Software & Tools

CAMEL: Kicking over the bibliographic traces in LaTeX

Frank G. Bennett, Jr.

1 Introduction

It gives me great pleasure to introduce readers to a prototype citation manager called CAMEL. This package, as it now appears on CTAN, is work in progress of the LaTeX3 bibliography team. The implicit goal of this project is as simple as it is precocious: to create a new citation interface for LaTeX that is simple, intuitively straightforward, and allows the bibliographic style of a document to be changed at will without major editing. In effect, we are seeking to implement logical markup for citation styles, cross-referencing schemes, and bibliography formats. If the savings in editorial time that this implies are of interest to you, read on.

While the long-term goal of our work is to produce a fresh standard bibliography package that will be widely used and well supported, I am duty bound to stress that CAMEL is currently a *prototype*. It pains me slightly to say so, because I am only too painfully aware of the dilemma that faces the would-be bibliographic revolutionary:

1. The designer needs real-world trials to discover what the world's formatting requirements actually are. This is an inductive, not a deductive process.
2. If the system is to be efficient, it must be tightly integrated. It therefore often will not be susceptible to quick fixes, and may undergo significant change in the course of its development.
3. Automated bibliography management is useless if it does not provide a complete solution to formatting problems. Until the system is complete, no one will use it for fear of holding up production or sinking work into text that relies on an unstable syntax.
4. Go to 1.

It remains to be seen whether CAMEL will break out of this circle of reticence, but before it attempts to do so it must mature further. That will require feedback. And the solicitation of feedback is one of the main purposes of this article.

That we have been able to overcome the first hurdle at all — the drafting of a working prototype of the new system — has been possible largely because of the diversity of inputs made by each of the

core contributors to the bibliography team. Pedro Aphalo, the first task coordinator for bibliography support in the LaTeX3 project, was first to propose an integrated interface for handling bibliographic formats. David Rhead carried out careful research into the typesetting needs and available tools beyond the TeX world, the results of which appear in (Rhead, 1993). I joined the team after writing the LexiTeX package, described in (Bennett, Jr., 1993), to meet my own needs as a law lecturer. Prompted in large part by new possibilities opened up by Oren Patashnik's commencement of work on BibTeX 1.0, I sought to bring my then-existing code up to, or at least close to, the standard that had been set by Pedro and David. The result was been Camel; and when the draft package hit CTAN, Matt Swift provided valuable feedback on the needs of editors of critical editions and works of literary criticism.

Camel thus arose at the confluence of two distinct streams of development, one based on the identification of general "operational requirements" in advance of code drafting, the other on the persistent refashioning and honing of what, as the person responsible for it, I can safely describe as ad hoc TeX hackery. Although the code of Camel was fashioned by me, incorporating my own preferred design choices and, no doubt, infelicities of design, even this modest effort could not have been accomplished without the groundwork and feedback offered by the other members.

The next two sections of this paper provide a descriptive overview of the LaTeX and the BibTeX user interfaces in the Camel prototype. This is followed by a final section which discusses various parts of the system which are either worthy of attention because they attempt to do something new and perhaps useful, or because they may be changed significantly in future releases. In particular, readers may wish to note the rather drastic structural changes that may become possible, following discussions with Oren Patashnik about his work on BibTeX 1.0.

2 The LaTeX User Interface

In a 1993 article in this journal (Rhead, 1993), David Rhead specified the fundamental requirements for a new citation management system, and suggested ways in which LaTeX3 might address these requirements. I have entirely accepted the specification, but have adopted a rather different method of addressing it from that recommended. The core of the specification lists five reasonably common forms for handling reference lists and cross-referencing:

1. Reference by number, with the reference-list in alphabetical order of author's names;
2. Reference by number, with the reference-list in order of first citation;
3. Reference by author and date of publication, with the reference-list sorted accordingly;
4. Short-form citations in footnotes, which refer to a reference list sorted by author and date of publication; and
5. Citation in footnotes, with the first citation giving full details.

He also noted two important capabilities that are currently beyond the capacity of vanilla LaTeX, at least where documents are typeset as a unified whole:

- The user should be offered the option of dividing the reference list into categories; and
- A document should be able to have separate reference lists for each section.

This was further supplemented by Matt Swift, commenting on the current release of Camel:

- The user should be free to place the reference list before or after the place at which the citations occur.

David's suggestion in 1993 was that (a) ample provision should be made for formatting both citation tags and the reference list using commercial bibliography support software, rather than BibTeX; and (b) unorthodox citation styles with special requirements, such as in-footnote citations without a reference list, should be treated by a separate "plug-in module" to reduce the bulk of the code, and the volume of documentation.

The arguments in support of these suggestions were pragmatic; David was concerned that the burden of supporting a wide variety of styles might overwhelm the voluntary efforts of LaTeX's maintainers. Nonetheless, Camel seeks to offer a unified interface for all bibliographic styles: one set of commands, one list of options, one user's guide. And BibTeX is at the core of its functions. Since bibliographic management involves the manipulation of data by a series of tools, it seems to me that the simpler and the more well-defined the interfaces for the exchange of data between those tools can be made, the more flexible the package will ultimately be. I therefore settled on an approach that diverges from David's original suggestions.

Much work has gone into the streamlining of the LaTeX user interface for Camel. In it, all requirements are met with just eight commands:

```
\citationstyle{<style_name>}
```

```
\citationdata{bib1,bib2...}
\bibliographymanager{<manager_name>}
\citationsubject[<opts>]{<subj>}{<header>}
\source[<opts>]{<nickname>}[<pages>]
\forcefootnotes
\newinterword{<word>}{<text>}
\printbibliography{<subj>}
```

The use and effect of these commands are explained in the CAMEL manual; I will give only the briefest of descriptions here to indicate their function.

The `\citationstyle` command is required in all documents. Much like `\documentclass`, it tells the bibliography manager what package of code to use in its formatting work. Currently only one style, `law`, exists. This command is accepted only within the scope of the `document` environment, and it may be used repeatedly, to change bibliographic styles on the fly for different sections of a document.[1]

The user may specify a database either using the `\citationdata` command or, alternatively, by using the `\bibliographymanager` command. The former serves a function similar to the `\bibliography` command in standard LaTeX bibliography processing; it tells BibTeX which `*.bib` files it should search for citation entries. The `\bibliographymanager` command accepts one argument, which is the name of an external bibliography manager that is used for the citations in the document. Currently, the command will accept the names of the "main 4" identified by David Rhead (1993, p. 27): EndNote, Papyrus, ProCite and Reference Manager. When this command is in force, LaTeX will write citation keys, delimited in the way required for the designated bibliography manager, to a special file.[2] The bibliography management software must then be used to perform a simple "find and replace" operation, to replace the keys with `*.bib` entries for the designated keys. This adds another step to processing, but requires a minimum of special setup in the alien management software and can be covered by generic documentation.

The `\citationsubject` command is used to create classified bibliographies. More than one such command may be issued for each bibliography. The mandatory arguments are the nickname of the subject category, and the actual text to use as a header when the subject category is printed in the bibliography. Best practice is to place the `\citationsubject` commands at the top of the document, in the order in which you wish the subject categories to be produced in the bibliography.[3] The optional argument may mark a subject as a sub-category of the immediately preceding subject. It may also indicate whether or not page references — to instances of the citation in the current document — should be included. In the latter case, input and output extensions must be given, to allow the data produced to be processed using `\makeindex`.

In CAMEL, the new `\source` command replaces the `\cite` command in the standard LaTeX bibliography interface. In its simplest form, the command can be invoked with the nickname of a command in the BibTeX nickname of the desired citation as its sole argument. The page number of the citation can be placed in square braces immediately after the citation. Page numbers may be separated by commas, ampersands or single hyphens (spaces are ignored). Alphabetic characters may be included, but the page or section prefix is supplied by the style; the user only needs to worry about providing the numbers.

Further options may be put in square braces between the `\source` command itself and the braced nickname. These options may be used to suppress the author's name or the title of the work where appropriate,[4] to append a volume number for a multivolume work by a single author, to specify a citation subject category by its nickname, and so forth. The citation subject category must be given for the `\source` command if, at that point in the document, one or more `\citationsubject` commands are in force.[5]

The `\source` command can be given a list of keys, as in:

```
\source{item1,item2,item3}
```

In this case, leading options and trailing page numbers will be ignored, and a default syntax for joining characters is used. Alternatively, `\source` commands may be joined using a single bridging word:

```
\source{item1} but-see \source{item2}
```

A chain of `\source` commands joined in this way will expand as a single macro, and the bridging words will expand into a phrase with text, punctuation and typefaces determined by the style package in force at the time.[6]

[1] This function is not yet available in the prototype.

[2] The naming convention for this file has not yet been fixed. Currently it is always called `camel.bib`.

[3] A sample is included in a test file contained in the CAMEL distribution.

[4] These options will, of course, only take effect in a short-form citation style.

[5] The plan at the moment is to make the assignments relating to citation subjects local, so that their scope can be limited by enclosing them in an environment.

[6] This facility is available in the prototype. A facility for providing easy access to the list of bridging words is not yet in place, and needs to be installed.

The `\forcefootnotes` command causes LaTeX to push the expanded text of all `\source` commands into footnotes. This will be useful, for example, for authors who wish to be able to submit an article alternatively to a social science journal that requires an author-date reference list, or to a law journal that requires in-footnote citations with no reference list; printed output in both forms can be produced from a single source file, simply by changing the `\citationstyle` declaration — assuming, of course, that a Camel style package for each journal's style exists!

The `\newinterword` command is used to create bridging words and matching text output that are not built into the Camel distribution. Existing definitions can also be overridden using this command.

Finally, the `\printbibliography` command is used to explicitly place the bibliography at a particular location in the document. This command takes a single argument, which is either the word `all`, or the nickname of a subject category. Any categories which have been associated with an input and output extension will be read from the designated file.

The eight commands of the Camel user interface open the possibility of logical citation markup of LaTeX documents, with the reduction in time spent on wheel-reinvention and the training of oneself or one's staff that that entails. All that will be required (he said) is the development of Camel style packages for all users' requirements.

3 The BibTeX User Interface

Camel uses BibTeX to generate a reference list that is carried in memory throughout the LaTeX run.[7] BibTeX is applied to the `*.aux` file in the normal way, following the first LaTeX run; but each entry written on the `*.bbl` file is written as arguments to a special control macro. To handle certain special requirements, the path of least resistance was to redraft a `*.bst` style file for Camel from scratch. In the course of doing so, I introduced certain enhancements, some of which are essential features of the new system, while others are optional extras which may or may not survive the test of community criticism.

3.1 New Entry Types

A new entry type, `@CASE`, has been added, for use in citing reported law cases. Because the content of citations to case reporters varies widely between jurisdictions, Camel uses only this one entry type, and varies the format of the citation according to its content. A result of this approach is that there are no 'required' fields in the usual sense for `@CASE` entries — BibTeX will not complain if something is missing. Instead, there is a set of 'core' fields for each of four different formatting styles. For someone familiar with legal resources, this is actually quite intuitively straightforward.

For reported United States law cases, the core fields are `title`, `volume`, `journal`, `pages` and `year`. In addition, you may wish to specify `court`.[8] Procedural histories cannot be represented in the BibTeX entry; this must be done explicitly in the text of the LaTeX document.

For reported Commonwealth cases, use `number` instead of `volume`. The effect of this will be to place the year at the front of the citation in square braces, with the `number` and `journal` following it. Again, you may specify `court` optionally.

For cases reported in ephemeral media such as newspapers, leave out `volume` and `number`, and give the full date in the `year` field instead (see below for the formatting of dates). The formatting of the citation will adjust accordingly.

For cases reported in those jurisdictions, such as Japan, which refer to cases by date rather than by title, the citation should include `casedate`, `court`, `journal`, `volume`, `pages` and `year`. Optionally, you may also wish to include `divno` and `division`, to specify the exact identity of the deciding court.

There is a `@STATUTE` entry type, but support for statutes is still in its infancy. You need, at minimum, to enter `title`, `year` and `jurisdiction`. Jurisdictions supported in this entry type so far include `japan`, `singapore` and `england`. The present arrangement is not particularly satisfactory; and I am certainly open to suggestions on how best to manage this type of entry.

3.2 Parsing of Fields

The entry of dates has been considerably simplified in Camel BibTeX entries. Always use the `year` field (or, if appropriate, the `casedate` field). Months may be entered as numbers or as a three-character string. The following date forms are all valid:

```
year = {10 jun 1995},
year = {jun 10 1995},
year = {jun 1995},
year = {1995},
year = {10/06/95},
```

The prototype reads only the `year` field, and ignores `month`; this will be repaired in due course.

[7] The `harvard` bibliography package works in a similar way.

[8] This is not yet implemented, but can be and will be.

For `@CASE` entries, it is possible to enter multiple citations in a single BibTeX entry using a short form of citation in a field called `cites`, separating entries with an `=` character. The syntax of this field is rather flexible. There must be a core of one or more words, giving the name of the journal. This must be followed, at least, by a page number. The journal name must be preceded either by a volume number standing on its own, by a volume number followed by a slash and an issue number, or by a year in braces, followed by an issue number for that year. Where the year is not given in braces before the citation, it should be give in parens at the end. For the citation of mainstream case reports, these forms closely follow the citation conventions used in law journals:

```
cites = {[1995] 1 All ER 25
       = [1995] 2 WLR 125}

cites = {123 Cal.3d 237 (1995)
       = 124 S.W.2d 235 (1995)}
```

You can string together an arbitrary number of parallel citations in this way. The `law` style will correctly insert any page pinpointing information given to the `\source` command into the citation in the appropriate places.

4 The Future

This section discusses some of the features of the Camel system that are most interesting, or most in need of further attention before a proper production version is released. I will not delve into the specifics of the code — Camel is on CTAN for anyone who wants to take a look at it — but will indicate why I feel a particular portion of the code is worthy of examination, or in need of further attention. This will, I hope, give the reader a clear sense of how settled or unsettled the code is at this phase of its development.

4.1 Things unlikely to change

The most complex set of code in Camel is the expansion of the `\source` command. Apart from incidental concerns about the code used for parsing options (see below), I am very happy with the performance of `\source` and its friends. The key to its robustness is a hidden command which I have named `\@ifoverword`, which was developed for Camel on the basis of the `\@ifnextchar` command in standard LaTeX2$_\varepsilon$. This command, like `\@ifnextchar`, takes three arguments: a comparison token, an argument to execute if the comparison is true, and an argument to execute if it is false. The macro absorbs all text following itself, up to a space. It then reads the following token, and returns true if it matches the comparison token, false otherwise. Within the true-text and the false-text arguments, the string that is swallowed to get at the text token used in the comparison is available in the macro `\@overword`.

This command is combined with a set of routines that iteratively store nickname keys and their matching arguments to list macros, so long as additional `\@ifoverword` matches continue to be found. These list macros are then expanded in one go, after the last match. Thus a string of `source` commands behaves in all respects as a single macro. Without this facility, the `\forcefootnotes` command would not be sufficiently robust in its effect to be of use in real-world typesetting.

Page number parsing is, I think, pretty satisfactory as written. The optional page number argument is scanned character-by-character. Each is compared with the keys in a parsing list. Those that do not match are pushed onto the end of a string, while those that do (i.e. commas, ampersands and single hyphens) are replaced with an appropriate macro before being pushed. Handled in this way, these characters behave as if they were active, but without any change to their category codes. The `\source` command is therefore robust, and should not break — unless some other style alters the category code of these characters.

I am also happy with the error handling routines in Camel. For a complex facility like bibliography support, detailed and specific error messages are important, and Camel does provide these where I felt help to be most clearly needed. The one qualification to this is that the error message macros do use up a large number of tokens — this while the core LaTeX3 team have been working overtime trying to reduce, one by one, the number of tokens used in the LaTeX2$_\varepsilon$ kernel. This problem will eventually be solved when the planned facility for drawing help text from external files is introduced.

For legal citations, Camel must be capable of identifying the logical context of the citation. Is this its first occurrence in the document? Is there another citation by the same author in this document? Was the immediately preceding citation to the same work? If that immediately preceding citation was in a different footnote, was it the sole citation in that footnote? And so forth. These routines have been specified with great care, and are now reasonably reliable. The intention is to embed these routines in the "kernel" of the final Camel system, so that the result of their evaluation can be

drawn upon by any style that requires information concerning the context of a citation.

Finally, the structure of the CAMEL code has improved considerably over time. Further work is required, particularly in adding and updating commentary, but the functions performed by the package have been clearly divided into sub-units, and the internal interfaces between those units are well-defined. Further clarification of the code will be carried out, because I must involve other people in the code itself if CAMEL is to grow.

4.2 Possible minor changes

Options are currently specified using parsing routines that I cooked up in ignorance of the `keyval` package by David Carlisle. In due course, I plan to shift to `keyval`-based parsing. This will slightly alter the interface, by allowing mnemonic names for options to be used, instead of the single-character flags in the current prototype.

There has been some criticism of the placement of page numbers after the citation nickname in the `\source` command. I have set things up in this way because special parsing routines must be applied to page number strings. Commas, in particular, have a special significance, and would clash with the function of commas in `keyval` options unless the page number string were enclosed in braces. On balance, I feel that the current arrangement is less cluttered and easier to use. I am open to suggestions and criticism on this, however, and would welcome feedback.

4.3 A possible major change

The drafting of BIBTEX 1.0, the final release, is currently being carried out by Oren Patashnik. Version 1.0 will include significant enhancements, some of which are outlined in (Patashnik, 1994). There is a possibility (at present no more than a possibility) that the finished BIBTEX 1.0 will include a facility for producing "citation listings". If this facility becomes available, it will lay the groundwork for dramatic simplification and efficiency improvements in the CAMEL system. All that will be required (he said) is the complete rewriting of large sections of its code.

In order to cope with legal citations, CAMEL must alter the form of citations depending upon certain conditions, as described above in Section 4.1. The way this functions at present is for LATEX first to pass a list of citations to the `*.aux` file, for BIBTEX to generate a `*.bbl` file containing "`\lexibib`" commands and arguments including pre-parsed citation elements. The `\lexibib` commands are read in by LATEX during the next run. Their effect is to create a set of list macros in TEX's memory, one for each citation. Each `\source` command is then expanded into the form appropriate to its logical context, using the details stored in these list macros.

This approach is wasteful of memory. A citation that occurs only once will be carried in memory throughout the second LATEX run. For works, such as a legal textbook, that may contain thousands of citations, the cost in memory will be excessively large. And on machines with limited memory capacity, CAMEL will simply break on large jobs. Unfortunately, this situation cannot be avoided with current tools; BIBTEX is designed to produce *reference lists*, writing one entry on output for each unique citation key. To use that information more than once — to be able to refer to a source more than once — all of the details that might be used must be stored in memory. In a short-form citation style, that means that *everything* must be stored in memory.

If a facility for producing citation lists were introduced into BIBTEX, a style could instruct it to produce, in addition to ordinary pre-sorted bibliography listings, a listing of citations, one per line, marked up ready for immediate printing through LATEX, without the need for any further parsing or post-processing. The latter would be read in one line at a time by the `\source` commands contained in the document.

This implies that (a) most of the context-identification and selective formatting work currently carried out in LATEX would need to be recast in BIBTEX code, and (b) there must be a method for passing the essential information concerning context from LATEX to BIBTEX. This is not as daft as it may at first seem; the interaction of BIBTEX and LATEX is quite difficult to grasp in standard LATEX. It is even more difficult, if anything, in CAMEL. My work on the `*.bst` library for CAMEL suggests that the postfix stack language used by BIBTEX can be tamed by the introduction of a library of high-level functions. And if all text formatting is carried out in a single forum governed by a single language, it will be much easier to see what it going on, and to work out bugs that appear in the course of the system's use. Apart from this, such an approach would have the following advantages:

- The bulk of the CAMEL style code in LATEX would be substantially reduced. All of the code relating to the selection of citation elements and bridging punctuation would be eliminated.
- The marginal cost in memory and TEX tokens of each additional citation would be *zero* — less even than in the standard bibliography configuration of current LATEX 2_ε.

References

Bennett, Jr., Frank G. "LexiTeX: context-sensitive legal citations for LaTeX". *TUGboat* **14**(3), 187–195, 1993.

Patashnik, Oren. "BibTeX 1.0". *TUGboat* **15**(3), 269–273, 1994.

Rhead, David. "The "operational requirement" for support of bibliographies". *TUGboat* **14**(4), 425–433, 1993.

◇ Frank G. Bennett, Jr.
Law Department
School of Oriental and African Studies
University of London
Thornhaugh Street
London WC1H 0XG
U.K.
fbennett@rumple.soas.ac.uk