OPmac-bib: Citations using `*.bib` files with no external program

Petr Olšák

Introduction

The OPmac package [1] is a set of plain TEX macros which implements the basic LATEX functionality in a simple way. The OPmac-bib module is part of OPmac. It provides the `*.bib` manipulation without any external program (such as BIBTEX [2] or `biber` [3]). This allows you to forget about encoding problems when using the ancient BIBTEX, forget about calling an external program and forget about working with the highly unconventional `*.bst` language. We don't need such complications in plain TEX. We are able to generate bibliography references directly from `*.bib` files only by simple TEX macros. This article introduces the main principles of OPmac-bib.

The `librarian.tex` [4] package is used by OPmac-bib for scanning the `*.bib` databases.

The OPmac and OPmac-bib packages are successfully used by students at CTU in Prague for creating their theses using the CTUstyle template [5].

Common principles

If we need to refer to the existence of another document in our document then such a reference is the subject of formal rules. The rules depend on the scientific discipline, journal requirement, normal conventions, or a mix of these aspects. We first need to declare the terminology about this.

We use *actual document* for the document where the citation is used (linked from) and *cited document* for the mentioned source (linked to). The place in the text of actual document where we need to refer to the cited document is the *citation point.* Common rules say that authors of an actual document only have to have a *citation mark* at the citation point, and they must put the *bibliographic entry* at the end of the actual document in a special section usually called "References" or "Bibliography". The citation mark can be repeated next to the bibliographic entry. The list of all bibliographic entries cited in the actual document is the *references list.* Each bibliographic entry must include *fields* (author, title, year of edition etc.). They must unambiguously specify the cited document. There are *mandatory fields* and *optional fields*, they depend on the *type* (book, article etc.) of the cited document and, of course, on the formal rules. The citation marks may be *numbered* or *non-numbered.*

The non-numbered marks may be *long* (like Knuth 1984) or *short* (like Kn84). Only one type of the citation marks must be used in the whole document.

The TEX user typically doesn't care about citation marks and bibliographic entries. These are generated automatically. The user puts only a *label* at the citation point into the source text of the actual document. The label is invisible in the printed version of the document but it must be the same as the label which determines the bibliographic entry in the `*.bib` database(s) used.

Various rules, conventions and standards govern the following:

- The format of citation marks.
- The rules for grouping citation marks if multiple documents are cited at the same citation point.
- The type of brackets used around citation marks, or around groups of them, possibly with other rules of printing the citation marks.
- Which fields are mandatory and which are optional (dependent on the type of the cited document) when printing the bibliographic entry.
- How to format the bibliographic entry: what separators (punctuation, reserved words or abbreviations) must be used between fields, what font to use for what types of fields, what ordering of fields to use in a given entry.
- The reserved words or abbreviations mentioned above (like "edition", "et al.", "pp.", "available from" etc., can be printed in the language of the actual document or in the language of the cited document. It depends on the kind of reserved word.
- The list of authors is a special field in the entry. There are rules about how to print the names of each individual author (the ordering of first name, last name and other names, the abbreviations of these names etc.), what separators are used between authors, what ordering of authors may be used (alphabetical or by credit).
- There are conventions about ordering the bibliographic entries in the reference list: alphabetically by the first author, by the year of printing, a mix of these, by the occurrence of citation points in the actual document, etc.

It is clear that implementation of these various rules for automatic generation of bibliographic references is a very complicated task. But OPmac-bib solves it and uses only clear and simple TEX macros. No more special external format, no external program is used. Anyone can simply change the predefined TEX macros in order to follow different conventions.

OPmac without bib

The OPmac package without the OPmac-bib module provides the basic manipulation with citations. We give a short summary in this section.

A user writes `\cite[`⟨*label*⟩`]` or `\cite[`⟨*more comma separated labels*⟩`]` at the citation point. The bibliographic entry (at the end of the document) can be created manually with:

`\bib [`⟨*label*⟩`]` ⟨*text of the entry*⟩

The citation marks are auto-generated, numbered by default. When the `\sortcitations` declaration is used then multiple citation marks at a shared citation point are sorted sequentially; and when `\shortcitations` is used then a continuous sequence of the marks is converted to ⟨*from*⟩–⟨*to*⟩ form, for example [3, 4, 5, 11, 12] is converted to [3–5, 11–12].

The OPmac documentation describes how to print different brackets around citation marks (square brackets are the default) or how to use a different format for printing marks.

When `\nonumcitations` is declared then non-numbered citation marks are used. The format of these marks can be declared as a `\bib` parameter when the entries are set manually:

`\bib [`⟨*label*⟩`] = {`⟨*mark*⟩`}` ⟨*text of the entry*⟩

OPmac without OPmac-bib allows the use of BIBTEX as an external program. You can write the following at the place of the reference list:

`\usebibtex{`⟨*bib-base*⟩`}{`⟨*bst-style*⟩`}`

and the reference list is generated automatically. The ⟨*bib-base*⟩ is the name (without extension) of the `*.bib` file used and ⟨*bst-style*⟩ is the name the `*.bst` style file used. The ⟨*labels*⟩ used in the actual document must, of course, correspond with the labels in the `*.bib` file. Four steps must be processed to create the document in such case: TEX, BIBTEX, TEX, TEX.

The format of the generated reference list is given by the `*.bst` style but the obscure language used in these files makes it difficult to modify the formatting. So, OPmac-bib (described in next sections) gives better flexibility for setting the format of reference list.

Bibliographic databases

The common format used for bibliographic entries in the TeX world is derived from BibTeX input and the files have extension `.bib`. The main advantage of this format is that it is a text file, and humans can read it and modify it with an ordinary text editor. It is a well-arranged text file from a human point of view (not the typically obfuscated XML, for example). Although it is a very old format, it can be exported from almost all current bibliographic software. And there are large amounts of data prepared in this format. GUI-oriented programs for manipulating and managing bibliographic databases in this format are available as well.

An entry in the `*.bib` database looks like this:

```
@Book{Knuth:1984:TB,
  author =    "Donald E. Knuth",
  title =     "The {\TeX}book",
  publisher = pub-AW,
  address =   pub-AW:adr,
  pages =     "ix + 483",
  year =      "1984",
  ISBN =      "0-201-13448-9 (paperback),
               0-201-13447-0 (hardcover)",
  ISBN-13 =   "978-0-201-13448-3 (paperback),
               978-0-201-13447-6 (hardcover)",
  LCCN =      "Z253.4.T47 K58 1984",
  bibdate =   "Fri Jul 22 09:08:51 1994",
  bibsource = "http://www.math.../texbook3.bib",
  price =     "US\$15.95 (paperback),
               US\$32.95 (hardcover)",
}
```

The main syntax of this format for one bibliographic entry can be expressed by:

```
@⟨type⟩ { ⟨label⟩,
   ⟨key⟩ = "⟨value⟩",
   ⟨key⟩ = "⟨value⟩",
   ...
}
```

where ⟨*type*⟩ is a type of the cited document (book, article etc.) and ⟨*label*⟩ is the label used in `\cite[`⟨*label*⟩`]`.

The ⟨*key*⟩ gives the type of the field and ⟨*value*⟩ of the field can be enclosed in quotes or braces. If no such delimiter is used, the value is purely numeric, or it is a string identifier or string operations. For example, `pub-AW` and `pub-AW:adr` are string identifiers in the example above. They are declared in the same `*.bib` file as:

```
@String{pub-AW  = "Ad{\-d}i{\-s}on-Wes{\-l}ey"}
@String{pub-AW:adr = "Reading, MA, USA"}
```

Unfortunately for our purposes, the `*.bib` format was designed to be read by BibTeX, and was never intended to be read directly by TeX. So, there are many TeX-unfriendly rules: the two types of delimiters for ⟨*value*⟩s, the case-insensitive identifiers for ⟨*type*⟩s and ⟨*key*⟩s, the very loose rule for setting names of given authors (described in the next section), the special `@string` manipulation (we don't need it because we now will have much more powerful TeX macro language for this), etc.

The OPmac-bib module uses the macro file `librarian.tex` by Paul Isambert for scanning the `*.bib` databases. This macro solves all the TeX-unfriendly aspects mentioned above except for one: the `@string` manipulation. I hope that this does not matter because nowadays many `*.bib` files (generated by bibliographic software or managed by GUI-oriented software) don't use the `@string` feature. And if somebody uses a `*.bib` file where `@string` is present then he/she can apply the `@string` operations manually or with a conversion script.

The Wikipedia page for BibTeX [6] mentions the common ⟨*type*⟩s and ⟨*key*⟩s used in original BibTeX and especially in the original `*.bst` styles interpreted by BibTeX (which are still widely used today). If your citations go beyond this scope, then you have a problem. For example, fields such as `isbn`, `url`, and `doi` are not interpreted by the original `*.bst` styles. You must modify the `*.bst` file — typically a difficult task because of the obscure postfix-based language. Or you can use OPmac-bib to be able to modify the "bib-style" files implemented by straightforward TeX macros.

OPmac-bib from a user's point of view

You write `\input opmac-bib` at the beginning of your document. You don't need `\input opmac` itself because `opmac-bib` loads `opmac.tex` if this isn't done already. The `librarian.tex` macro file is loaded too, so the Librarian package must be installed. It requires ε-TeX activated in the format.

You can use `\cite[`⟨*label*⟩`]` or `\cite[`⟨*labels*⟩`]` as usual. All ⟨*label*⟩s must correspond with ⟨*label*⟩s in the used `*.bib` database. Similarly, you can use `\nocite[`⟨*labels*⟩`]` to insert the corresponding bibliographic entry in the reference list without printing a citation mark at the citation point. If you want to print the whole `*.bib` database then you can write `\nocite[*]`.

The reference list is generated by

```
\usebib/s (⟨style⟩) ⟨bib-base⟩
```

where ⟨*bib-base*⟩ is the name of a `*.bib` file, without extension. You can read multiple `*.bib` files: use a comma-separated list of such file names (without spaces). The ⟨*style*⟩ parameter specifies the used bib-style which determines the printed format of the reference list. Namely, the ⟨*style*⟩ must be a part of a file named `opmac-bib-`⟨*style*⟩`.tex` which will be used for the reference list format. The option `/s` says that the ordering of bibliographic entries is determined by the ⟨*style*⟩. Instead of `/s`, you can use `/c`, which says that the ordering of entries is given by the order of `\cite` or `\nocite` commands in the actual document.

Two processing steps are usually required to create the document: TeX, TeX. In the first, the connection between citation marks and labels is established and in the second, the citation marks are printed correctly.

Two "bib-style" files are provided in the OPmac-bib package: `opmac-bib-simple.tex` and `opmac-bib-iso690.tex`. The second of these prints the reference list in accordance with the ISO 690 standard; the first one is a simple implementation of the style and you can use it as a starting point for your own projects. Moreover, a file `op-example.bib` is included with OPmac-bib, as an example of a `*.bib` file.

You can start experimenting with the following code:

```
\input opmac-bib

Here is \cite[tbn,texbook] and also \cite[lech].
\bigskip
\usebib/s (simple) op-example
\bye
```

For instance, you can try changing `simple` to `iso690` for the style. If you are not using `csplain` but normal `pdftex`, then you will see that the accented letters in the name Olšák are lost. This is due to the fact that `op-example.bib` uses accented letters in the UTF-8 encoding and classic `pdftex` is unable to easily interpret this. You can use `pdfcsplain` instead of `pdftex`. Or you can use `xetex`, but then appropriate Unicode-ready fonts must be loaded, for example with

```
\input ucode
\input lmfonts
```

If multiple entries in the database have the same ⟨*label*⟩ then the rule "first entry wins" is applied (as of the Jan. 2016 version of OPmac-bib). This makes it possible to store the exceptions for a particular document in a (for example) `local.bib` file saved in the same directory as the document and then use a list of database files like this:

```
\usebib/s (iso690) local,global,op-example
```

The list of database files can be arbitrarily long. Once all desired entries (declared by `\cite` and `\nocite`) have been satisfied, then any remaining files in the list are simply skipped.

Features of the `iso690` bib-style

The detailed documentation of the `iso690` style is placed in the file `opmac-bib-iso690.tex` after `\endpinput`. We mention only the basic features in this article. On the other hand, we write here in more detail of the `author` field in order to show the complexity of the problem.

The `iso690` style ultimately accepts the same ⟨*type*⟩s and ⟨*key*⟩s in a `*.bib` file as the standard `*.bst` styles used by BibTeX. So, you can use existing `*.bib` files and the result is essentially the same. Moreover, you can use the following fields for each entry in a `*.bib` file:

```
option    ... space separated parameters, they
              specify more rules for formatting
lang      ... two-letters specification of the
              language of cited document
              (en, cs, sk, de, etc.)
bibmark   ... the non-numbered citation mark
ednote    ... the editorial info
              (illustrations etc.)
citedate  ... the date of citation in the
              YYYY/MM/DD format.
numbering ... alternative format for
              numbering of Journal volumes.
isbn      ... ISBN
issn      ... ISSN
doi       ... DOI
url       ... URL
```

The `author` field. This field type (i.e. a ⟨*key*⟩ used in `*.bib` files) is well-known from BibTeX. But I'll include a short review here and describe new features of `author` field using the `iso690` style.

The `author` field includes one or more authors of the cited document. All names in the author field must separated by ␣`and`␣ separator. Each author can be written in various formats (the "von" part is typically missing):

```
Firstname(s) von Lastname
or
von Lastname, Firstname(s)
or
von Lastname, After, Firstname(s)
```

Only the Lastname part is mandatory. Example:

```
Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
or
da Vinci, painter, Leonardo Piero
```

The separator ␣and␣ between authors is usually changed to a comma for printing, but between the second-to-last and final author the word "and" (or something equivalent, depending on the language of the cited document) is printed.

The name of the first author is printed in reverse order: "LASTNAME, Firstname(s) von, After", while all following authors are printed in normal order: "Firstname(s) von LASTNAME, After". This follows the ISO 690 standard. The Lastname is capitalized using uppercase letters, but if the `\sc` command is defined, then it is used as a font switcher in the form `{\sc Lastname}`. You can declare the "Caps and small caps" font here.

You can specify an option `aumax:`⟨*number*⟩. Here, the ⟨*number*⟩ denotes the maximum number of authors to be printed. Any additional authors are ignored and the "et al." is appended to the list of printed authors. This text is printed only if the `aumax` value is less than the real number of authors. If you have the same number of authors in the `*.bib` file as you need to print but you want to append "et al." then you can use the `auetal` option.

There is also an option `aumin:`⟨*number*⟩ to specify the definitive number of printed authors if the author list cannot be fully printed due to `aumax`. If `aumin` is unused then `aumax` authors are printed in such case.

All authors are printed if the `aumax:`⟨*number*⟩ option isn't given. There is no internal limit. But you can set the global options in your document by setting the `\biboptions` macro. For example:

```
\def\biboptions {aumax:7 aumin:1}
```

means that if there are 8 or more authors, only the first author name is printed.

Some general examples:

```
author = "John Red and Bob Brown and Tom Black",
```

output: RED, John, Bob BROWN, and Tom BLACK.

```
author = "John Red and Bob Brown and Tom Black",
option = "aumax:1",
```

output: RED, John, et al.

```
author = "John Red and Bob Brown and Tom Black",
option = "aumax:2",
```

output: RED, John, Bob BROWN, et al.

```
author = "John Red and Bob Brown and Tom Black",
option = "aumax:3",
```

output: RED, John, Bob BROWN, and Tom BLACK.

```
author = "John Red and Bob Brown and Tom Black",
option = "auetal",
```

output: RED, John, Bob BROWN, and Tom BLACK, et al.

If you need to add text before or after the authors list, you can use the `auprint:{`⟨*value*⟩`}` option. The ⟨*value*⟩ is printed instead of the authors list. The ⟨*value*⟩ can include the `\AU` macro which expands to the authors list. Example:

```
author = "Robert Galbraith",
option = "auprint:{\AU\space [pseudonym
                     of J. K. Rowling]}",
```

output: GALBRAITH, Robert [pseudonym of J. K. Rowling].

Another option is `autrim:`⟨*number*⟩. All Firstnames of all authors are trimmed (i.e. reduced to initials) if the number of authors in the author field is greater than or equal to ⟨*number*⟩. There is an exception: `autrim:0` means that no Firstnames are trimmed. This is the default behavior. Another example: `autrim:1` means that all Firstnames are trimmed.

```
author = "John Red and Bob Brown and Tom Black",
option = "auetal autrim:1",
```

output: RED, J., B. BROWN, T. BLACK, et al.

Language of reserved words. There are two kinds of reserved words and abbreviations which are automatically inserted in the bibliographic entries:

- The word is a part of a field. Two examples are the conjunction "and" between the second-to-last and final author and the "et al." phrase. Such words have to be printed in the language of the cited document.
- The word is prepended to a field value, and it is desired to use the same language for it throughout the entire reference list. For example, the prefix phrase "available from" is used before the `url` field, required by the ISO 690 norm. Such words have to be printed in the language of the actual document.

The language of the actual document is declared by the selection of hyphenation patterns. For example, when we are using `csplain` the default hyphenation is for English, but this is changed to Czech when `\chyph` or `\cslang` is selected at the beginning of the document. You can experiment with our example of `\cite[tbn,texbook]` mentioned in the previous section. Replace `simple` by `iso690`, and see the result: "Available from" is prefixed before the url. But if you use `csplain` and declare `\chyph`, then "Available from" is changed to "Dostupné na" (the Czech equivalent).

The language of the cited document is supposed to be the same as the language of the actual document unless the `lang` field is specified. If the `lang` field is given then it declares the language of the cited document. For example:

```
author = "John Red and Bob Brown and Tom Black",
option = "auetal autrim:1",
lang   = "cs"
```

output: RED, J., B. BROWN, T. BLACK a kol.
We see that "a kol." (the Czech phrase) is used instead of "et al.".

Each entry is printed with hyphenation patterns locally set to the language of the cited document, as declared in the `lang` field.

OPmac knows only the `en`, `cs` and `sk` languages by default. If you are using the `etex.src` macros (i.e. you are using `etex`, `pdftex`, `xetex` or `luatex`) then you can declare new languages with `\isolangset{`⟨*long*⟩`}{`⟨*short*⟩`}`; for example, `\isolangset{espanol}{es}`. However, if you are using `csplain` or `pdfcsplain` then you must reinstall the format with new hyphenation patterns; see the file `hyphen.lan`. Each language has an explicitly declared number in this file. Use this number for declaration of the language identifier by `\sdef{lan:`⟨*number*⟩`}{`⟨*short*⟩`}`, for example `\sdef{lan:26}{es}` and `\sdef{lan:126}{es}`.

Sorting of entries. When `\usebib/c` is used then the order of the entries in the reference list is given by the order of `\cite` and `\nocite` in the document. If (more usually) `\usebib/s` is specified, then the order of the entries follows the ISO 690 standard: sort by the first author (last name, first names) alphabetically; if entries remain the same from this, then use the year of the edition (from older to newer). If both of these sort keys (first author and the year) are the same then the norm does not specify the ordering.

But ... what does *alphabetical ordering* mean? We can have many cited documents with authors from different parts of the world. We may need to sort names from ancient Babylon, but there is no standard for this. Sorting has standards for particular languages only. Thus, the idea of using the standard for sorting in the language of the actual document is bad because no language includes all characters used in possible author names.

This is the reason why I did not attempt to solve this problem; instead, I simply used the character-code sorting provided by `librarian.tex`. Of course, this may give bad results, but we can deal with exceptions when needed. For example, an entry with the author "Světla Čmejrková" is placed at the end of the reference list and this is wrong; "Č" must be sorted between "C" and "D" in Czech. But we can declare the `key` field. When this field is used, its value is used for sorting instead of the names in the `author` field. So,

```
author    = "Světla Čmejrková",
key       = "Czzmejrkova Svetla",
```

does the desired correction of sorting.

If someone needs the rule of automatically putting self-citations before all others, `key = "@"` can be added to all entries with his/her name. This works because the code of the `@` character is less than the codes of all alphabetic characters.

Writing bib-styles

Documentation for bib-style programmers can be found in the `opmac-bib.tex` file after its `\endinput`. The existing styles `opmac-bib-simple.tex` and `opmac-bib-iso690.tex` may be helpful for examples and inspiration.

The style file is read inside a TEX group when the `\usebib` macro is processed. The `*.bib` files are read in the same group and the reference list is printed afterwards. Then the group is closed. This means that all settings and definitions done in a style file are local to this group. The `\bibtexhook` macro is expanded immediately after the style file is read, but before processing any `*.bib` files. Users can set this macro in order to redefine some bib-style features. The macro is empty by default.

The bib-style file must define a `\print:`⟨*type*⟩ macro for each ⟨*type*⟩ of bibliographic entry. These macros are expanded to print an entry with the given ⟨*type*⟩. For example, `\print:book` is expanded when a processed entry has the type `@book`. You must use lowercase letters in the control sequence.

The programmer can use the helper macros `\bprinta` and `\bprintb`. Both have the same syntax:

```
\bprinta [⟨key⟩] {⟨if exists⟩} {⟨if doesn't exist⟩}
```

If the field given by the ⟨*key*⟩ exists then the ⟨*if exists*⟩ part is processed, else the ⟨*if doesn't exist*⟩ part. The first parameter of `\bprinta` can include the `*` character: this symbol is replaced by the value of the given field. The `*` character cannot be "hidden" in next level of braces `{...}`. You can use `\bprintb` instead of `\bprinta` with the same effect but use `##1` instead of `*` and this parameter can be hidden in braces — though nested macro calls need more hashes.

Example for printing an `@Book` entry:

```
\sdef{print:book}{%
 \bprinta [!author] {*\.\ }        {\bibwarning}%
 \bprintb [title]   {{\em##1}\.\ } {\bibwarning}%
 \bprinta [edition] {*~\mtext{bib.edition}.\ }{}%
 \bprinta [address] {*: }          {\bibwarning}%
 \bprinta [publisher] {*, }        {\bibwarning}%
 \bprinta [year]    {*.\ }         {\bibwarning}%
 \bprinta [isbn]    {ISBN~*.\ }    {\bibwarning}%
 \bprintb [url]     {\preurl\url{##1}. }      {}%
}
```

The list of authors is printed first. The exclamation mark before the key `author` means that the value of this field is printed by a special rule (using `\authorname` macro, see below). The list of authors is printed in place of `*`, followed by the "maybe dot" represented by the `\.` macro. This macro prints a dot only if the preceding character is not a dot, exclamation or question mark. Why do we need this? The name can be ended with a dot (due to abbreviating the first name, for example) and we do not want to print a second dot in such cases. A normal interword space (`\ `) follows after the "maybe dot". If the field `author` is missing then `\bibwarning` prints a warning about a missing mandatory field for type `@Book`.

Then the title is printed. It is printed in italics using `\em` macro. The "maybe dot" is used again, because the title might end with a question mark, for example. A space follows. `title` is another mandatory field, so again `\bibwarning` prints the warning if the field is missing.

The optional field `edition` follows. The text "edition" is appended but this text depends on the language. So, the `bib.edition` label is declared in the style file by:

```
% Multilinguals:    English   Czech    Slovak
\mtdef{bib.edition} {edition} {vydání} {vydanie}
```

As seen here, only the Czech, Slovak and English languages are provided by default. If you need to support another language then you need to add the phrases of the language like this:

```
\sdef{mt:bib.edition:es}{liberación}
```

Let us return to the example above. The mandatory field `address` follows. (Typically the city of the publisher is here, not a full address.) Then a colon and space are printed, then the mandatory `publisher` field, and a comma and space. The mandatory field `year` follows, then a period and space. The mandatory field `isbn` is prefixed by the text "ISBN" and a non-breaking space, and followed by a period and normal space. Finally, the optional field `url` is printed; the `\preurl` macro can print something before such a url.

The `\authorname` macro must be defined in the style file. This macro processes the authors' names. It is called for each author name (repeatedly, if there is more than one author in one `author` field). The following data are available in the macro body: `\Namecount` is the number of this author name in the `author` field, `0\namecount` expands to the number of all authors in the field, the `\Lastname` macro expands to the last name of the processed author and similarly with `\Firstname`, `\Von` and `\Junior`. Here is the definition from the "simple" style:

```
\def\authorname{%
  \ifnum\NameCount>1
     \ifnum0\namecount=\NameCount
        \mtext{bib.and}\else , \fi
  \else
     \ifx\dobibmark\undefined
        \edef\dobibmark{\Lastname}\fi
  \fi
  \bprintc\Firstname{* }\bprintc\Von{* }%
  \Lastname\bprintc\Junior{, *}%
}
```

We can see that the last name of the first author is saved to the `\dobibmark` macro (for further processing of non-numeric citation mark). Otherwise the comma+space is printed before the author, except for the last author, when `\mtext{bib.and}` is used instead. This expands to the "and" conjunction, according to the language of the cited document.

Next, the first name(s) is printed, then the "Von" part (if it exists), then the last name, and then the "Junior" part (if this exists, it means the "After" part of the name). The optional parts of the name are printed using the `\bprintc` macro. This prints nothing if its first parameter is empty, otherwise it prints its second parameter and replaces `*` by its first parameter.

The style must generate non-numeric citation marks too, but this is another story not shown here. Interested readers can see the existing implementations in the files `opmac-bib-simple.tex` and `opmac-bib-iso690.tex`.

More features

The bib-style files specify the format of each entry in a reference list, but only in the context of printing the fields in an (imaginary) infinite line. How to break this into paragraph lines, what indentation to use, where to place the citation mark (if any) etc. is another story. This is solved by the `\printbib` macro defined in OPmac itself. This macro is expanded before each entry and it has the following default definition:

```
\def\printbib{\hangindent=\iindent
 \ifx\citelinkA\empty
   \noindent\hskip\iindent\llap{[\the\bibnum] }%
 \else \noindent \fi
}
```

This means: The indentation is set by `\iindent` value and when `\nonumcitations` isn't used, the numbered citation mark `\the\bibnum` is printed by `\llap`. Else no citation mark is printed and only `\noindent` is processed.

This is a typical OPmac approach: it doesn't provide/document a plethora of options and doesn't define complicated macros. It gives a simple default macro with the idea: "hey, you can just redefine it if you want something else". This is an important difference between OPmac and most LaTeX packages or the ConTeXt format. You needn't remember (or repeatedly read in documentation) the syntax and values of dozens of options. Only one thing you need: to be able to create macros in TeX.

There are many variants of the `\printbib` macro on the OPmac tricks web page [7]. For example, OPmac trick 0040 [8] gives a recipe for indenting all entries by the width of the citation mark with the maximal number. OPmac trick 0041 [9] solves the analogous problem for non-numbered citation marks.

The default `\printbib` macro assumes that non-numbered citation marks are in "long" format with the full last name of the first author, and the reference list is ordered by this last name. This is a reason why these long citation marks are not printed in the reference list again. If you do need to (re)print these citation marks then you can follow OPmac trick 0096 [10].

The style files generate non-numbered citation marks in the format [Last name, Year] by default. If the `bibmark` field is present then the value from this field is used instead of the generated value. OPmac trick 0097 [11] shows how to convert these long non-numbered marks such as (Knuth, 1984) to the abbreviations like [Kn84] without any change in the style file or `*.bib` file. These abbreviations are sometimes part of citation rules.

There are special rules for grouping long non-numbered citation marks. For example, it isn't a good idea to repeat the full citation mark for multiple entries with the same author (Olšák, 1995, Olšák, 1997, Olšák, 2013). It is much better to print (Olšák, 1995, 1997, 2013). How to do this automatically using macros is described in OPmac trick 0035 [12]. The comma after the name can be removed with OPmac trick 0043 [13].

When you are using non-numbered citation marks, it might happen that two different entries have the same citation mark. OPmac trick 0098 [14] gives a recipe for diagnosing this problem. If such a situation occurs, the author can set a different citation mark using the `bibmark` field in `*.bib` file, for example.

When you are preparing a proceedings or a long monograph then you probably need to treat the citation marks and reference lists independently and locally in each part of the work (each article, each chapter, etc.). The citation marks and reference list in one part must be independent of the reference list in another part. How to handle this in a single document is solved in OPmac trick 0042 [15].

References

1. `http://petr.olsak.net/opmac-e.html`
2. `http://www.bibtex.org/`
3. `http://biblatex-biber.sourceforge.net/`
4. `https://www.ctan.org/pkg/librarian`
5. `http://petr.olsak.net/ctustyle-e.html`
6. `http://en.wikipedia.org/wiki/BibTeX`
7. `http://petr.olsak.net/opmac-tricks-e.html` = ⟨*opmac-tricks*⟩
8. ⟨*opmac-tricks*⟩`#bibnumindent`
9. ⟨*opmac-tricks*⟩`#bibmarkindent`
10. ⟨*opmac-tricks*⟩`#abib`
11. ⟨*opmac-tricks*⟩`#bibmark`
12. ⟨*opmac-tricks*⟩`#ccite`
13. ⟨*opmac-tricks*⟩`#modcite`
14. ⟨*opmac-tricks*⟩`#bibmarkcheck`
15. ⟨*opmac-tricks*⟩`#morebibs`

◇ Petr Olšák
Czech Technical University
in Prague
`http://petr.olsak.net`