Names in BibTeX and mBibTeX
Jean-Michel Hufflen

To cite this version:
Jean-Michel Hufflen. Names in BibTeX and mBibTeX. TUGB, 2006, pp.243–253. hal-00644463

HAL Id: hal-00644463
https://hal.science/hal-00644463
Submitted on 24 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Names in \texttt{BibTeX} and \texttt{MlBibTeX}

Jean-Michel Hufflen
LIFC (FRE CNRS 2661)
University of Franche-Comté
16, route de Gray
25030 Besançon Cedex
France
hufflen (at) lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

Within the bibliographical entries managed by \texttt{BibTeX}, the bibliography processor usually associated with \texttt{LaTeX}, person and organisation names are specified with a rough syntax, whose details are not very well known. Likewise, the features related to formatting names within bibliography styles are often viewed as obscure. We explain these points in detail, giving some cases difficult or impossible to handle with \texttt{BibTeX}. Then we show how these problems can be solved within \texttt{MlBibTeX}, our reimplementation of \texttt{BibTeX} focusing on multilingual features and using an extension of XSLT as the language for bibliography styles.

Keywords \texttt{BibTeX}, \texttt{MlBibTeX}, Bibliographies, bibliography styles, specifying and formatting person names.

1 Introduction

Specifying meta-information related to persons is a difficult problem within databases from a general point of view. It is well known that the parts constituting the name of a person are insufficient to characterise only one person. However, we do not go thoroughly into this point and only focus on names we can find within bibliographical databases. Managing this information is crucial: it may be used for searching bibliographical databases, and many bibliography styles sort references w.r.t. alphabetical order of authors or editors.

When users typeset documents with the \texttt{BibTeX} word processor [15], searching bibliographical databases for citations and assembling references into a ‘Bibliography’ section, put at the end of a printed document, is usually done with the \texttt{BibTeX} bibliography processor [20]. Formats for names are defined as part of bibliography styles: first names may be abbreviated or put \textit{in extenso}, they can be written before or after the last name, . . . \texttt{BibTeX} uses a rough syntax for specifying authors’ and editors’ names within bibliographical entries. This syntax is suitable for simple cases, and powerful since it includes many interesting and useful features, sometimes not well known due to their complexity. In addition, some of these features are documented only partially, as far as we know, although many details are given in the second edition of the \textit{\texttt{BibTeX} Companion} [16, § 13.2.2]. Likewise, the primitive \texttt{format.name$} function, which formats names within \texttt{BibTeX}’s bibliography styles [19] is powerful but uses quite complicated patterns, documented only partially. In fact, only a small fraction of its expressive power is used in practice.

So we begin by describing \texttt{BibTeX}’s syntax for names, with its advantages and limitations. We also show how names are formatted, as precisely as possible. Then we explain how these points have been improved within \texttt{MlBibTeX} (for ‘\texttt{MultiLingual BibTeX}’) [7], our reimplementation focusing on multilingual aspects. Last, we discuss aspects of internationalization.

Reading this article requires good knowledge of \texttt{BibTeX} as an end-user, and a bit of experience with bibliography styles, that is, a good knowledge of the notions described in [16, § 13.6]. We will recapitulate more technical points, however.

2 How names are processed by \texttt{BibTeX}

2.1 Names’ components

Inside \texttt{AUTHOR} and \texttt{EDITOR} fields, \texttt{BibTeX} allows the specification of successive names, separated by the ‘and’ keyword, as shown in Figure 1. A large list of names that is not typed \textit{in extenso} is ended with ‘and others’:

\{Karl-Heinz Scheer and Clark Darlton and others\}
2.2 Using braces

If the first letter of a word is surrounded by braces, it is supposed to be uppercase, unless it follows a TeX command and is lowercase. Here are some examples:

- Alfred Elton \{van\} Vogt
- Alfred Elton \{\relax van\} Vogt
- \{\relax is a dummy command [14, Ch. 24]),
- Alfred Elton \{\relax Van\} Vogt

Remark 1 Non-letter characters are ignored when Bib\TeX determines whether the first letter of a word is lower- or uppercase. That is why it considers ‘\{-\}' (see above) to begin with a lowercase letter, ignoring the hyphen sign between braces.

As mentioned in [16, § 13.2.2], enclosing some characters between braces serves four purposes:

- treating accented letters — accented by means of \TeX commands — as single letters:
  \begin{verbatim}
  Christian Vil\{'(a)\'}
  \end{verbatim}
- treating multiple words as one, especially when a component (Last) consists of a single word by default:
  \begin{verbatim}
  Michael \{\textit{Marshall Smith}\}
  \end{verbatim}

this feature being commonly used for specifying an organisation name, which consists of only a Last part:

\begin{verbatim}
EDITOR = \{\textit{Science Fiction and Fantasy Writers of America, Inc.}\}
\end{verbatim}
- treating ‘and’ as an ordinary word (see the example above);

---

1 Only the following name is imaginary (although it is derived from a French poet’s name). All the others — some being pseudonyms — name real persons, even if some look strange. That is, the problems raised by Bib\TeX in the examples we give may arise in real situations.

2 In the following, we do not consider the braces that might delimit the value of a Bib\TeX field. ‘Braces’ should be understood as ‘additional braces’ in such a case.

3 ‘\textit{and}’ is viewed as a keyword only at the topmost level, not surrounded by additional braces.
• delimiting substrings which should not change case when the `change-case$` function [16, Table 13.8] is applied.\footnote{ Although this use is rare within \texttt{AUTHOR} or \texttt{EDITOR} fields. See more details in Figure 3.}

2.3 Words vs tokens

In the above subsections, ‘word’ has been used in the common sense, that is, ‘an independent unit of the vocabulary of a language’ [18], so that adjacent words are syntactically separated by space characters or punctuation signs. In fact, if we wish to characterise this notion in the sense of successive tokens handled by \texttt{Bib\TeX}, inside a string representing a name, the separators are whitespace characters\footnote{ This terminology originates from the Scheme programming language [11, \S 6.3.4]: such characters are recognised by the `char-whitespace`? function of this language.} — space, tabulation, line feed, form feed, and carriage return — unbreakable space characters, specified by ‘\texttt{-}’ as in \texttt{TeX}, and the hyphen sign ‘\texttt{-}’. Such a choice allows \texttt{Bib\TeX} to process all the components of a first name more easily when it is abbreviated, even if these tokens are separated by hyphen signs, as in French:

\begin{verbatim}
Jean-Pierre Andrevon
\end{verbatim}

However this design choice causes strange behaviour to happen in some particular cases:

\texttt{\textbf{Jean-Claude Smit-le-B\{\textbackslash e\}n\{\textbackslash e\}dicte}}

whereas this name has only \texttt{First} and \texttt{Last} parts, as suggested by the single space character. Of course, this name should be specified by:

\texttt{\textbf{Jean-Claude Smit-le-B\{\textbackslash e\}n\{\textbackslash e\}dicte}}

More precisely, \texttt{Bib\TeX} retains only the first separator between two tokens, omitting any additional separators from the output:

\begin{verbatim}
Edgar\_Rice \Rightarrow \texttt{Edgar\_Rice}
Edgar\_"Rice \Rightarrow \texttt{Edgar\_Rice}
Edgar\~"Rice \Rightarrow \texttt{Edgar\~Rice}
Karl\~Heinz \Rightarrow \texttt{Karl\~Heinz}
\end{verbatim}

It also omits any separator before the first token.

\textbf{Remark 2} That is why we put the hyphen sign between braces in ‘\texttt{-}\texttt{ky}’ (see above). If the braces are removed, the hyphen disappears.

Now let us assume that there is no \texttt{von} part.

• If the \texttt{Last} part follows the ‘\texttt{-}’ sign, the ‘\texttt{-}’ is omitted:

\begin{verbatim}
Kenneth\~Robeson
\end{verbatim}

• If the \texttt{Last} part follows the ‘\texttt{-}’ character, the word before belongs to this part, too:

\begin{verbatim}
Louis-\texttt{Albert}
\end{verbatim}

2.4 Applying a bibliography style

In general, we can see that strings are not handled homogeneously in \texttt{bst}, the language used to write \texttt{Bib\TeX}'s bibliography styles [19]. Let us consider the three following \texttt{bst} functions:

\begin{verbatim}
S text.length$
S \ I text.prefix$
S \ I_1 \ I_2 substring$
\end{verbatim}

where \texttt{S} is a string and \texttt{I}, \texttt{I}_1, \texttt{I}_2 are integers. The \texttt{bst} language is based on handling a stack, so the arguments are passed to a function by putting their \texttt{values} before the function name.\footnote{ \texttt{MIBib\TeX}'s compatibility mode allows users to see how this stack works [9].} These three functions respectively push (return) the length of \texttt{S}; the first \texttt{I} characters of \texttt{S}; and \texttt{I}_2 characters of \texttt{S}, starting at position \texttt{I}_1 if \texttt{I}_1 > 0, or ending at this position if \texttt{I}_1 < 0, in which case positions are counted backward from the end of the string.

Braces do not count when the \texttt{text.length$} function is called, but they do for the \texttt{substring$} function, as shown by the examples of Figure 2. We also see that the \texttt{substring$} function may push a string where braces are unbalanced. Braces do not count for the \texttt{text.prefix$} function, either, and we can observe strange behaviour:

• if an unbalanced left or right brace is encountered, it is put into the string pushed,

• right braces closing unbalanced left braces are added at the end of the string pushed.

The \texttt{substring$} function counts each character, in the sense that any typed character is relevant, including enclosed braces and characters surrounded by braces. Other functions — \texttt{text.length$} and \texttt{text.prefix$} — consider that if a left brace is immediately followed by a ‘\texttt{\textbackslash}’ character, the complete group between the left enclosing brace and the corresponding right one is viewed as one single character. Such a group is called \texttt{special character} within \texttt{Bib\TeX}'s terminology [16, pp. 768–769]. This distinction between special characters and other groups surrounded by braces explains some results shown in Figure 2. Anyway, let us notice that this distinction is recognised by the \texttt{change-case$} function:

\begin{verbatim}
S S_0 change-case$
\end{verbatim}

— where \texttt{S} is a string — converts \texttt{S} to lowercase (resp. uppercase) if \texttt{S}_0 is the string ‘\texttt{l}’ (resp. ‘\texttt{u}’).
Figure 2: Examples with functions dealing with strings in bst.

Figure 3: Effect of ... "u" case.change$.

and pushes the result. If $S_0$ is "t", $S$ is converted to lowercase except for the first character or the first group if $S$ begins with a left brace. As shown by the examples given in Figure 3, the non-command parts of a special character are processed like ordinary characters—even if some subparts are surrounded by braces—whereas all the other groups surrounded by braces are left unchanged by the change.case$ function. That recalls what is written in [16, p. 768] about protecting some uppercase letters ... provided that the first character after a left brace is not a 'v' character.

Let us now go back to the operations returning subparts of a string. We see that this data structure is handled with difficulty in the bst language, except for simple strings. If we consider values of the AUTHOR and EDITOR fields, the usual way to deal with them is the format.name$ function, used as follows:

$S_1 I S_2 format.name$

where $S_1, S_2$ are strings and $I$ is a natural number. This function formats the $I$th name of $S_1$ according to the pattern given by $S_2$, and pushes the result. The pattern is written as follows:

- if characters are not enclosed by braces, they are unconditionally inserted into the result;
- if this part is absent within the person name, the complete specification surrounded by braces is ignored;
- if this part is not empty:
  - all the characters before the subpattern are inserted before the corresponding part;
  - if the subpattern is immediately followed by a group surrounded by braces, this last group—which may be empty—replaces the separator between two adjacent tokens of the corresponding part;

If an error occurs, such a bst function pushes a dummy value: an empty string in this case.

This terminology is used within the source files of MiBibTeX's compatibility mode.
– the other characters following the subpattern are inserted after the corresponding part.

If ‘~’ characters are put at the end of characters following a subpattern or a separator replacement:

- one ‘~’ causes a space character (\texttt{\textbackslash u}) to be inserted after the name’s part,
- if there are several ‘~’ characters, the first is dropped, while the others are inserted after the name’s part;

at other places, tilde characters are inserted like ordinary characters.\footnote{Let us notice that, in contrast, any occurrence of the space character is processed w.r.t. a ‘standard way’.

Figure 5: Examples of using patterns with the \texttt{format.name$} function of \TeXX.}

Let us assume that there are several tokens for a name’s part. By default — without separator redefinition — a ‘~’ character is inserted:

- always between the next-to-last and last tokens,
- between the first and second tokens: if there are three or more tokens, and the first token is one or two letters long, a special character or period belonging to an abbreviated word is counted as one letter.

All these cases are summarised in the examples given in Figure 5.

A token is abbreviated by retaining only its first letter. So, other characters inserted before this first letter may be dropped. When \TeXX abbreviates a name part, it recognises special characters, as shown in Figure 6. But it does not insert braces for other groups surrounded by braces, as the \texttt{text.prefix$} function would do: compare the examples given in Figures 2 & 6 for the string \{Ch\}arles. Last, let us mention that by default — without separator redefinition — this first letter is followed by a period character and the separator — a space, ‘~’ or ‘\texttt{-}’ character — put after this token. This mark after the first letter is not appended after the abbreviation of the last token. Some of these rules might seem strange if we think of them only in relation to abbreviating first names, but let us recall that some styles use initials of the von and \texttt{Last} parts as labels of bibliographical references. For example, the alpha bibliography style uses the \{v\}{l\{} pattern to generate these labels.

2.5 Criticism

The functions provided by \TeXX work fine in most practical cases, in the sense that most \TeXX users accept the results of the standard bibliography styles. However, the cases not ‘naturally’ included in this framework are difficult to handle. A simple example is given by abbreviations. Let us consider the two following names:

\begin{verbatim}
Edgar Rice Burroughs J. L. White
\end{verbatim}

The \{f.\} subpattern would cause a period to be appended after ‘R’ in ‘E. R. Burroughs’, but incorrectly abbreviates the second example to ‘J. L. White’.\footnote{Anyway, [2, § 14.4] recommends the systematic use of a final period. But such a sign is supposed to replace some letters omitted.} The \{f\} subpattern correctly puts ‘J. L White’ in the second case, but then a period is missing after Edgar Rice Burroughs’ middle name. In addition, it is difficult to specify a particular abbreviation for a middle name only. For example, Nicholas deBelleville Katzenbach’s middle name is correctly abbreviated to ‘dEB.’.
Jean-Michel Hufflen

\begin{center}
\begin{tabular}{l c l}
Charles & $\rightarrow$ & C \\
\{Cha\}rles & $\rightarrow$ & C \\
\{relax Ch\}arles & $\rightarrow$ & \{relax Ch\} \\
\{-\}ky & $\rightarrow$ & k
\end{tabular}
\end{center}

\textbf{Figure 6}: Abbreviating first names in bst.

From a general point of view, some workarounds exist, but may be viewed as hacks. Often they consist in inserting \TeX commands into fields’ values. For example:

- a command deferring the right case \cite[p. 767]{16}:

\begin{verbatim}
Maria \{\MakeUppercase{d e La}\} Cruz
\end{verbatim}

so \texttt{BibTeX} interprets \{\texttt{de...}\} as the beginning of the \texttt{von} part, because the ‘d’ letter is lowercase, even though \texttt{BibTeX} will typeset ‘De’ when the command is processed;

- inserting a dummy accent command \cite[§ 251]{13}:

\begin{verbatim}
\{\relax Ph\}ilippe Djian
\end{verbatim}

in order for this French first name to be abbreviated to ‘Ph.’ because French digraphs — ‘ch’ is a digraph for ‘[j]’ — should not be reduced.

Other ‘tricks’ are more subtle. For example, standard bibliography styles put a space character between the \texttt{von} and \texttt{Last} part. That is suitable for most cases, e.g., ‘Lyon Sprague de Camp’, but not when the particle ends with an elision, e.g., ‘Guy d’Antin’. Testing whether or not the \texttt{von} part ends with an apostrophe character (“’”) is tedious, because the \texttt{Bib} language does not provide a natural way to store part of a name in a variable. In addition, let us not forget that there may be several names inside an \texttt{AUTHOR} or \texttt{EDITOR} field, which complicates the search of the accurate indices. One solution is:

\begin{verbatim}
Guy d’\unskip Antin
\end{verbatim}

— see \cite[Ch. 24]{14} about the \texttt{\unskip} command — but this is not fully satisfactory: it works only if we are sure that the \texttt{\text.prefix} function is not applied to the \texttt{von} part for a prefix length greater than 2. Likewise, the \texttt{\change.case} function should not be used (\textit{cf. supra}). Protecting this \texttt{\unskip} command by braces:

\begin{verbatim}
Guy d\{\unskip\} Antin
\end{verbatim}

would solve these problems but annihilate the effect of this command. The insertion of an \texttt{\aftergroup} command \cite[Ex. 24.7]{14}, deferring some tokens until the end of a group is processed:

\begin{verbatim}
Guy d\{\aftergroup\unskip\} Antin
\end{verbatim}

is useless because the offending space character \texttt{follows} the brace closing the group with \texttt{\aftergroup}. And this space, not surrounded by braces, is needed when \texttt{\BibTeX} separates the \texttt{von} and \texttt{Last} parts. The best solution — \textit{cf.} \cite[Ch. 24]{14} — is:

\begin{verbatim}
Guy d\{\aftergroup\ignorespaces\} Antin
\end{verbatim}

in the sense that it works in most cases, provided that the abbreviated first name is not to be followed by a delimiter, as in ‘Guy d’Antin’.

As mentioned above, the \texttt{\text.prefix} function allows good control of separators between the tokens belonging to a same part. However, some limitations exist. For example, let us consider:

\begin{verbatim}
Ursula Kroeber \{Le\}Guin
\end{verbatim}

By default — that is, using the \{\texttt{ff}\} pattern — an unbreakable space character will be inserted between the ‘actual’ first name (‘Ursula’) and the middle name (‘Kroeber’). If we would like to allow a line-break after the first name because the bibliography will be typeset on a small text width, we can do that by the \{\texttt{\{ff\}\}\} pattern. But such a redefinition is impossible for the two tokens of the last name (‘Le Guin’), surrounded by braces. In such a case, the name may be specified by:

\begin{verbatim}
Le Guin, Ursula Kroeber
\end{verbatim}

but braces are needed for an organisation name:

\begin{verbatim}
AUTHOR = \{\{Hidalgo~Trading~Company\}\}
\end{verbatim}

and redefining the separator between words becomes impossible by means of the \texttt{\text.prefix} function since \texttt{\BibTeX} considers there to be only one token. We can program this operation with the functions \texttt{\substring} and \texttt{*} — concatenation of two strings \cite[Table 13.8]{16} — but it is very tedious.

From our point of view, the use of additional braces belongs to \texttt{\BibTeX}’s ‘philosophy’, but is complicated in the sense that some functions process braced groups opening a \TeX command differently from other groups surrounded by braces. Often this design choice is good — for example, it allows the \texttt{\change.case} function to change the case of accented letters typed by means of \TeX commands — but other operations are difficult to perform. The insertion of \texttt{\BibTeX} commands seems to us to be just a workaround. It works since \texttt{\BibTeX} is used in conjunction with \TeX. That was true when \texttt{\BibTeX} came out,\footnote{Initially, \texttt{\BibTeX} was designed to work with \texttt{\LaTeX} \cite{22}.} but is not always the case nowadays. \texttt{\BibTeX} may be used to generate bibliographies displayed on Web pages written in \texttt{HTML},\footnote{\texttt{\HyperText Markup Language}. \textit{See} \cite{17} for an introduction.}
Figure 7: Internal representation of names using XML in MiBibTeX.

by means of a converter like Bib\TeX2HTML [3]. Another example, closer to \LaTeX, is Hans Hagen’s format Con\TeXt [4]. Using \LaTeX commands within values associated with Bib\TeX fields can cause trouble when these programs run (some examples concerning Con\TeXt and solutions are given in [10]).

3 How names are processed by MiBib\TeX

3.1 Implementation issues

We suggest that the components of a name should be directly accessible by means of different placeholders within the functions of a bibliography style. As explained in [7], parsing bibliographical entries from a .bib file results in an XML tree in MiBib\TeX.\footnote{EXtensible Markup Language. See [21] for an introduction.} The organisation of our elements is a revision and extension of the DTD\footnote{Document Type Definition. A DTD defines a document markup model [21, Ch. 5].} given in [6], influenced by Bib\TeX. Concerning names, fields related to authors and editors are split into subtrees, as shown in the example of Figure 7. In fact, this figure is a ‘pretty-printing’ of such a tree: in reality, the contents of text nodes — e.g., \first, \von, \last — are ‘space-normalised’, that is, stripped of leading and trailing whitespace characters, multiple consecutive occurrences of whitespace characters being replaced by a single space character. Likewise, most of the blank nodes\footnote{Anonymous text nodes whose contents are only whitespace characters; two examples can be found around the and tag in Figure 7. In XML, all characters are preserved [21, p. 38].} pictured in Figure 7 do not exist in the ‘actual’ representation. Besides, this representation uses Latin 1 encoding,\footnote{Future versions of MiBib\TeX should be able to deal with Unicode characters [24].} and some special characters of \LaTeX are processed; for example, the ‘~’ character is replaced by an unbreakable space character (numbered 160 in Latin 1). In addition, some \LaTeX commands are expanded; for example, accent commands applied to suitable letters are replaced by the corresponding accented letters included in Latin 1 [9].

Bibliography styles are written using the nbst\footnote{New Bibliography STyles.} language, close to XSLT,\footnote{EXtensible Stylesheet Language Transformations. See [21, Ch. 6] for a short introduction.} a language of transformations used for XML texts. This nbst language is described in [7].

An example written using nbst is given in Figure 8. If the \von part of a name exists, this template is invoked, the contents of this \von part is written down. This part is followed by a space character, unless its last character is an apostrophe. This template allows the two examples given above — Lyon Sprague de Camp and ‘Guy d’Antin’ — to be displayed nicely. Of course, this is an \textit{ad hoc} solution, but it shows that we get the full expressive power of MiBib\TeX...
3.2 Syntactic issues

MiBibTeX can process any .bib file designed for ‘old’ BibTeX, except that square brackets are syntactic delimiters used for multilingual features [7]. So, most of the ‘tricks’ used within ‘old’ .bib files should work. In addition, MiBibTeX allows more explicit syntax for the components of a person name and the abbreviation of a first name, when needed:

\texttt{first => ..., von => ..., last => ..., junior => ..., abbr => ...}

The order of the keywords is irrelevant and some may be absent, provided that the last name is specified. For example:

\texttt{first => Kim Stanley, last => Robinson}

where the \texttt{von} field is empty, and the abbreviation of the first name is standard, that is, ‘K. ’. Let us notice that in MiBibTeX, the period character ending an abbreviation belongs to it by default. In addition, this new syntax can be used with ‘old’ bibliography styles, written in bst, as we show in Appendix A.

You can mix the ‘old’ and ‘new’ syntaxes, in which case a name is parsed like (i) if no comma occurs before a keyword, like (ii) (resp. (iii)) if the number of commas not followed with a keyword is one (resp. two) and the keywords give additional information. Let us give some examples:

\texttt{Robinson, first => Kim Stanley}

is allowed, because ‘Robinson’ is parsed as the Last part, so ‘Kim Stanley’ is allowed to be the First part. But:

\texttt{Kim Stanley, last => Robinson} \textbf{[WRONG!]} is an incorrect specification, because ‘Stanley’ is supposed to be the Last part, so this part cannot be redefined to ‘Robinson’.

In practice, mixing the old and new syntaxes is useful when we have just to give a specific abbreviation for a first name:

\texttt{Lyon Sprague de Camp, abbr => L. Sprague}

Roughly speaking, this syntax is close to Ada’s for passing values inside a subprogram call [23, § 6.4].

Other keywords can be used for both an organisation name and a key for sorting:

\texttt{EDITOR = \{org => \TUG 2006, sortingkey => TUG2006\}}

As in BibTeX, co-authors are connected with the ‘and’ keyword. After the different co-authors, MiBibTeX allows the addition of collaborators, introduced by the ‘with’ keyword:20

\texttt{Clive Cussler with Paul Kemprecos}

The format for several co-authors and collaborators:

\texttt{... and ... and ... with ... with ...}

In the present article’s bibliography, reference [16] gives an example of how such an entry using several co-authors and collaborators is formatted. As in BibTeX, the ‘others’ keyword can be used when additional names are left unspecified: ‘and others’ (resp. ‘with others’) for additional unspecified co-authors (resp. collaborators).

Multilingual switches with a default21 are allowed for names, which is useful for names originating in languages using other alphabets:

\texttt{AUTHOR = \{[Сергей Сергеевич Прокофьев] * russian [Sergey Sergeyevich Prokofiev]\}}

4 Internationalization of names

MiBibTeX allows names to be displayed according the cultural background of a language. For example, accurate templates of the nbst language [8] allow names of Hungarian people:

\texttt{AUTHOR = \{[Béla Bartók] : magyar [Béla Bartók] : magyar\}}

to be displayed like ‘Bartók Béla’ — that is, the last name at first, followed by the first name — whereas other names (English, French, ...) are displayed as usual, that is, the first name followed by the last name. However, the specification of any person name has to be dispatched into the four components inherited from BibTeX. This is not a problem related to parsing, because the new keywords allow us to specify each component separately. In practice, some cases are solved by extending the First part in order to include:

- a middle name for American people:
  \texttt{Ursula K. Le Guin}

- a patronym (father’s first name) for a Russian name, here ‘Sergey, Sergey’s son’:
  \texttt{Сергей Сергеевич Прокофьев}

Other cases may more difficult to handle. Here are some exotic examples given in [27]:

- an Assyrian name consisted of a personal name, the father’s name, and the grandfather’s name:

\footnotesize

\begin{itemize}
  \item \texttt{... at the topmost level only. See Footnote 3, p. 1002.}
  \item \texttt{This means that something is always produced, even if no information is available in the reference’s language. In other words, this kind of switch never yields nothing. See [7] for more details.}
\end{itemize}

\footnotesize

20 ... at the topmost level only. See Footnote 3, p. 1002.
21 This means that something is always produced, even if no information is available in the reference’s language. In other words, this kind of switch never yields nothing. See [7] for more details.

\normalsize
• in South India, a personal name may be pre-
ceeded by the father’s name, usually written as an initial, and possibly replaced or supple-
mented by the birthplace or mother’s house
name, e.g. ‘Trivandrum R. S. Mani’.22

We plan to go thoroughly into this notion when
the document model of the bibliographies handled
by MiBibTEX is revised by using schemas. For small
examples of using the XML Schema standard for
describing an organisation for names, [25, pp. 91–
107] can be consulted. DocBook, an XML system
for writing structured documents [26], proposes a
more open approach: some optional elements are
defined — e.g., honorific (see Figure 9)— includ-
ing a hold-all element, othername, for information
that does not fit in other categories. This element —
which may be repeated — has a role attribute that
classifies the different kinds of these ‘other’ names.
However, official documents do not make precise the
possible values for this attribute: so names’ taxon-
omy is not really fixed in DocBook, except for classic
cases, for example, ‘mi’ for ‘middle names’, as given
in [26, p. 149]. Figure 9 shows a DocBook example,
including language information. This information is
not only a single identifier, like an option of the babel
package, but allows the specification of variants of a
language. It consists of a two-letter language code
using lowercase letters, optionally followed by a two-
letter country code using uppercase letters [1]: ‘fr’
is for the French language, ‘BE’ for Belgium. Such el-
ements are not limited to a bibliography: as another
example, the specification of an author’s article also
uses them [26, p. 143].

Another formalism belonging to the XML family
widely used for bibliographical metadata is the
Dublin Core,23 but here the contents of the elements
representing people’s names are just strings; such el-
ements are not structured by means of subelements
as in DocBook. For example, an entity primarily re-
ponsible for making the content of a resource may
be a person name and is specified by the dc:creator
element. BibTEX-like syntax is used in practice, as
shown in Figure 10; [27] gives some guidelines.

5 Further development and conclusion
As mentioned above, MiBibTEX includes a com-
patibility mode, so it can apply ‘old’ bibliography styles,
written using bst.24 To put this implementation of
bst into action, of course, we studied the behaviour

---

22 In such a case, periods are sometimes omitted.
23 The Dublin Core metadata standard is a simple yet ef-
fective element set for describing a wide range of networked
resources. See [5] for more details.
24 See [9] for an overview of the functions of this mode.

---

of this language’s functions precisely, preparing very
many tests. That is why we hope that, in particular,
we know precisely how BibTEX deals with names.
This task of reverse engineering showed us that it
was designed in order for current format operations
to be specified concisely. The price paid is compli-
cated specification of general cases.

Due to the huge number of BibTEX database
files written by end-users, a successor of BibTEX has
to ensure compatibility with existing .bib files. So
we have to pay attention whenever we introduce new
syntactic sugar: could existing files be processed as
previously? Another question is: could we add more
and more syntactic sugar to a formalism without
damaging it? Redefining a new one might be bet-
ter...

We think that the work on the new features of
MiBibTEX should be done based on a specification
in XML, and some improvements can be given syn-
tactic sugar, usable within .bib files. For example,
we plan to add a space-after attribute to the von
element:

---

<author>
  <honorific>Sir</honorific>
  <firstname>Arthur</firstname>
  <surname>Conan Doyle</surname>
</author>

<author lang="fr-BE">
  <firstname>J.-H.</firstname>
  <surname>Rosny</surname>
  <lineage>
    <!-- Junior, Senior, etc. [26, p. 308]. -->
    aîné
  </lineage>
</author>

---

Figure 9: Names specified using DocBook.

<dc:creator>Kay, Guy Gavriel</dc:creator>
<dc:creator xml:lang="fr">
  <!-- xml:lang is a predefined attribute in XML
  [21, p. 41], its values are codes described in
  [1]. -->
  Pierre Pelot
</dc:creator>

---

Figure 10: Examples with the Dublin Core.
and specify empty space after a particle by an additional comma in AUTHOR and EDITOR fields:

\{first => Guy, von => d', last => Antin\} or — when the von part is specified last —:
\{first => Gilles, last => Argyre, von => d'\},

On the contrary, future features related to person names whose structure is to be studied will be added only to the XML document model of bibliographies, when it is ready. We think that BibTeX has not been much used for such names, so there should be no need for additional syntactic sugar and MiBibTeX should be able to get such information by parsing XML files.

6 Acknowledgements

Many end-users told me that they had difficulty understanding names’ specification within BibTeX. I was thinking of them when I was writing this article and I hope they will enjoy reading it as much as I enjoyed writing it. Many thanks to Karl Berry and Barbara Beeton, both of whom proofread a first version, suggested some improvements and additional examples.

A MiBibTeX’s names within the compatibility mode

As abovementioned, MiBibTeX allows users to run ‘old’ bibliography styles of BibTeX [19], by means of a compatibility mode, sketched in [9]. This compatibility mode, programmed in Scheme, actually uses a stack and allows users to learn the bst language easily, since they can run a bst program step by step. The data used within the result of parsing .bib files are serialised w.r.t. types used within the bst language’s functions, that is, strings, integers, and literals [16, Table 13.8]. Each function belonging to the bst library is implemented by a Scheme function.

If you push a string and would like to give it to the Scheme function implementing format.name$, only the conventions of BibTeX will be put into action. If the value of an AUTHOR or EDITOR has been processed by MiBibTeX’s parser and has to be passed to the compatibility mode, the transmitted value is a list consisting of a string S, followed by a vector, whose size is the number of the names of S. Let us consider a name belonging to S, for each part of it — First, Junior, Last, von, w.r.t. this order — a pair of indices shows where this part begins and ends. This part is replaced by the false value — #f — if it does not exist. Unless a name only consists of a Last part, the possible syntaxes for the string S are either (ii) or (iii) — cf. § 2.1.

An example is given in Figure 11: given three names specified by means of keywords, we show the three elements of the corresponding structure passed to the compatibility mode. Index pairs are organised into vectors: one vector for the parts of each, these three vectors being elements of a vector of vectors. If such a list, beginning with the ‘*name*’ marker, is popped by another function than b-bst-format.name$ and b-bst-num.names$, the implementations of format.name$ and num.names$, only the string is retained.

This design choice allows us to show BibTeX’s behaviour by default if these two functions are given only a string. But if end-users take advantage of the specification of name parts by means of keywords, we do not need workaround within the strings passed to these two functions. We do not have to put additional braces or dummy \TeX commands. Besides, our indices are coherent with the way used by Scheme for selecting substrings: they are characters beginning with a first inclusive index and ending with a second exclusive index. For example, let s to be the long string given in Figure 11: the expression (substring s 14 19) evaluates to the string "J.-H."

Last, the names of collaborators, introduced after an occurrence of the ‘with’ keyword (cf. § 3.2), are ruled out. Anyway, let us remark that these names would be processed improperly by BibTeX’s bibliography styles.

References


\textit{TUGboat}, Volume 0 (2060), No. 0 — Proceedings of the 2060 Annual Meeting
Names in \texttt{Bib\TeX} and \texttt{MiBib\TeX}

first \Rightarrow J.-H., last \Rightarrow Rosny, junior \Rightarrow jeune and first \Rightarrow Pierre Alexis, last \Rightarrow Ponson du Terrail and first \Rightarrow Eric, von \Rightarrow Van, last \Rightarrow Lustbader

First, Junior, Last, von. The First part of the first

\begin{verbatim}
#((14 . 19) (7 . 12) (0 . 5) #f) ; The order is First, Junior, Last, von. The First part of the first
; element starts at position 14 and ends just before position 19. This
; element does not have a von part.

#((43 . 56) #f (24 . 41) #f)
#((76 . 80) #f (65 . 74) (61 . 64))
\end{verbatim}

Figure 11: How \texttt{MiBib\TeX}'s names are sent to ‘old \texttt{Bib\TeX}'s functions.


[13] Marie-Paule KLUTH: \textit{FAQ \texttt{Bib\TeX} française pour débutants et confirmés}. Vuibert


[19] Oren PATASHNIK: \textit{Designing \texttt{Bib\TeX} Styles}. February 1988. Part of the \texttt{Bib\TeX} distribution.

[20] Oren PATASHNIK: \textit{\texttt{Bib\TeX}ing}. February 1988. Part of the \texttt{Bib\TeX} distribution.


Jean-Michel Hufflen


