Programming bibliographies

Jean-Michel Hufflen

Abstract

We are interested in situations such that using the full expressive power of a programming language is needed when 'References' sections are generated for a source text suitable for LaTeX. The data model used by BibTeX is inadequate from this point of view; the biblatex package is based on a more efficient data model, but workarounds may be needed in some circumstances.

1 Introduction

Early computer programs were very different from those running nowadays. Sometimes they were written using programming languages with syntactical features now viewed as strange; they were running on computers whose performance was not comparable with today's. Besides, these early programs, in the 1950s, were only handled by people specialised in computer science: on the one hand, some rigid syntax was required for inputs, on the other hand, outputs were provided using raw forms, and graphical interfaces were nonexistent.

Things have evolved rapidly for years now, and nowadays many people are able to use programs and operating systems even if they do not have any knowledge about programming. A good example is given by interactive word processors (WYSIWYG[1]), such as Microsoft Word. Secretaries are able to use them, as end users simply typing input texts. Any end user of Word is able to customise it, mainly by means of graphical menus. So nowadays a rich collection of programs — including Word — can be used as any object of everyday life, like a washing machine or a dishwasher.

What is the point of non-interactive typesetting systems (WYSIWYM[2]), such as LaTeX, according to this point of view? In fact, if an end user only deals with default constructs or predefined document classes, LaTeX can be viewed as a kind of *black box* simply accepting input source texts and producing output texts. Some customisation and extension can be reached by means of *packages*, introduced by LaTeX 2_ε [10]. Stronger customisation is allowed by means of *commands*, written using TeX's programming language. Theoretically, this language has the same expressive power as a Turing machine's language, so any function can be programmed using it.[3]

In practice, TeX's language — which is based on *macros* — has been mainly designed to handle *text fragments*. As a kind of counter-example, even though we can implement a sort procedure using this language [9], that is a worthwhile exercise for its own sake, rather than an actual advantage for using (La)TeX. In fact, some 'pre-computations' — before typesetting a fragment — such as capitalising some words, or putting them using lower or uppercase characters, can be easily expressed using TeX's language, but more advanced features related to programming features are difficult to handle.

This point is one of the reasons why LuaTeX has been developed [2], allowing tasks more related to programming to be delegated to a more modern programming language. Since the initial versions of LuaTeX, (La)TeX is fully able to typeset the result of a computation performed by a program, for example, sorting an array before displaying its successive elements. Typesetting the results built by a spreadsheet program is another example.

In this article, we propose to apply such a view to *bibliography processors*, that is, generating 'References' sections for LaTeX documents. In other words, many end users should be able to use such a program without any knowledge about programming, but some specific applications may need such knowledge — provided that the format used by this processor is open — and it is important for such a bibliography processor to allow the direct programming of some specific functions.

In Section 2, we recall the tasks a bibliography processor should perform and we make precise our terminology. Section 3 briefly reports the main bibliography processors' current state. Then Section 4 goes thoroughly into some specific points. After some discussion, Section 5 concludes about our approach.

2 Tasks of a bibliography processor

Let us consider a source text (`.tex` file) including some bibliographical citations by means of LaTeX's `\cite` command, whose argument is a so-called *citation key*.

(i) A bibliography processor can use citation keys to *search* bibliography databases (`.bib` files) for corresponding *entries*.

[1] **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et.
[2] **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **M**ean.
[3] There are many online documents about this subject, more or less easy to read. A didactic written document is [6], showing how to implement a kind of λ-calculus in TeX.

doi.org/10.47397/tb/42-2/tb131hufflen-bibprog

(ii) These entries should be *sorted*, unless the document's bibliography is *unsorted*, that is, the order of the bibliography's items is the order of first citations throughout the document's body.

(iii) Finally, each bibliographical entry should be arranged into a bibliographical *reference* that future readers can consult, most often at the document's end, in which case they are stored in a `.bbl` file. Citation keys used throughout the document are replaced during this step: a reference may be identified by a number in *plain* bibliography styles, by an alphanumeric label in *alpha* styles. More advanced bibliography styles, such as *author-date* or *short-title*, have been successfully implemented within LaTeX, as reported in [11, Ch. 12].

3 Bibliography processors

For a long time, BibTeX [13] was the only bibliography processor usually associated with LaTeX. It uses information stored in auxiliary (`.aux`) files and is able to perform the steps (i), (ii) and (iii). BibTeX is still used, at least by some conference submission tools, although it is an old program, which does not address modern requirements such as multilingual encodings. BibTeX's bibliography styles have been implemented using the `.bst` language [12], sometimes complemented by a LaTeX 2_ε package within some advanced styles such as `natbib` or `jurabib` [11, Ch. 12]. This language for bibliography styles — based on handling a stack — is old-fashioned, more suitable for small changes than programming a new bibliography style. However, this language expresses general *algorithms*. That is, an end user developing a new style using this language has available the possible sequence of operations a computer can perform. Nevertheless, the same observation can be made as for using TeX's language: this `bst` language is difficult to handle practically.

During the last decade, a new *modus operandi* has been put into action by the biblatex package [8]: formatting bibliographical references — that is, most of step (iii) — is wholly controlled by TeX macros and deferred to LaTeX's next pass. Steps (i) and (ii) can be delegated to BibTeX, but biber [7] is preferred, because more features are available. The biber program does not use `.aux` files, but *control files* (`.bcf` files), built when the biblatex package is loaded with the option `backend=biber`. In addition, this biblatex/biber duo has introduced many new fields, many new bibliography types and styles; it seems widespread in the humanities.

The `bib` module of ConTeXt — another format built on top of TeX [1] — works analogously, in the sense that many tasks are deferred to ConTeXt's next pass; modern versions of this module are mostly implemented using the Lua programming language.

As another possible bibliography processor, we have personally put into action the implementation of a successor to BibTeX: MlBibTeX [3], written in the Scheme programming language. We think that a functional programming language is very suitable for this kind of task, because such languages allow the use of *functions as parameters*, e.g. in:

```
(lambda (f2) (f2 1 2))
```

To add (resp. multiply) `1` and `2`, just apply this expression to `+` (resp. `*`). A bibliography style can be compared to this expression, because the *resources* are known — they are the successive bibliographical entries — whereas the way to arrange these data together is to be determined w.r.t. the bibliography style chosen. MlBibTeX has been developed according to such an approach. It has been used rather occasionally, but as far as we know, using it has always proven successful. In particular, it has been used to populate the French official site for open archives from a collection of `.bib` files, as reported in [4]. A new version has been announced in [5]; its new features include a better interface with Scheme definitions. This point seems to us to be important and we are going to explain why.

4 Some operations

4.1 Sorting

Let us consider again the tasks enumerated in §2. Step (ii) — sorting the extracted bibliographical entries — would be very difficult to program in TeX's macro language, as mentioned in the introduction. Although the biblatex package tends to use LaTeX to perform as many operations as possible, this sort step is still performed by the bibliography processor associated, BibTeX or biber. The former provides a 'basic' lexicographical sort procedure, in practice only suitable for the English language (without manual adjustments). The biblatex package allows some customisation of the sort procedure performed by biber, by means of the `sorting` option and *mnemonics* denoting sort keys — e.g., `nyt` is a sorting scheme for name, year, and title [8, §3.1.2.1].

Neither of these two processors can perform a numeric sort:[4] with these two bibliography processors, the year information is correctly processed because years are supposed to use the same number of digits. Let us go thoroughly into chronological

[4] . . . although biblatex's documentation is very precise about the *types* used within fields.

order: BibTeX does not sort w.r.t. the month information; biber could do that, since the month information handled by biblatex consists of numbers,[5] but in practice no predefined sorting scheme does so.

The biblatex/biber duo does offer more ways to sort bibliographical entries in comparison with 'old' BibTeX. Moreover, its sorting schemes allow people unfamiliar with computer science to specify precise sort keys, but:

- a *descending* sort can be applied only to years within predefined schemes;[6]
- the number of *person names* — for authors or editors — considered for sorting — given by the `maxnames` constant [8, §3.1.2.1] — is limited;[7] the sort procedure does not deal with *unbounded* number of person names, even if the maximum number considered can be changed by end users;
- some bibliographical fields mainly used as sort keys — e.g., `AUTHOR`, `TITLE`, `YEAR` — may be substituted by sort-suitable fields during this procedure — `SORTNAME`, `SORTTITLE`, `SORTYEAR`: excess markup for corresponding information is avoided, but there is some risk of *information redundancy*;
- some exotic sorts can be handled by redefining the beginning and end of the sort procedure, by means of the bibliographic fields `PRESORT` and `SORTKEY` [8, §3.6]; more difficult cases can be processed by defining new sorting schemes by means of the `\DeclareSortingTemplate` command [8, §4.5.6] — to be put in a source `.tex` file — but this last command defines intermediate sort keys by means of additional fields rather than new sort procedures.

In our personal opinion, the biblatex package's conventions allow people unfamiliar with programming to deal with a rich collection of possible sorts, but if you need to add a new sort procedure, you do not have the full expressive power of a programming language.[8] As an ambitious example, we personally were in charge of the publication list of our laboratory some years ago: we had to sort this list first by research teams, second by *categories*,[9] third by decreasing years, fourth by authors' names (increasing), fifth by decreasing months. That would have been possible with biblatex/biber but quite difficult and/or requiring some information redundancy.

4.2 Labelling references

The automatic generation of non-ambiguous bibliographical keys for *references* — the keys that will appear within the resulting document — is now satisfactory. Let us remark that if end users would like to use their own keys, they should use a *short-title* system.[10] If we consider *alpha* styles, we would like to point out that such a style does not belong to the author-date system. BibTeX and biblatex add a letter if several references share the same author and year, that is:

[Rob 1964a] Kenneth Robeson. *The Man of Bronze*. No. 1 in *Doc Savage Series*. Bantam Books, October 1964.

[Rob 1964b] Kenneth Robeson. *The Thousand-Headed Man*. No. 2 in *Doc Savage Series*. Bantam Books, October 1964.

According to another approach, we can label a first or unique reference with a name's abbreviation and a year. If need be, a possible second reference with the same information is given an additional letter, that is:

[Rob 1965] Kenneth Robeson. *The Polar Treasure*. No. 4 in *Doc Savage Series*. Bantam Books, April 1965.

[Rob 1965a] Kenneth Robeson. *Brand of the Werewolf*. No. 5 in *Doc Savage Series*. Bantam Books, April 1965.

The first way is often used in modern documents, but the second can be observed. Our opinion is that this problem can be solved with access to the function generating successive labels within the bibliography processor.

5 Conclusion

We think that MlBibTeX's new version will be finished at this year's end. We expect to provide an open system, comparable with the Emacs[11] editor. That is, usable as it is, but also *customisable* and *extensible*. As mentioned in the introduction, LaTeX

[5] Two-digit numbers, of course.

[6] To do this for new sorting schemes, use the `\sort` construct with the option `direction=descending` inside a `\DefineSortingTemplate` command [8, §4.5.6].

[7] That is the same within 'old' BibTeX, because a unique string is built as a sort key for each entry. So the number of co-authors or co-editors used during BibTeX's sort procedure is limited by this string's length.

[8] ... even if you can program using Perl, the language used for biber's implementation.

[9] That is, articles in well-known international journals and in other international journals, articles in journals having 'national' scope, papers in conferences, etc.

[10] With 'old' BibTeX, end users can use the predefined `KEY` field with a suitable bibliography style. But it seems that such a *modus operandi* has very rarely been put into action in practice.

[11] **E**diting **MAC**ros.

has these qualities, and the success of LuaTEX shows that the addition of a modern programming language's power has opened new windows.

Even if not all of a bibliography processor's customers are computer scientists, tasks performed by such a tool belong to programming. When the Unix operating system emerged, the idea was that each person using it would be close to someone who precisely knew how Unix worked. That is, a kind of *synergy*. We think that a comparable synergy should be reached for an open bibliography processor, which implies that the structures it uses are open. This was already the case for MlBibTEX's last version. Likewise, most of the new fields and types introduced by biblatex will be recognised and handled by MlBibTEX's last version, so we can claim that we are not in opposition to biblatex. We aim to bring another view of what a bibliography processor should be.

Acknowledgements

It is a great pleasure to thank the first version's proofreaders, Barbara Beeton and Karl Berry.

References

[1] ContextGarden: *Bibliographies in MkIV*. July 2012. https://wiki.contextgarden.net/Bibliography_mkiv.

[2] Hans Hagen: "LuaTEX: Howling to the Moon". *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, vol. 23, pp. 63–68. April 2006.

[3] Jean-Michel Hufflen: "MlBibTEX's Version 1.3". *TUGboat*, vol. 24, no. 2, pp. 249–262. July 2003. https://tug.org/TUGboat/tb24-2/tb77hufflen.pdf

[4] Jean-Michel Hufflen: "Using MlBibTEX to Populate Open Archives". In: Tomasz Przechlewski, Karl Berry, Gaby Gic-Grusza, Ewa Kolsar and Jerzy B. Ludwichowski, eds., *Typographers and Programmers: Mutual Inspirations. Proc. BachoTEX 2010 Conference*, pp. 45–48. April 2010.

[5] Jean-Michel Hufflen: "From MlBibTEX 1.3 to 1.4". In: Tomasz Przechlewski, Karl Berry, Bogusław Jackowski and Jerzy B. Ludwichowski, eds., *Various Faces of Typography. Proc. BachoTEX 2015 conference*, pp. 13–17. Bachotek, Poland. April 2015.

[6] Alan Jeffrey: "Lists in TEX's Mouth". *TUGboat*, vol. 11, no. 2, pp. 237–245. June 1990. https://tug.org/TUGboat/tb11-2/tb28jeffrey.pdf

[7] Philip Kime and François Charette: *biber. A Backend Bibliography Processor for biblatex. Version biber 2.16 (biblatex 3.16)*. 19 December 2020. https://ctan.org/pkg/biber.

[8] Philip Kime, Moritz Wemheuer and Philipp Lehman: *The biblatex Package. Programmable Bibliographies and Citations*. Version 3.16. 31 December 2020. https://ctan.org/pkg/biblatex.

[9] Kees van der Laan: "Sorting within TEX". *TUGboat*, vol. 14, no. 3, pp. 319–328. October 1993. https:///TUGboat/tb14-3/tb40laan-sort.pdf

[10] Leslie Lamport: *LATEX: A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[11] Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LATEX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[12] Oren Patashnik: *Designing BibTEX Styles*. February 1988. Part of the BibTEX distribution. https://ctan.org/pkg/bibtex

[13] Oren Patashnik: *BibTEXing*. February 1988. Part of the BibTEX distribution. https://ctan.org/pkg/bibtex

◇ Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) &
University of Bourgogne Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
jmhuffle (at) femto-st dot fr
members.femto-st.fr/Hufflen-Jean-Michel/en