Abstract

This is the documentation of the \texttt{\LaTeX}git package. Several macros are defined to fetch \texttt{git} information and typeset it. The macros defined by \texttt{\LaTeX}git can be helpful to documentation authors and others to whom clear document versioning is important.

Contents

1 Overview 1

1.1 Related packages 1
1.2 Dependencies 2
1.3 Getting information 2
1.4 Interface 2

2 Options 2

3 Examples 3

4 Implementation 4

5 Index 9

1 Overview

1.1 Related packages

Brent Longborough’s \texttt{gitinfo} and \texttt{gitinfo2} packages have similar features, with a different interface. Instead of running \texttt{git} commands, it relies on special \texttt{git} files. This has the advantage that it works on Windows, however, \texttt{git} hooks have to be put into place.

*Email: info@camilstaps.nl
André Hilbig’s `gitfile-info` does something similar as `gitinfo` and `\LaTeX`git, but the package manual is in German so who knows. It seems that Python and `git` hooks are required.

### 1.2 Dependencies

`\LaTeX`git gets its information by running `git` commands. It uses `\write18` for this, so documents using `\LaTeX`git will have to be compiled with the `-shell-escape` flag.

Most of the information is retrieved using `git` but then parsed using `grep`, `cut` and similar tools. Unfortunately, this is necessary: `git log` accepts arbitrary formats, but uses the percent sign in these formats, and running commands with percent signs does not seem possible using `\write18`.

An unfortunate side effect of this is that this package will not work on Windows.

### 1.3 Getting information

`\LaTeX`git runs a shell command using `\git@command`. This macro reads the result into `\git@rawresult`. This result contains a newline character at the end that needs to be removed to avoid unwanted spacing. Therefore, `\git@result` is defined as a wrapper for `\git@rawresult` that removes this spacing with `\unskip`.

### 1.4 Interface

For each information-fetching command, two versions are defined: one, which only executes the command (leaving the result available in `\git@result`); and one, which executes the command and writes the result. An example is `\git@commithash` with its counterpart `\gitcommithash`. Usually, the latter is most useful.

### 2 Options

A number of options is available to all macros that fetch information through a `pgfkeys` interface. All keys are recognised by all macros, but not all are considered by each macro. Check the documentation for specific macros to see which options are relevant.

`directory=DIR`

Use the git repository in directory `DIR` (or its first parent directory that is a git repository). `\LaTeX`git will `cd` to this directory before executing the actual `git` command.

`formatDate`

Format dates using `datetime`’s `\formatdate`.

`formatTime`

Format times using `datetime`’s `\formattime`.

2
formatInterDateTime

If both formatDate and formatTime are set, this is put in between (e.g. \textquote{space}at\textquote{space}) in English — this is also the default.

revision=REV

Get the hash of revision REV (e.g. master, HEAD\textasciicircum\textasciicircum\textasciicircum, etc.). Note that if multiple circumflexes are desired, the catcode of \textasciicircum\textasciicircum\textasciicircum has to be changed. For example:

\begin{verbatim}
\textbackslash{catcode}\textasciicircum=11
\textbackslash{gitcommithash}[\text{revision}={\textbackslash{HEAD}\textasciicircum\textasciicircum\textasciicircum}]
\end{verbatim}
05c430b

showTimeZone

When formatTime is set: show the time zone between parentheses after the time.

shortHash

Get only the first seven characters of the hash, as in \texttt{git log --oneline}.

3 Examples

These macros are used to get and print basic commit information.

\begin{verbatim}
\textbackslash{catcode}='11
\textbackslash{gitcommithash}\textbackslash{[shortHash=false]} \textbackslash{gitcommitmsg} \\
Three commits ago was \textbackslash{gitcommithash}\textbackslash{[revision=HEAD\textasciicircum\textasciicircum\textasciicircum]}.

The latest commit was by \textbackslash{gitcommitauthorname} \textbackslash{\texttt{[\textbackslash{gitcommitauthoremail}]}}. 
In raw format, the author is \texttt{\textit{\textbackslash{gitcommitauthor}}}.

The last three commits were:
\texttt{\small
\textbackslash{gitcommithash} \textbackslash{gitcommitmsg} \\
\textbackslash{gitcommithash}[\text{revision}={\textbackslash{HEAD}\textasciicircum}] \textbackslash{gitcommitmsg}[\text{revision}={\textbackslash{HEAD}\textasciicircum}] \\
\textbackslash{gitcommithash}[\text{revision}={\textbackslash{HEAD}\textasciicircum\textasciicircum\textasciicircum}] \textbackslash{gitcommitmsg}[\text{revision}={\textbackslash{HEAD}\textasciicircum\textasciicircum\textasciicircum}]
}
\end{verbatim}

This is commit d175d01d96404952c71e6b20a9c575c6ba8e79e8, or in short d175d01. Three commits ago was 05c430b.
The latest commit was by Camil Staps (info@camilstaps.nl). In raw format, the author is Camil Staps <info@camilstaps.nl>.
The last three commits were:
d175d01 Preparing CTAN release
8ba4321 readline is less prone to errors with special characters than read
50a7eee Added gitchanges

\textbackslash{gitcommitdate}

This macro displays the \texttt{git commit date}. The following example shows a usage example and shows the results of the options \texttt{formatDate}, \texttt{formatTime} and \texttt{showTimeZone}.

\begin{verbatim}
This is commit d175d01d96404952c71e6b20a9c575c6ba8e79e8, or in short d175d01. Three commits ago was 05c430b.
The latest commit was by Camil Staps (info@camilstaps.nl). In raw format, the author is Camil Staps <info@camilstaps.nl>.
The last three commits were:
d175d01 Preparing CTAN release
8ba4321 readline is less prone to errors with special characters than read
50a7eee Added gitchanges
\end{verbatim}

\begin{verbatim}
This is commit d175d01d96404952c71e6b20a9c575c6ba8e79e8, or in short d175d01. Three commits ago was 05c430b.
The latest commit was by Camil Staps (info@camilstaps.nl). In raw format, the author is Camil Staps <info@camilstaps.nl>.
The last three commits were:
d175d01 Preparing CTAN release
8ba4321 readline is less prone to errors with special characters than read
50a7eee Added gitchanges
\end{verbatim}

3
\footnotesize
\begin{tabular}{c c c c}
\texttt{formatDate} & \texttt{formatTime} & \texttt{showTimeZone} & Result \\
\hline
\checkmark & & (any) & \gitcommitdate[showTimeZone] \\
& \checkmark & & \gitcommitdate[formatTime] \\
& \checkmark & \checkmark & \gitcommitdate[formatTime,showTimeZone] \\
\checkmark & & (any) & \gitcommitdate[formatDate,showTimeZone] \\
\checkmark & \checkmark & & \gitcommitdate[formatDate,formatTime] \\
\checkmark & \checkmark & \checkmark & \gitcommitdate[formatDate,formatTime,showTimeZone] \\
\end{tabular}

\footnotesize
\begin{tabular}{|c|c|c|}
\hline
\texttt{formatDate} & \texttt{formatTime} & \texttt{showTimeZone} \\
\hline
(any) & 2016-08-24 15:01:41 +0200 \\
\checkmark & 15:01 \\
\checkmark & \checkmark & 15:01 (+0200) \\
\checkmark & (any) & Wednesday 24\textsuperscript{th} August, 2016 \\
\checkmark & \checkmark & Wednesday 24\textsuperscript{th} August, 2016 at 15:01 \\
\checkmark & \checkmark & \checkmark & Wednesday 24\textsuperscript{th} August, 2016 at 15:01 (+0200) \\
\end{tabular}

4 Implementation

Define the package and load required packages.

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexgit}[2016/08/24]
\RequirePackage{pgfkeys}
\RequirePackage{datetime}
\latexgit
Prints the name of the package.
\git@result
When a git command is run, the result is stored in \git@rawresult. However, using \read results in spacing at the end of the macro. Hence, we need to use \unskip to remove this spacing. This is a wrapper for \git@rawresult that adds this \unskip.
\newcommand{\latexgit}{\LaTeX{}git}
\newcommand{\git@result}{\git@rawresult\unskip}

We now define the keys accepted by git macros. The following options are recognised. Check the documentation of the macro to see which options are considered.
\newif\ifgit@opt@FormatDate
\newif\ifgit@opt@FormatTime
\newif\ifgit@opt@ShortHash
\newif\ifgit@opt@ShowTimeZone
\pgfkeys{
  /git/.is family, /git,
default/.style={
directory=.,
fmtDate=false,
fmtInterDateTime=\space{}at\space{},
fmtTime=false,
revision=HEAD,
showTimeZone=false,
shortHash=true,
},
directory/.estore in=\git@opt@Directory,
fmtDate/.is if=git@opt@FormatDate,
fmtInterDateTime/.estore in=\git@opt@FormatInterDateTime,
fmtTime/.is if=git@opt@FormatTime,
revision/.estore in=\git@opt@Revision,
showTimeZone/.is if=git@opt@ShowTimeZone,
shortHash/.is if=git@opt@ShortHash,
}
\git@command
Run a \texttt{git} command and read the output into \texttt{\git@rawresult}. Before running the command, the directory will change to \texttt{\git@opt@Directory}.
\git@stream
\newcommand{\git@command}[1]{%
\def\git@rawresult{}%
\openin\git@stream="cd \git@opt@Directory; #1"
\ifeof\git@stream%
\PackageError{latexgit}{invoke \LaTeX \texttt{\ -shell-escape} flag}%
\else%
\begingroup%
\catcode`\^^M=9%
\endlinechar=-1%
\readline\git@stream to \git@streamoutput%
\PackageError{latexgit}{invoke \LaTeX\texttt{\ -shell-escape} flag}%
\else%
\fi%
}
\gitcommithash
Get a commit hash. Recognised options: \texttt{directory}, \texttt{revision}, \texttt{shortHash}.
\git@commithash
Like \texttt{\gitcommithash}, but does not return the output.
\newcommand{\git@commitmsg}[1][]{% 
  \pgfkeys{/git,default,#1}\
  \ifgit@opt@ShortHash
  \git@command{git log -n 1 --oneline \git@opt@Revision
    | cut -d' ' -f1}\
  \else
  \git@command{git log -n 1 \git@opt@Revision
    | head -1 | cut -d' ' -f2}\
  \fi\
}\gitcommitmsg

Get a commit message. Recognised options: directory, revision.

\newcommand{\git@commitmsg}[1][]{% 
  \pgfkeys{/git,default,#1}\
  \git@commitmsg[#1]\git@result}

\git@commitmsg
Like \gitcommitmsg, but does not return the output.

\newcommand{\git@commitmsg}[1][]{% 
  \pgfkeys{/git,default,#1}\
  \git@command{git log -n 1 --oneline \git@opt@Revision
    | cut -d' ' -f2-}\
}\git@formatCommitDate

Format a commit date in ISO format using datetime's \formatdate.

\def\git@formatCommitDate#1-#2-#3 #4:#5:#6 +#7\relax{\
  \formatdate{#3}{#2}{#1}\
}\git@formatCommitTime

Format a commit time using datetime's \formattime. Recognised options: showTimeZone.

\def\git@formatCommitTime#1-#2-#3 #4:#5:#6 +#7\relax{\
  \formattime{#4}{#5}{#6}\
  \ifgit@opt@ShowTimeZone\
    \space(+#7)\
  \fi\
}\gitcommitdate

Get a commit date. If \formatDate is set, \git@formatCommitDate will be used. If \formatTime is set, and \formatDate is unset, \git@formatCommitTime will be used. Recognised options: directory, \formatDate, \formatTime, revision, showTimeZone.
\ifx\else\git@result\fi\fi

\texttt{\textbackslash git\textunderscore commitdate}\ Like \texttt{\textbackslash gitcommitdate}, but does not return the output.
\newcommand{\git@commitdate}[1][]{%
\pgfkeys{/git,default,#1}%
\git@command{git log -n 1 --date=iso \git@opt@Revision | grep Date | head -1 | cut -d' ' -f2-}%
}

\texttt{\textbackslash git\textunderscore commitauthor}\ Get a commit author. Recognised options: directory, revision.
\newcommand{\git@commitauthor}[1][]{%
\pgfkeys{/git,default,#1}%
\git@command{git log -n 1 \git@opt@Revision | grep Author | head -1 | cut -d' ' -f2-}%
}

\texttt{\textbackslash git\textunderscore commitauthorname}\ Get a commit author’s name. Recognised options: directory, revision.
\newcommand{\git@commitauthorname}[1][]{%
\pgfkeys{/git,default,#1}%
\git@command{git log -n 1 \git@opt@Revision | grep Author | head -1 | cut -d'<' -f1}%
}

\texttt{\textbackslash git\textunderscore commitauthoremail}\ Get a commit author’s email address. Recognised options: directory, revision.
\newcommand{\git@commitauthoremail}[1][]{%
\pgfkeys{/git,default,#1}%
\git@command{git log -n 1 \git@opt@Revision | grep Author | head -1 | cut -d' ' -f2- | cut -d'<' -f2 | cut -d'>' -f1}%
}
\texttt{\git@commitparent} Get the hash of the first parent.
\begin{verbatim}
117 \newcommand{\git@commitparent}{1}{}
118 \pgfkeys{/git,default,#1}\
119 \git@command{git log -n 1 --pretty=raw \git@opt@Revision
120 | grep parent | head -1 | cut -d’ ’ -f2}\
121 }
\end{verbatim}

\texttt{\gitchanges} Record the full git commit history (following first parents using \texttt{\git@commitparent}) using \texttt{\changes}.
\begin{verbatim}
122 \newcommand{\gitchanges}{1}{}
123 \git@changes[HEAD]{1}\
124 }
\end{verbatim}

\texttt{\git@changes} Like \texttt{\gitchanges}, but accepts an extra argument for the revision.
\begin{verbatim}
125 \newcommand{\git@changes}{2}{}
126 \edef\git@@revision{#2}\
127 \git@commithash[revision=\git@@revision]\%\n128 \edef\git@@thishash{\git@rawresult}\
129 \git@command{git log -n 1 --date=iso \git@@revision
130 | grep Date | head -1 | cut -d’ ’ -f4}\
131 \edef\git@@thisdate{\git@rawresult}\
132 \git@commitmsg[revision=\git@@revision]\%\n133 \edef\git@@thismsg{\git@rawresult}\
134 \changes{\git@@thisdate\unskip: \git@@thishash}\git@@thisdate\git@@thismsg\
135 \git@commitparent[revision=\git@@revision]\%\n136 \let\git@@parent\git@rawresult\
137 \setbox0=\hbox{\git@@parent\unskip}\ifdim\wd0=0pt\else\fi\
139 \git@changes{\git@@parent}\
140 \fi\
141 }
\end{verbatim}
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Commit ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-07-23</td>
<td>721067b</td>
<td>General: Initial commit</td>
</tr>
<tr>
<td>2016-07-24</td>
<td>0548c53</td>
<td>General: documentation: added overview section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016-07-24</td>
<td>2ed89f0</td>
<td>General: Removed readme from makefile</td>
</tr>
<tr>
<td>2016-07-24</td>
<td>3abab51</td>
<td>General: documentation fixes; protecting GitCommitDate and the like with @</td>
</tr>
<tr>
<td>2016-07-24</td>
<td>6ce1a94</td>
<td>General: first implementation: commit hashes, messages, dates and authors</td>
</tr>
<tr>
<td>2016-07-24</td>
<td>8cbd796</td>
<td>General: gitignore, makefile</td>
</tr>
<tr>
<td>2016-07-25</td>
<td>14e6347</td>
<td>General: added new intermediary files to Makefile</td>
</tr>
<tr>
<td>2016-07-25</td>
<td>ea7048e</td>
<td>General: documentation: related packages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016-07-25</td>
<td>f070e93</td>
<td>General: documentation: added options, examples, index, history</td>
</tr>
<tr>
<td>2016-08-01</td>
<td>05e430b</td>
<td>General: moved table about date formatting to examples section</td>
</tr>
<tr>
<td>2016-08-02</td>
<td>50a7eee</td>
<td>General: Added gitchanges</td>
</tr>
<tr>
<td>2016-08-02</td>
<td>8ba4321</td>
<td>General: readline is less prone to errors with special characters than read</td>
</tr>
<tr>
<td>2016-08-24</td>
<td>d175d01</td>
<td>General: Preparing CTAN release</td>
</tr>
</tbody>
</table>
6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

\begin{verbatim}
G
\git@changes ................. 123, 125
\git@command ................. 31
53, 56, 64, 92, 99, 106, 113, 119, 129
\git@commitauthor ............ 96, 97
\git@commitauthoremail ...... 110, 111
\git@commitauthorname ...... 103, 104
\git@commitdate ............... 77, 90
\git@commithash .............. 49, 50, 127
\git@commitmsg ............... 61, 62, 132
\git@commitparent ............ 117, 135
\git@formatCommitDate ...... 67, 79
\git@formatCommitTime ...... 70, 82, 85
\git@opt@Directory .......... 23, 34
\git@opt@FormatInterDateTime 25, 81
\git@opt@Revision ........... 27,
53, 56, 64, 92, 99, 106, 113, 119
\git@rawresult .............. 7,
33, 44, 79, 82, 85, 128, 131, 133, 136
\git@result ................. 7, 49, 61, 87, 96, 103, 110
\git@stream ................. 31, 34, 35, 43
\git@streamoutput .......... 43, 44
\git@changes ................. 122
\git@commitauthor .......... 3, 95
\git@commitauthoremail ...... 3, 100
\git@commitauthorname ...... 3, 102
\git@commitdate .............. 3, 76
\git@commithash ............. 3, 48
\git@commitmsg .............. 3, 60
\git@commitparent .......... 117, 135
\git@formatCommitDate ...... 67, 79
\git@formatCommitTime ...... 70, 82, 85
\ifgit@opt@FormatDate ...... 8, 78
\ifgit@opt@FormatTime ...... 9, 80, 84
\ifgit@opt@FormatInterDateTime 25, 81
\ifgit@opt@Revision ........ 10, 52
\ifgit@opt@ShowTimeZone ...... 11, 72
53, 56, 64, 92, 99, 106, 113, 119
\latexgit ........................ 6
\end{verbatim}