-unchanged bit's. Skip-worktree also takes precedence over assume-unchanged bit when both are set.

CONFIGURATION

The command honors core.filemode configuration variable. If your repository is on a filesystem whose executable bits are unreliable, this should be set to *false* (see **git-config(1)**). This causes the command to ignore differences in file modes recorded in the index and the file mode on the filesystem if they differ only on executable bit. On such an unfortunate filesystem, you may need to use *git update-index --chmod=*.

Quite similarly, if core.symlinks configuration variable is set to *false* (see **git-config(1)**), symbolic links are checked out as plain files, and this command does not modify a recorded file mode from symbolic link to regular file.

The command looks at core.ignorestat configuration variable. See *Using assume unchanged bit* section above.

The command also looks at core.trustctime configuration variable. It can be useful when the inode change time is regularly modified by something outside Git (file system crawlers and backup systems use ctime for marking files processed) (see **git-config(1)**).

SEE ALSO

git-config(1), **git-add(1)**, **git-ls-files(1)**

GIT

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