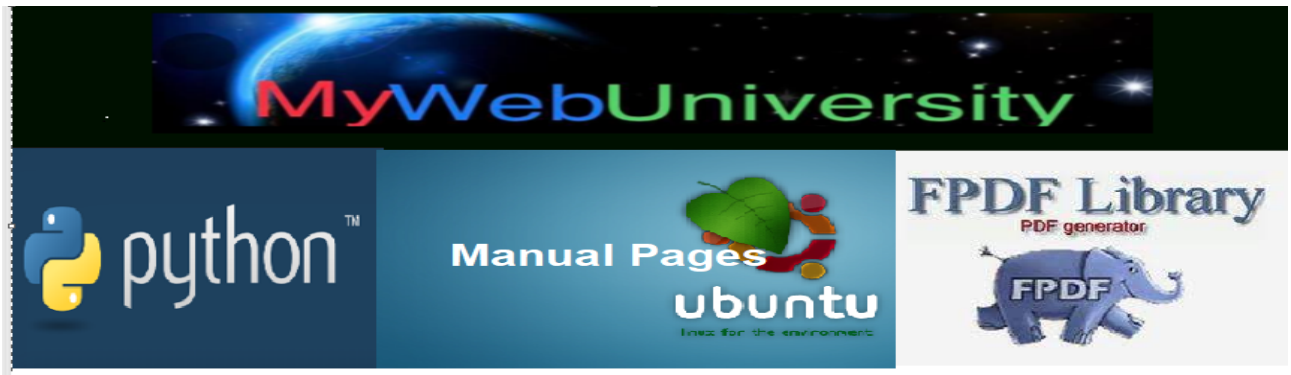

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man git-merge-file.1

GIT-MERGE-FILE(1) Git Manual GIT-MERGE-FILE(1)

NAME

git-merge-file - Run a three-way file merge

SYNOPSIS

git merge-file [-L <current-name> [-L <base-name> [-L <other-name>]]]

[--ours|--theirs|--union] [-p|--stdout] [-q|--quiet] [--marker-size=<n>]

[--[no-]diff3] [--object-id] <current> <base> <other>

DESCRIPTION

Given three files <current>, <base> and <other>, git merge-file incorporates all changes that lead from <base> to <other> into <current>. The result ordinarily goes into <current>. git merge-file is useful for combining separate changes to an original. Suppose <base> is the original, and both <current> and <other> are modifications of <base>, then git merge-file combines both changes.

A conflict occurs if both <current> and <other> have changes in a common segment of lines. If a conflict is found, git merge-file normally outputs a warning and brackets the conflict with lines containing <<<<<<< and >>>>>>> markers. A typical conflict will look like this:

```
<<<<<<< A
lines in file A
=======
lines in file B
>>>>>>> B
```

If there are conflicts, the user should edit the result and delete one of the alternatives. When --ours, --theirs, or --union option is in effect, however, these conflicts are resolved

favouring lines from <current>, lines from <other>, or lines from both respectively. The length of the conflict markers can be given with the --marker-size option.

If --object-id is specified, exactly the same behavior occurs, except that instead of specifying what to merge as files, it is specified as a list of object IDs referring to blobs.

The exit value of this program is negative on error, and the number of conflicts otherwise (truncated to 127 if there are more than that many conflicts). If the merge was clean, the exit value is 0.

git merge-file is designed to be a minimal clone of RCS merge; that is, it implements all of RCS merge's functionality which is needed by git(1).

OPTIONS

--object-id

Specify the contents to merge as blobs in the current repository instead of files. In this case, the operation must take place within a valid repository.

If the -p option is specified, the merged file (including conflicts, if any) goes to standard output as normal; otherwise, the merged file is written to the object store and the object ID of its blob is written to standard output.

-L <label>

This option may be given up to three times, and specifies labels to be used in place of the corresponding file names in conflict reports. That is, git merge-file -L x -L y -L z a b c generates output that looks like it came from files x, y and z instead of from files a, b and c.

-p

Send results to standard output instead of overwriting <current>.

-q

Quiet; do not warn about conflicts.

--diff3

Show conflicts in "diff3" style.

--zdiff3

Show conflicts in "zdiff3" style.

--ours, --theirs, --union

Instead of leaving conflicts in the file, resolve conflicts favouring our (or their or both) side of the lines.

--diff-algorithm={patience|minimal|histogram|myers}

Use a different diff algorithm while merging. The current default is "myers", but selecting more recent algorithm such as "histogram" can help avoid mismerges that occur due to unimportant matching lines (such as braces from distinct functions). See also git-diff(1) --diff-algorithm.

EXAMPLES

git merge-file README.my README README.upstream

combines the changes of README.my and README.upstream since README, tries to merge them and writes the result into README.my.

git merge-file -L a -L b -L c tmp/a123 tmp/b234 tmp/c345

merges tmp/a123 and tmp/c345 with the base tmp/b234, but uses labels a and c instead of tmp/a123 and tmp/c345.

git merge-file -p --object-id abc1234 def567 890abcd

combines the changes of the blob abc1234 and 890abcd since def567, tries to merge them and writes the result to standard output

GIT

Part of the git(1) suite