gitinfo.sty

A package for accessing metadata from the git DVCS

Brent Longborough

29th August, 2011

Version: 1.0

CONTENTS

Introduction
  How gitinfo works ..............................................

Setup

Using the package
  Package options ..............................................
  The metadata .................................................
  For memoir users .............................................

Etc
  Acknowledgements & dependencies .........................
  Copyright & licence .........................................
  From the author .............................................
More and more, writers are using version control systems to manage the progress of their works. One popular distributed version control system commonly used today is *git*.

Among other blessings, *git* provides some useful metadata concerning the history of the developers’ work, and, in particular, about the current state of that work.

*gitinfo* allows writers to incorporate some of this metadata into their documents, to show from which point in their development a given formatted copy was produced.

**How *gitinfo* works**

1. Whenever you commit work or check out a branch in *git*, *git* executes a *post-commit* or *post-checkout* hook.
2. The *gitinfo* package includes a sample hook (placed in your *git* hooks directory), which extracts metadata from *git* and writes it to a *TeX* file, named *gitHeadInfo.gin* (*gin* for *git info*).
3. When you format your document, *gitinfo* reads *gitHeadInfo.gin* and stores the metadata in a series of *TeX* commands.
4. You may use these commands to insert the metadata you need at any point in the document.

It is important to note that *gitinfo* reads the metadata with the equivalent of `\input{./gitHeadInfo.gin}`. This is necessary to avoid possible future confusion in a *TeX* distribution where more than one package includes its metadata in this way; but it requires that for every document which uses *gitinfo*, a *gitHeadInfo.gin* must be generated in the same directory.

If you actually want to use *gitinfo*, then please read on. But you may just be reading this to see whether it will be useful; in this case, please skip the next chapter and go on to (Using the package) on page 4. If you like what you see, you can come back to read (Setup) later.
**SETUP**

*gitinfo* will be installed by your favourite package or distribution manager, but before you can start to use it, you need to configure each of your *git* working copies by setting up hooks to capture the metadata. Every document in your working copy that uses *gitinfo* will need a copy of *gitHeadInfo.gin* in the same directory.

These copies can be generated by the *git* post-hooks, which need to be configured once, at the start.

If you’re familiar with tweaking *git*, you can probably work it out for yourself. If not, I suggest you follow these steps:

1. First, you need a *git* repository and working tree. For this example, let’s suppose that the root of the working tree is in `~/compsci`

2. Copy the file `post-xxx-sample.txt` as `post-checkout` into the *git* hooks directory in your working copy. In our example case, you should end up with a file called `~/compsci/.git/hooks/post-checkout`

3. With your favourite editor, edit the file you have just created. Very near the start of the file, you will see two lines:

   ```
   prefixes=". test docs"
   prefixes="."
   ```

   The first line is an example of how you might configure for multiple copies of *gitHeadInfo.gin*; the second is the default, and means “the working copy root”.

   If all of your documents will be kept in the root of the working copy, you may leave this file alone. If not, you should delete the line `prefixes="."`, and edit the other line to include the paths, relative to the working copy root, where the documents that need access to *gitinfo* are kept.

   Suppose, for example, that you have a project called ‘compsci’. Your working copy is `~/compsci` with two paths containing documents, `~/compsci/docs/essays` and `~/compsci/docs/reference`. Then you might set up the prefixes like this:

   ```
   prefixes="docs/essays docs/reference"
   ```

   which will cause copies of the metadata to be saved in just those two directories. In this example, it will not be accessible to documents kept in the working copy’s root (designated by “.”, omitted in this case).

4. Test your setup with “git checkout master” (or another suitable branch name). This should generate copies of *gitHeadInfo.gin* in the directories you intended.

5. Now make two more copies of this file in the same directory (hooks), calling them *post-commit* and *post-merge*, and you’re done.
Once you’ve set up your *git* hooks, and done your first commit, merge, or checkout to drive them, you can start incorporating the metadata into your document.

The `gitinfo` package is loaded in the usual way:

\usepackage[< options >]{gitinfo}

### Package options

The following options are available:

- **grumpy**: By default, If `gitinfo` can’t find `gitHeadInfo.gin` (the metadata file), it will set all the metadata to a common value, “(None)”, issue a package warning, and carry on. If the grumpy option is used, this warning becomes an error, and processing stops.

- **missing=**text**: If `gitinfo` can’t find `gitHeadInfo.gin` (the metadata file), it will normally set all the metadata to a default common value, “(None)”. If you wish, you can change this value to something else with, say `missing=Help!` or `missing={What a mess}`.

  If you have complex needs, as in the second example, don’t forget to enclose your text in {}s.

- **footinfo**: For *memoir* users, `gitinfo` will set certain items of metadata into some *memoir* page footer categories, provided that (a) the memoir package is loaded, and (b) this option is given.

  No warning is given, and no action taken, if this parameter is used with another document class. More detail is given below (For *memoir* users).

- **pcount**: For *memoir* users, this option, when used together with `footinfo`, will, in the affected footers, replace the folio with one of the form \(x/y\), where \(x\) is the folio and \(y\) is the page count.

  No warning is given, and no action taken, if this parameter is used with another document class, or if the `footinfo` option is omitted. More detail is given below (For *memoir* users).
The metadata

The `git` metadata, for the current HEAD commit, is made available in the document as a series of parameter-less \TeX commands. Here they are:

\begin{itemize}
  \item \texttt{\gitAbbrevHash} The seven-hex-char abbreviated commit hash
  \item \texttt{\gitHash} The full 40-hex-character commit hash
  \item \texttt{\gitAuthorName} The name of the author of this commit
  \item \texttt{\gitAuthorEmail} The email address of the author of this commit
  \item \texttt{\gitAuthorDate} The date this change was committed by the author, in the format \texttt{yyyy-mm-dd}
  \item \texttt{\gitAuthorIsoDate} The date and time this change was committed by the author, in ISO format, e.g. \texttt{2011-08-29 13:05:54 +0100}
  \item \texttt{\gitAuthorUnixDate} The date and time this change was committed by the author, as a Unix timestamp, e.g. \texttt{134619554}
  \item \texttt{\gitCommitterName} The name of the committer of this commit
  \item \texttt{\gitCommitterEmail} The email address of the committer of this commit
  \item \texttt{\gitCommitterDate} The date this change was committed by the committer, in the format \texttt{yyyy-mm-dd}
  \item \texttt{\gitCommitterIsoDate} The date and time this change was committed by the committer, in ISO format, e.g. \texttt{2011-08-29 13:05:54 +0100}
  \item \texttt{\gitCommitterUnixDate} The date and time this change was committed by the committer, as a Unix timestamp, e.g. \texttt{134619554}
  \item \texttt{\gitReferences} A list of any `git` references (tags, branches) associated with this commit, e.g. `(HEAD, master)`
\end{itemize}

Three more commands are available, but their use should be considered experimental. `gitinfo` searches the `git` references metadata for anything (probably a `git` tag) that looks like a number with a decimal point. The first such number it finds is taken as a “Version Number” and made available in three different formats, explained here:

\begin{itemize}
  \item \texttt{\gitVtag} The version number, without any decorations. If no version number is found, empty (i.e. zero width).
  \item \texttt{\gitVtags} The version number, with a leading space. If no version number is found, empty.
  \item \texttt{\gitVtagn} The version number, with a leading space. If no version number is found, a space, followed by the default or specified value of the missing package option.
\end{itemize}
For *memoir* users

If you use *memoir*, *gitinfo* provides some preset page headers and footers for you, if you specify the `footinfo` option as described above.

With the `footinfo` option, *gitinfo* redefines the *plain, ruled*, and *headings* pagestyles.

For the *plain* and *ruled* pagestyles, the folio is moved from the centre to the outer margin of the footer, and a revision stamp is placed in the inner margin.

For the *headings* pagestyle, the folio is moved from the outer margin of the header to the outer margin of the footer, and a revision stamp is placed in the inner margin of the footer.

If, as well, you use the `pcount` option, a solidus, and the page count, are appended to the folio.

The revision stamp id generated by this fragment:

Revision\gitVtags: \gitAbbrevHash{} (\gitAuthorDate)

which is set at tiny in the sans-serif font. You can see an example in the footer of this page.
Acknowledgements & dependencies

The \TeX.SE community has been a constant source of help, inspiration, and amazement. In particular, I’d like to thank Joseph Wright, who rescued me from the jaws of the TeX parser by explaining `\expandafter`. I’m also like to register my thanks to the owners of the packages on which gitinfo depends: etoolbox, kvoptions, and xstring.

The failings, of course, are all mine.

Copyright & licence

Copyright © 2011, Brent Longborough.

This work — gitinfo — may be distributed and/or modified under the conditions of the LaTeX Project Public License: either version 1.3 of this license, or (at your option) any later version.

The latest version of this license is at \url{http://www.latex-project.org/lppl.txt}, and version 1.3 or later is part of all distributions of \LaTeX{}version 2005/12/01 or later.

This work has the LPPL maintenance status ‘maintained’; the Current Maintainer of this work is Brent Longborough.

This work consists of the files gitinfo.sty, gitsetinfo.sty, gitinfo.tex, gitinfo.pdf, post-git-sample.txt, and gitHeadInfo.gin.

From the author

Although my limitations as a \TeXnician mean that I’ve implemented gitinfo in a rather simplistic way that needs some setup that is more complicated than I wanted, I hope you find the package useful. I’ll be very happy to receive your comments by email.

Brent Longborough

brent+gitinfo (at) longborough (dot) org
and at \TeX.SE