BibTeX2MlBibTeX

a *Graphical User Interface* to manage MlBibTeX files

Olivio Mariano – University of Franche-Comte (Master 1 Info)

Abstract

The aim of this project is to introduce a new utility program that allows the user to convert BibTeX syntax to its new implementation MlBibTeX. This paper will explain its capabilities, its limitations and how it works within a UNIX-based environment. It also introduces readers to some of the new technologies used to better the interaction between the user and the operating system in conjunction with a graphical user interface.
Keywords LaTeX, BibTeX, MlBibTeX, Graphical User Interface *(GUI)* , bibliography files, Ruby, Qt.

Acknowledgment

This project has been developed thanks to the help and support of Mr. J.-M. Hufflen.

Introduction

BibTeX is a program originally designed to generate bibliographies in conjunction with the LaTeX Document Preparation System. LaTeX, available for most computer systems, is a system for typesetting documents, independent of the output device (it is based on the TeX typesetting system by Donald Knuth [6]). BibTeX is a separate program that produces the source list for a document, obtaining the information from a bibliographic database. BibTeX has been a reference for a long time since it has been very stable and it makes so easy to manage a big database of bibliographic references. This is a great instrument to work with, even if "modern features" such as the support of multiple languages are not implemented; that's why a new version of BibTeX has been developed. This is called MlBibTeX (MultiLingualBibTeX). As readers will see, there are some differences – at the syntactic level – between BibTeX and MlBibTeX. This paper focuses on this last problem and presents a new easy-to-use GUI-based script that will allow, to the end-user, the conversion between these two main formats.

1 Core

The core of this project is the implementation of an easy-to-use utility application for the end-user to convert the standard BibTeX syntax to its new implementation MlBibTeX. We will see in the next sections how the two standards work and we will explain the structural choices to accomplish the entire job.

1.1 Theory

1.1.1 BibTeX

As written in the article *"MlBibTeX a new implementation of BibTeX"* [2] BibTeX is a program used in conjunction with LaTeX to make easy to use LaTeX's built-in bibliography and citation mechanisms. The numbering of bibliographic references manually and the referenced texts at the end of the document is a typical problem for any text editor's user. Using BibTeX, it is possible to compile a large number of references into a bibliography database, extracting those cited in the document. BibTeX automatically extracts required references, and puts them into a document, correctly sorted, formatted and arranged with a bibliography style. All the user has to do is to identify each reference within the bibliography database with a unique (and for the own sake, informative) key. It is possible to use the `\cite{}` command to cite that particular reference. Trying to run LaTeX on the file, collects all of these keys; we will then run the scanning of the bibliography database for them extracting and formatting them according to certain styles. By running LaTeX a second time the reference list is incorporated into the document. As reference, we'll say that there's a standard procedure to follow:

- run LaTeX on the file. This implies some warnings from the compiler such as:

  ```
  Citation COMPANION on page x3 undefined on input line 62.
  ```

  At this stage LaTeX will have listed in an `.aux` file the citations it came across.
- run BibTeX on the same file. Now, when we run the BibTeX tool, we will actually be running it on our `.aux` file, rather than our `.tex` file. BibTeX looks in the `.aux` file for what references it needs to look out for. It grabs the data from that file and formats it how we've asked. This data will be written in a `bbl` file.
- run LaTeX again to produce a fully-referenced document.

The whole process LaTeX - BibTeX - LaTeX is classical when trying to produce a document using BibTeX. Notice that another pass through LaTeX will be needed to complete the task.

1.1.2 BibTeX' Limitations

Some important things can be noticed using BibTeX. This is, for sure, a great instrument to work with, but is far to be easy to use

in some particular fields for the end-user. An important limitation is the name handling functions. If the target is managing a simple `John Smith` style name, everything is Ok, but problems arise from more complicated names from different cultural origins. To deal with tricky surnames like "Julian de Silva" it is possible to use the form `von Surname, Firstname` [1]. So, in our example it will be `de Silva, Julian`. `de` uncapitalized is supposed to be a particle; that is, the "von" part with BibTeX terminology. The whole is supposed to match the `Fist,von,Last` pattern. The "von" part may be capitalized in some non-english first names; that implies that we want that the "von" part is still considered as the middle bit. Here we are: we need to trick BibTeX and we can see how its syntax can be complex. If we enter :

```
Julian {\uppercase{d}e La} Silva
```

the middle part "De La" will be the "von" part since BibTeX will not understand the LaTeX uppercase command and only will see the lower-case "de". This shows how BibTeX, for the moment, implements names. In summary, there are three ways to type a name in BibTeX and we need to trick it sometimes to get a good output. Indeed it is possible to find others – so called – limitations of BibTeX regarding *multiple names, titles, capilatization* simply seeing how BibTeX deals with LaTeX commands. Let's do some examples:

- the bibliography style decides whether a title should be uncapitalized. However sometimes, it is needed that a capital must remain in a title. There's a method to trick BibTeX and force it: using the braces like that:

  ```
  TITLE = "Ponti di {V}iaggiu"
  TITLE = "Ponti di {Viaggiu}"
  ```

- BibTeX can get confused by LaTeX commands generating accented letters. Another time, braces are important. The accented character should be sorrounded by braces so that BibTeX knows it doesn't have to worry about it. Note that it is not possible to enclose braces into other braces and a backslash should be the first character inside the braces.

So, as far as BibTeX is used in this situation, it will remain sometimes *tricky* to use and less linear for the end-user. That's why some of these *limitations* are treated as "problems to be solved" by the new implementation of BibTeX.

1.1.3 MlBibTeX

As said before, MlBibTeX is a new implementation of BibTeX ([4], [2],[5]). So, after explaining how BibTeX can be hard to manage (from a syntactic point of view), we can now see how a better BibTeX can be useful. Indeed, BibTeX does not support *multilingual* features nicely; that's to say it isn't possible, within the present program, to manage

multiple languages for titles, authors, and other `.bib`[1] fields except by the use of workarounds. MlBibTeX will do the task, opening the BibTeX technology to multilingual entries. Indeed, MlBibTeX will have some custom syntax changes to better BibTeX.

1.1.4 Improvements

MlBibTeX uses a new and more ergonomic syntax to process the entries. Clearly, "ergonomic" is the right word to use when trying to explain how MlBibTeX is in comparison with the classical BibTeX.

So, first of all, as said before, MlBibTeX introduces the managing of multiple languages in the entries. This aims to interact with the LaTeX interpreter to decide what language is used to be implemented compiling the document. Let's display an entry in MlBibTeX :

```
@BOOK{king1990,
        AUTHOR = {Beniamino Rovere},
        TITLE = {Il cammino},
        NOTE = {[Unter der Pseudonym ] * german
                [Sullo pseudonimo di] * italian
                Beslacco Ivra},
        PUBLISHER = {Florandi \&~C\textsuperscript{o}},
        ADDRESS = {Firenze},
        YEAR = 1990,
        LANGUAGE = english}
```

This is what LaTeX will have as output, when the main language is Italian:

> [1] Beniamino Rovere. Il cammino. Florandi, Firenze 1990. Sullo pseudonimo di Beslacco Ivra.

or when it is German:

> [1] Beniamino Rovere. Il cammino. Florandi, Firenze 1990. Unter der Pseudonym Beslacco Ivra.

As it is possible to see MlBibTeX incorporates a new predefined field. It is the `LANGUAGE` field. This gives to the entry the main language to follow. Either this is important for multiple entries and its value is set, by default, to `english`. In the same way we can notice that the MlBibTeX entry has others custom features, such as the `[...] * english` syntax. These are called *language switches* . This particular feature can occur in the `NOTE`, in the `AUTHOR` or `EDITOR` fields. This, because the transliteration of person names originating from languages using a non-Latin alphabet, can be tricky. Note that others features of the *multilanguage BibTeX* is the use of a more clear syntax for names and surnames. Basically, as seen before, in BibTeX it is quite common to write names in this syntax-style:

```
AUTHOR = {Julian {\uppercase{d}e La} Silva}
```

[1] .bib is the extension for BibTeX files

In MlBibTeX the same author could have this syntax:

```
AUTHOR = {first => Julian, von => De La, last => Silva}
```

As it is possible to see, this last kind of syntax, is more clear and more usable by the end-user. Bringing users to MlBibTeX pushed the developers to build a "better" BibTeX: that's why a part of the limitations depicted in the previous sections are now updated with new syntax and better ergonomic features. Some of these features are explained in *Names in BibTeX and MlBibTeX* [3]

1.1.5 The BibTeX2MlBibTeX conversion

So, as we can see from the previous sections, we'll have to deal with some changes trying to use the new MlBibTeX technology, even though this one fully supports the "old" BibTeX format. The conversion is quite tricky to deal with and the aim of this project points directly to this target: easy-to-use conversions between the two formats, trying to help the user to implement the all-new features of MlBibTeX.

1.2 Experimental method

1.2.1 Ruby

To design this utility for BibTeX users, the main question was what language was possible to use in this context.

So, some of the main faces of this project brought us to think that a good choice would be the use of the Perl for example, thanks to its object-oriented programming capability. Definitely this was not our choice: the language we used to implement the program exposed in this report is Ruby. So, why this choice ? *Because* :

- Ruby is open-source with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write.
- Ruby is very easy to use when written to be iplemented in different operating systems. That's to say, Ruby is a cross-platform language.
- regular expressions are quite well supported by ruby language.
- Ruby supports Qt libraries for the graphical user interface.
- regarding web-compatibility, Ruby is web-oriented and rb scripts can be used linked to web pages.

- the compatibility with different operating systems is a main feature of Ruby.

In terms of speed, Ruby's performance is inferior to that of many compiled languages (as is any interpreted language) and other major scripting languages such as Python and Perl. It is clear that the Ruby implementation in this project is still experimental and future releases will point to the better performances of this application. Ruby, in itself, is evolving fast and, in the last release of its implementation used in this project, performances are not far from expected.

1.3 Ruby and regular expressions

The direct implementation of *regular expressions* in Ruby is a quite powerful feature that is fully exploited in this project. In Ruby, a regular expression is written in the form of `/pattern/modifiers` where `pattern` is the regular expression itself, and `modifiers` are a series of characters indicating various options. The *modifiers* part is optional. This syntax is borrowed from Perl. Ruby supports a huge number of *modifiers* in Perl-style language: in this way source code can be very *compacted* even if not always easy to understand and write. A *regular expression* :

```
line = line.gsub(
/^ *author *([^}]+)\{([^,]+) ([^}]+)\},(\s+)$/i,
'    AUTHOR = {first => \2, last => \3},\4')
```

deals with the `first => <name>, last => <surname>` syntax and, even if it could be quite tricky to understand, it is the better way to use *regular expressions* integrated in Ruby. In this particular regular expression we are programming a substitution using the `gsub` Ruby function: we will identify the interesting parts to be changed using `()`. In our regular expression the first part to be identified will be `[^}]+` identified by a `\1` . For the second and the third part of the regular expression enclosed by parenthesis we will deal with the same situation, identifying them using a `\2` and a `\3`.

In this way it is possible to replace a word simply matching the *regular expressions* and the word with the replacement text. As readers will see in the example above, we *captured* a group of words to match the *regular expression* using the parenthesis: in the replacement string we invoked these *groups of words* using the `\1, \2, \n` syntax.

1.4 Use of Ruby and Qt

The ruby script itself perfoms well in the terminal. Otherwise, a GUI interface is needed for the end-user. To integrate a graphical user interface the choice is between a web-based GUI (Ruby on Rails [2]) or a Qt-based

[2]Ruby On Rails is an open source project written in the Ruby programming language. The applications using the Rails framework are developed using the Model-View-Controller design pattern.

GUI. Qt uses standard C++, but extends the language by providing an additional preprocessor that generates the C++ code which is necessary to implement Qt's extensions. As it is clear, using Qt implies writing almost ruby and C-styled language in the same code. In computer programming Qt is cross-platform, widely used for the development of GUI programs. This grants the compatibility within different platforms, allowing the crossing of our BibTeX2MlBibTeX through Linux and Mac OS X.

The implementation of the couple Qt + Ruby, known as **QtRuby** , will be noticeably present in next versions of Mac OS X and Debian [3] releases. This allows the end-user to use BibTeX2MlBibTeX in a very simple way without have to install any big packages or compile them.

Figure 1: *The MlBibTeX command line interface.*

A first, developer-side version of BibTeX2MlBibTeX is build to interact with the Unix user, considering the fact that the ruby interpreter is installed in almost operating systems. Run the script is almost easy and will require a `.bib` file as input; after processed, the script will write the output in a standard `.bib` file, integrating the multi-language conversion.

To develop our application in a graphical user styled frontend we needed some developer programs: it is assumed that we ran almost on a Unix operative system (either Linux, Mac OS or SunOS) and we had the typical development software installed:

- Ruby itself
- a GUI interface designer, either qtDesigner or kdevdesigner (essentially equivalent)
- the Qt libraries (standard on Linux using the KDE desktop)
- the Ruby Qt interface qtrubyinit
- the `rbuic`[4] program to translate qtDesigner configuration files into Ruby source.

[3]Debian is a project based around the development of a free, complete operating system through the collaboration of volunteers from around the world. The project's primary focus, Debian GNU/Linux, is a popular Linux distribution.

[4]Rbuic is the Qt Ruby UI Compiler. It generates Ruby code from a XML UI Description file. This XML UI Description file can be generated by Qt Designer, Qt's graphical UI designer. Rbuic is the Ruby equivalent for the C++ uic program.

To create a Ruby GUI program we proceeded in these steps, following the QtRuby standard interactive modes:

- Creation of a user interface using qtdesigner or kdevdesigner. At the end Qt or Kdev saves a `.ui` file. This is an *XML* data file that contains a description of the user's GUI interface design.
- `rbuic` on the command line to create a Ruby source equivalent to the designer data file. That's to say:

  ```
  $ rbuic bibtex2mlbitex.ui -o bibtex2mlbibtex_ui.rb
  ```

  this genearated the Ruby source file containing the class `BibTeX2MlBibTeXUI`, a Ruby source equivalent of the designer program's description of the GUI interface.
- Creation of a Ruby source file to subclass the GUI interface class. The `BibTeX2MlBibTeXUI` class, present in the `bibtex2mlbibtex_ui.rb` file, needs to be subclassed to separate the implementation code from the volatile interface class that is overwritten on each interface design change.

Trying to run our `.rb` [5] script resulted in executing a GUI (*figure 2*)

```
$ ./bibtex2mlbibtex.rb
```

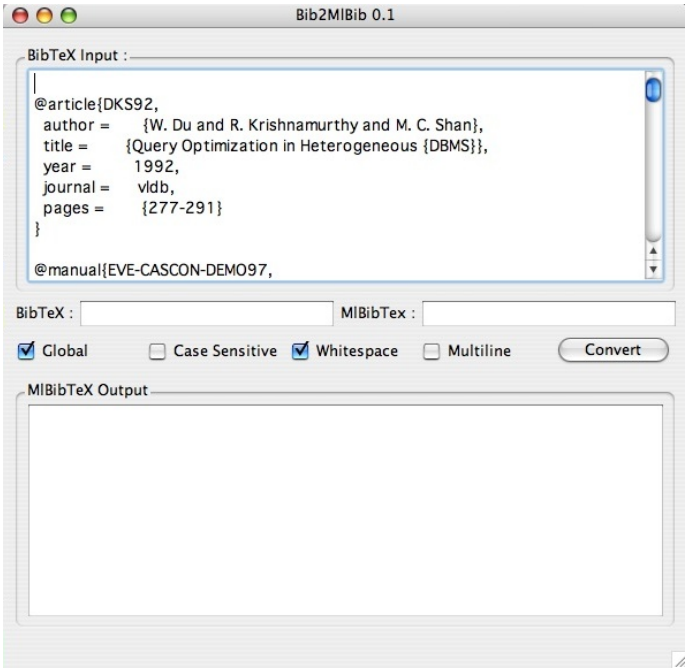


Figure 2: *The GUI of BibTeX2MlBibTeX 0.1*

[5] `.rb` is the extension used to identify ruby source files.

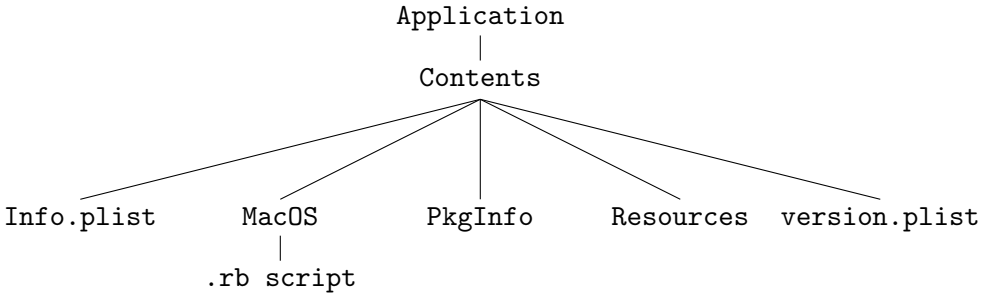
1.4.1 BibTEX2MlBibTEX: a cross-platform application

It would be nice to live in a world where there's only one platform to worry about. One platform to code on, the same platform to test on, and the very same platform to deploy on. While some developers can make the decision to do this, this project tries to build up an application all-platforms-capable. This project is developed in a context where the development of the applications are done on a rack of Linux or Solaris-based machines: that's why we think that choosing a Mac for our development is a great idea, as we have a lot of options for cross-platform development.

Basically, the GUI-based Qt application is developed under the Mac OS X platform. That's why the starting **.rb** script is developed only for the *Darwin – FreeBSD* kernel. Then, to make this script available for all platforms, Qt libraries by *Trolltech* are converted to the Linux platform and for the SunOS one.

Any program written in Ruby works correctly and equally well on Linux or Mac OS X. Eventually a simple change on the `make` file will be necessary, but, definitely twiddling the configuration variables is enough to get running with the application and its Qt background.

In our starting context (a Mac OS X system) we build up a *classical* `.app` application for the *Quartz* [6] environment that has, essentially, this architecture:



As you can see, from the `.app` strtucture, the `.rb` is intact and doesn't change from a platform to another. That's why a Linux users or a SunOS user can manage this script and its Qt interface simply by executing it without any major changes.

1.5 Results

The result of this project is a `rubyw`[7] application, perfectly handled by Unix-based environments. The interface we show up is a first implementation of what the end-user will look in the first release of this tool. As it is possible to see the interface integrates an *options* menu to help the user manage case sensitive expressions, white-spaces or multiline entries. BibTEX and MlBibTEX fields are there only to help the user by testing the input BibTEX expressions and their MlBibTEX output; this feature probably will go out in next releases. At the bottom of the

[6] The *Graphical User Interface* environment for Mac OS X users

[7] `rubyw` is a widget, part of the `rbuic` suite, and is installed by default on KDE systems

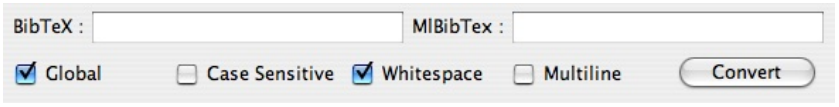


Figure 3: The *options* menu.

interface, the program integrates a *status* bar that allows the user to check the loaded file or the conversion status.

Figure 4: The *status* bar at the bottom of the interface.

As result the interface looks pretty good in this first release using the *Quartz* environmental objects (such as the Aqua[8] interface). The application works well and the conversion can be saved in an external document after being processed by the program.

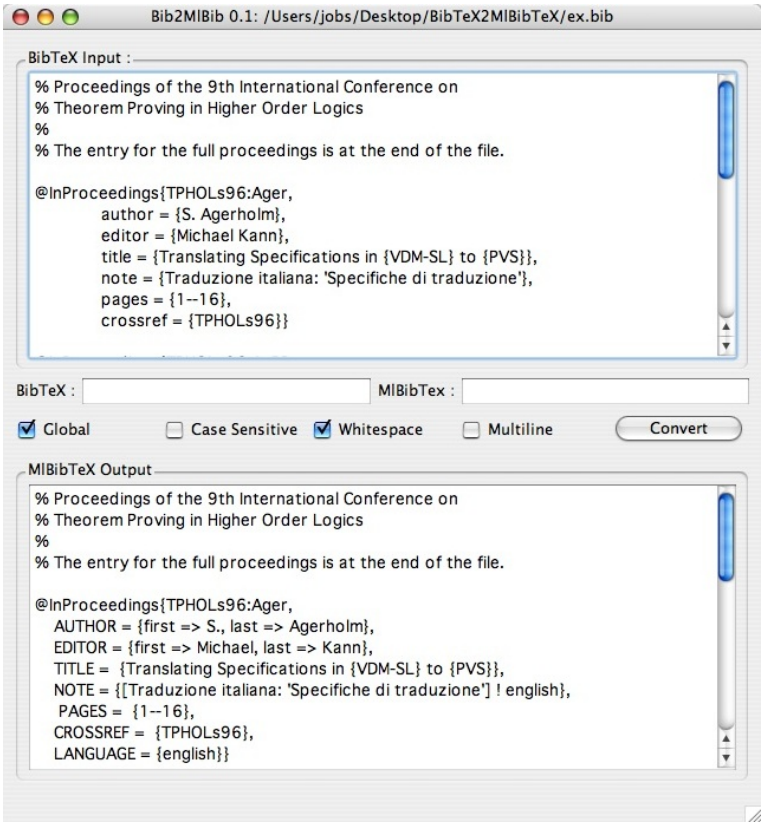


Figure 5: The final *GUI* of the *0.1* version of BibTeX2MlBibTeX

[8]The interface of Mac OS X, coming from the NextOS by NextStep

1.6 Discussion

Future developments of this application will include a debugging process to eliminate some of the problems that the user could find using the interface and the output of the program itself. The output for the MlBibTeX file generated by our application is still under examination to explore the multiple possible problems of its syntax.

2 Conclusion

Ruby is a highly-productive and industrial-strength program language application-oriented framework. It scales from the simplest expense tracking application we built to full-featured applications with respectable numbers of users. As with any useful tool, it's not suited to handle every job, but it's a great complement to the development environment. With these multiple qualities it is clear that it is a language perfectly adapted to interact with regular expression search and substitutions (as in our case). It implements a vastly rich environment also for the GUI programmer using the *Qt* libraries. The result of this project displays:

- a great utility for BibTeX users, supposed to deal with the BibTeX to MlBibTeX conversion
- a great flexibility of the developed application, thanks, in most cases, to the Ruby language cross-platform features
- a cross-platform application, based on the best technologies taken from every operating system

References

[1] Michel Goosens, Frank Mittelbach, and Alexander Samarin, *The LaTeX companion*, Addison-Wesley Publishing Company, Reading, MA, 1994.

[2] J.-M. Hufflen, *MlBibTeX : a new implementation of BibTeX*, Proc. of EuroTeX'2001 (Kerkrade, The Netherlands), September 2001, pp. 74–94.

[3] ———, *Names in bibtex and mlbibtex*, TUGboat, Volume 0 (2060) (2005), 1001–1011.

[4] ———, *BibTeX, mlBibTeX and bibliography styles*, Biuletyn GUST **23** (2006), 76–80 (English), in BachoTeX 2006 conference.

[5] ———, *Sprachen in mlBibTeX*, DANTE 2006 Tagung, Berlin (2006), –.

[6] Donald E. Knuth, *The TeX book*, Addison-Wesley, Reading, Massachusetts, 1984, Reprinted as Vol. A of *Computers & Typesetting*, 1986.