Extracting Information from (La)TeX Source Files

Jean-Michel Hufflen

TUG 2022

24 July 2022
Extracting Information from \LaTeX{} Source Files

Jean-Michel Hufflen

Contents

Introduction

Description

Discussion

Presently

Conclusion

2/2/15
is a wonderful tool for typesetting texts $\Leftarrow$ well known;
TEX is a wonderful tool for typesetting texts $\Leftarrow$ well known; knows only its own formats $\Leftarrow$ well known, too.
is a wonderful tool for typesetting texts \iffalse well known; well known, too. Some information belonging to (L\LaTeX{} source files may be usable for other purposes than typesetting, e.g., generating metadata for Web search engines.
\TeX\ldots

is a wonderful tool for typesetting texts $\Leftarrow$ well known; knows only its own formats $\Leftarrow$ well known, too. Some information belonging to (\La)\TeX source files may be usable for other purposes than typesetting, e.g., generating metadata for Web search engines. (\La)\TeX’s commands can do such jobs, but this is misuse and complicates the writing of classes. \TeX has not been designed for that, it is preferable to use modern programming languages, with more suitable structures.
Functional programming

Functions are first-class objects, as other data.
Functional programming

Functions are first-class objects, as other data. Functions can be results of a computation.
Functional programming

Functions are first-class objects, as other data. Functions can be results of a computation. So we can easily write *generators* of functions.
Functional programming

Functions are first-class objects, as other data. Functions can be results of a computation. So we can easily write *generators* of functions. Scheme $\leftarrow$ elegant, data and programs have the same format, as in any Lisp dialect.
Building a *parsing function*

\[ (g\text{-}mk\text{-}tex\text{-}parsing\text{-}f \textit{directive} \ldots) \]

All the *directives* are grouped, ‘compiled’ into a function ready to parse a source file.
Directives

\[ (g\text{-retain-command } \text{command-name } \text{arg-nb } \text{optional-arg?} \]
\[ \text{top-level? } \text{recursive? } \text{preamble?} \]
\[ \text{occ-nb-info function} ) \]

where:

\text{command-name} \text{ is the name of the command to be caught;}

\text{arg-nb} \text{ is the argument number of this command;}

\text{optional-arg?} \text{ is true if the first argument is optional, surrounded by square brackets, false otherwise;}

\text{top-level?} \text{ is true if this command is to be searched only at the top level, false otherwise;}

\text{recursive?} \text{ is used when } \text{input} \text{ commands are encountered: if it is true, corresponding files are searched recursively, otherwise such an } \text{input} \text{ command is just skipped;}

\text{preamble?} \text{ stops searching after a preamble if it is bound to true; otherwise, goes on.}
Other arguments

\texttt{occ-nb-info} may be bound to:

- 0 or the false value (\texttt{#f}): the command should not appear within the file, this is checked;
- a positive integer \( n \): the first \( n \) occurrences of this command are processed, and following ones are ignored;
- the true value (\texttt{#t}): all the occurrences of this command are processed;

\texttt{function} the Scheme function to call, it \textit{must} have the same number of arguments than \texttt{\textbackslash command-name}. 
Directives (con’d)

\begin{verbatim}
(g-retain-match command-name s top-level?
   recursive? preamble occ-nb-info
   function)
\end{verbatim}

is used when \texttt{\textbackslash{command-name}}’s arguments are expressed by means of a pattern, e.g., "\texttt{#1\endcsname}" for the \texttt{\csname} command. \texttt{s} is a string bound to such a pattern, the other arguments have the same meaning than g-retain-command’s.
Directives \textit{(con’d)}

\begin{verbatim}
(g-retain-match command-name s top-level?
  recursive? preamble occ-nb-info function)
\end{verbatim}

is used when \texttt{\textbackslash command-name}'s arguments are expressed by
means of a pattern, e.g., "\texttt{\#1\endcsname}" for the \texttt{\csname}
command. \texttt{s} is a string bound to such a pattern, the other
arguments have the same meaning than \texttt{g-retain-command}'s.
The arguments of corresponding functions in Scheme are
strings in both cases.
g-mk-tex-parsing-f builds a function that applies to a filename and returns:

- **false** if something went wrong, or a forbidden command is included into the file;
- **true** in all other cases.
Result’s result

g-mk-tex-parsing-f builds a function that applies to a filename and returns:

- **false** if something went wrong, or a forbidden command is included into the file;
- **true** in all other cases.

You have to update your own structures when a file is parsed. If an error occurs, they may be in an inconsistent state.
Destructuring

These functions can be used to destructure an optional argument—\(s, s_0\) are strings—:
Destructuring

These functions can be used to destructure an optional argument—\( s, s_0 \) are strings—:

\[(g\text{-}parse\text{-}to\text{-}list \ s)\] returns the elements of a comma-separated list;

\[(g\text{-}parse\text{-}to\text{-}alist \ s \ s_0)\] returns the successive associations of a comma-separated list whose elements are \( key=value \) pairs; if a key is given without a value, this missing value is replaced by \( s_0 \).

In both cases, the original order is preserved.
Destructuring

These functions can be used to destructure an optional argument—$s, s_0$ are strings—:

$$(g\text{-}parse\text{-}to\text{-}list \ s)$$ returns the elements of a comma-separated list;

$$(g\text{-}parse\text{-}to\text{-}alist \ s \ s_0)$$ returns the successive associations of a comma-separated list whose elements are $key=value$ pairs; if a key is given without a value, this missing value is replaced by $s_0$.

In both cases, the original order is preserved.

**Remark** Note that $g\text{-}parse\text{-}to\text{-}list$, $g\text{-}parse\text{-}to\text{-}alist$, $g\text{-}retain\text{-}command$ and $g\text{-}retain\text{-}match$ are functions, whereas $g\text{-}mk\text{-}tex\text{-}parsing\text{-}f$ is a *macro*. 
Example

Considering a source text for \LaTeX, give:

- the used options of the babel package,
- the title,
- the number of occurrences of the \texttt{\textbackslash emph} command.

(Show.)
When I realised \texttt{MiBibTeX}, I needed to know which options of the babel package were used. Only the preamble was to be searched.
‘History’

When I realised \texttt{M\textsc{I}B\textsc{I}B\textsc{T}E\textsc{X}}, I needed to know which options of the babel package were used. Only the preamble was to be searched.
The same to determine which encoding was used.
‘History’

When I realised MiBIBTEX, I needed to know which options of the babel package were used. Only the preamble was to be searched.

The same to determine which encoding was used. One year and a half ago, I became the new editor of the journal of the French-speaking TEX user group.
When I realised MiBibTEX, I needed to know which options of the babel package were used. Only the preamble was to be searched.

The same to determine which encoding was used. One year and a half ago, I became the new editor of the journal of the French-speaking TeX user group. I decided to revise the class used for this journal and discovered that the previous class was used to build other files, such as metadata for Web search engines.
‘History’

When I realised MiBibTEX, I needed to know which options of the babel package were used. Only the preamble was to be searched.
The same to determine which encoding was used.
One year and a half ago, I became the new editor of the journal of the French-speaking \TeX user group.
I decided to revise the class used for this journal and discovered that the previous class was used to build other files, such as metadata for Web search engines.
From my point of view, \TeX is intended for typesetting, and other tasks should be delegated to more modern programming languages.
When I realised \texttt{M\LaTeX B\LaTeX}, I needed to know which options of the babel package were used. Only the preamble was to be searched.

The same to determine which encoding was used.

One year and a half ago, I became the new editor of the journal of the French-speaking \TeX user group.

I decided to revise the class used for this journal and discovered that the previous class was used to build other files, such as metadata for Web search engines.

From my point of view, \LaTeX is intended for typesetting, and other tasks should be delegated to more modern programming languages.

I could have used Lua...
'History'

When I realised MiBibTeX, I needed to know which options of the babel package were used. Only the preamble was to be searched.

The same to determine which encoding was used.

One year and a half ago, I became the new editor of the journal of the French-speaking TeX user group. I decided to revise the class used for this journal and discovered that the previous class was used to build other files, such as metadata for Web search engines.

From my point of view, TeX is intended for typesetting, and other tasks should be delegated to more modern programming languages.

I could have used Lua... but LuaTeX was unable to process some texts designed for pdfTeX or XeLaTeX.
Principles

Simple extractions—e.g., a title—must be simple.
Principles

Simple extractions—e.g., a title—must be simple. If addressing occurrences of a \TeX command inside a source text, it should be possible to catch it.
Principles

Simple extractions—e.g., a title—must be simple. If addressing occurrences of a $\TeX$ command inside a source text, it should be possible to catch it. But more experience will be needed.
Current state

Some bugs to fix for the pattern-matching of \TeX command's arguments.
Current state

Some bugs to fix for the pattern-matching of \TeX command’s arguments.
The other points seems to be OK.
Current state

Some bugs to fix for the pattern-matching of \TeX command’s arguments.
The other points seems to be OK.
Available as a Scheme library.
For many years, we have seen that in addition to \TeX{}‘s works, many other tasks are more related to ‘classical’ programming. In particular, that is why Lua\TeX{} emerged.
Ending

For many years, we have seen that in addition to \TeX’s works, many other tasks are more related to ‘classical’ programming. In particular, that is why Lua\TeX emerged. We relate our work to avoiding *information redundancy*. 