NAME

　　git-annotate - Annotate file lines with commit information

SYNOPSIS

　　git annotate [<options>] [<rev-opts>] [<rev>] [--] <file>

DESCRIPTION

　　Annotates each line in the given file with information from the commit which introduced the line.
　　Optionally annotates from a given revision.

　　The only difference between this command and **git-blame**(1) is that they use slightly different output
　　formats, and this command exists only for backward compatibility to support existing scripts, and
　　provide a more familiar command name for people coming from other SCM systems.

OPTIONS

　　-b

　　　　Show blank SHA-1 for boundary commits. This can also be controlled via the
　　　　blame.blankBoundary config option.

　　--root

　　　　Do not treat root commits as boundaries. This can also be controlled via the **blame.showRoot**
　　　　config option.

　　--show-stats

　　　　Include additional statistics at the end of blame output.

　　-L <start>,<end>, -L :<funcname>

　　　　Annotate only the line range given by *<start>,<end>*, or by the function name regex *<funcname>*.
　　　　May be specified multiple times. Overlapping ranges are allowed.

　　　　<start> and *<end>* are optional. **-L <start>** or **-L <start>,** spans from *<start>* to end of file. **-L
　　　　,<end>** spans from start of file to *<end>*.

　　　　<start> and *<end>* can take one of these forms:

　　　　⊕

　　　　If *<start>* or *<end>* is a number, it specifies an absolute line number (lines count from 1).

⊕

This form will use the first line matching the given POSIX regex. If *<start>* is a regex, it will search from the end of the previous **-L** range, if any, otherwise from the start of file. If *<start>* is **^/regex/**, it will search from the start of file. If *<end>* is a regex, it will search starting at the line given by *<start>*.

⊕

or -offset

This is only valid for *<end>* and will specify a number of lines before or after the line given by *<start>*.

If **:<funcname>** is given in place of *<start>* and *<end>*, it is a regular expression that denotes the range from the first funcname line that matches *<funcname>*, up to the next funcname line. **:<funcname>** searches from the end of the previous **-L** range, if any, otherwise from the start of file. **^:<funcname>** searches from the start of file. The function names are determined in the same way as **git diff** works out patch hunk headers (see *Defining a custom hunk-header* in **gitattributes**(5)).

-l

Show long rev (Default: off).

-t

Show raw timestamp (Default: off).

-S <revs-file>

Use revisions from revs-file instead of calling **git-rev-list**(1).

--reverse <rev>..<rev>

Walk history forward instead of backward. Instead of showing the revision in which a line appeared, this shows the last revision in which a line has existed. This requires a range of revision like START..END where the path to blame exists in START. **git blame --reverse START** is taken as **git blame --reverse START..HEAD** for convenience.

--first-parent

Follow only the first parent commit upon seeing a merge commit. This option can be used to determine when a line was introduced to a particular integration branch, rather than when it was introduced to the history overall.

-p, --porcelain
    Show in a format designed for machine consumption.

--line-porcelain
    Show the porcelain format, but output commit information for each line, not just the first time a
    commit is referenced. Implies --porcelain.

--incremental
    Show the result incrementally in a format designed for machine consumption.

--encoding=<encoding>
    Specifies the encoding used to output author names and commit summaries. Setting it to **none**
    makes blame output unconverted data. For more information see the discussion about encoding in
    the **git-log**(1) manual page.

--contents <file>
    Annotate using the contents from the named file, starting from <rev> if it is specified, and HEAD
    otherwise. You may specify - to make the command read from the standard input for the file
    contents.

--date <format>
    Specifies the format used to output dates. If --date is not provided, the value of the blame.date
    config variable is used. If the blame.date config variable is also not set, the iso format is used. For
    supported values, see the discussion of the --date option at **git-log**(1).

--[no-]progress
    Progress status is reported on the standard error stream by default when it is attached to a terminal.
    This flag enables progress reporting even if not attached to a terminal. Can't use **--progress**
    together with **--porcelain** or **--incremental**.

-M[<num>]
    Detect moved or copied lines within a file. When a commit moves or copies a block of lines (e.g.
    the original file has A and then B, and the commit changes it to B and then A), the traditional
    blame algorithm notices only half of the movement and typically blames the lines that were
    moved up (i.e. B) to the parent and assigns blame to the lines that were moved down (i.e. A) to the
    child commit. With this option, both groups of lines are blamed on the parent by running extra
    passes of inspection.

    <num> is optional but it is the lower bound on the number of alphanumeric characters that Git
    must detect as moving/copying within a file for it to associate those lines with the parent commit.

The default value is 20.

-C[<num>]

In addition to **-M**, detect lines moved or copied from other files that were modified in the same commit. This is useful when you reorganize your program and move code around across files. When this option is given twice, the command additionally looks for copies from other files in the commit that creates the file. When this option is given three times, the command additionally looks for copies from other files in any commit.

<num> is optional but it is the lower bound on the number of alphanumeric characters that Git must detect as moving/copying between files for it to associate those lines with the parent commit. And the default value is 40. If there are more than one **-C** options given, the <num> argument of the last **-C** will take effect.

--ignore-rev <rev>

Ignore changes made by the revision when assigning blame, as if the change never happened. Lines that were changed or added by an ignored commit will be blamed on the previous commit that changed that line or nearby lines. This option may be specified multiple times to ignore more than one revision. If the **blame.markIgnoredLines** config option is set, then lines that were changed by an ignored commit and attributed to another commit will be marked with a **?** in the blame output. If the **blame.markUnblamableLines** config option is set, then those lines touched by an ignored commit that we could not attribute to another revision are marked with a *.

--ignore-revs-file <file>

Ignore revisions listed in **file**, which must be in the same format as an **fsck.skipList**. This option may be repeated, and these files will be processed after any files specified with the **blame.ignoreRevsFile** config option. An empty file name, **""**, will clear the list of revs from previously processed files.

--color-lines

Color line annotations in the default format differently if they come from the same commit as the preceding line. This makes it easier to distinguish code blocks introduced by different commits. The color defaults to cyan and can be adjusted using the **color.blame.repeatedLines** config option.

--color-by-age

Color line annotations depending on the age of the line in the default format. The **color.blame.highlightRecent** config option controls what color is used for each range of age.

-h

Show help message.

SEE ALSO
    git-blame(1)

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