OPmac: Macros for plain TEX

Petr Olšák

The OPmac package provides simple additional macros on top of plain TEX. It enables users to take advantage of basic LATEX functionality: font size selection, automatic creation of tables of contents and indices, working with bibliography databases, tables, references optionally including hyperlinks, margin settings, etc. In this paper, the significant properties of OPmac are shown. The complete source of the macros and user and technical documentation is available through CTAN and the usual TEX distributions, and its home on the web is `http://petr.olsak.net/opmac-e.html`.

Introduction

I have decided to publish my macros together with the new version of CSplain. I have been using these macros for a long time for many purposes in my own work. Now, I have made them cleaner, added user and technical documentation, and released them.

The main reason is to give a set of macros which solves common authorial tasks for plain TEX users. A side benefit is that the macros demonstrate that it is possible to do TEX code simply and effectively. Most LATEX macro packages don't have this feature. All macros are in the single (documented) file `opmac.tex` with only 1500 lines. On the other hand the LATEX code which solves comparable tasks is placed in a kernel and dozens of LATEX packages with many tens of thousands of lines in total.

Here are the main principles which I followed when creating this macro package:

- Simplicity is power.
- Macros are not universal, but are readable and understandable.
- Users can easily redefine these macros as they wish.

Each part of the macro code is written to maximize readability for humans who want to read it, understand it and change it.

The OPmac package offers a markup language for authors of texts (like LATEX), i.e. a fixed set of tags to define the structure of a document. This markup is different from LATEX markup. It offers the possibility of writing the source text of a document somewhat more clearly and attractively. The OPmac package, however, does not deal with the many possible typographic designs of a document. A simple, sober document is created if no additional macros are used. We assume that authors will be able to modify the look of the document to suit their requirements. You can see a complex example of using OPmac with added macros for typesetting design at `http://petr.olsak.net/ctustyle.html`: CTUstyle is the recommended design style for bachelor, master or doctoral theses at Czech Technical University in Prague.

The following text is a short digest of the documentation. It illustrates the capability of the OPmac package.

Using OPmac

OPmac is not compiled as a format. To use it in plain TEX, you can simply `\input opmac` at the beginning of your document. Here's a trivial document as a first example:

```
\input opmac
\typosize[11/13] % set basic font size
                 % and baselineskip
\margins/1 a4 (1,1,1,1)in % set 1in margins
                          % for A4 paper
Here is the text.
\bye
```

Font sizes

The commands for font size setting described here are all local. In other words, if you use them in a TEX group, the font sizes are selected locally within the group, not globally.

The command

`\typosize[`⟨*fontsize*⟩`/`⟨*baselineskip*⟩`]`

sets the font size of text and math fonts and the baselineskip. If one of the two parameters is empty, the corresponding feature stays unchanged. The metric unit is `pt` by default; this unit isn't written in the parameter values. You can change the unit by the command `\ptunit=`⟨*something-else*⟩, for instance `\ptunit=1mm`. Examples:

```
\typosize[10/12]   % default in plain TeX
\typosize[11/12.5] % font size 11pt,
                   %   baselineskip 12.5pt
\typosize[8/]      % font size 8pt,
                   %   baselineskip left unchanged
```

The command

`\typoscale[`⟨*font-factor*⟩`/`⟨*baselineskip-factor*⟩`]`

sets the text and math fonts size and baselineskip to a multiple of the current font size and baselineskip. The factor is written like TEX's `scaled` values, meaning that 1000 leaves the value as-is. An empty parameter is equivalent to 1000. Examples:

```
\typoscale[800/800]     % fonts and baselineskip
                        % re-sized to 80%
\typoscale[\magstep2/] % fonts bigger by 1.44x
```

The sizes declared by these macros (for example in titles) are relative to the basic size selected for the font (this may be an arbitrary size, not only 10pt).

The size of the current font can be changed with the command \thefontsize[⟨*font-size*⟩] or rescaled with \thefontscale[⟨*factor*⟩]. These macros do not change the math font sizes or the baselineskip.

The commands \resizefont, \regfont and \resizeall are available for generally resizing fonts. They're described in the companion article on 𝒞𝒮plain, but can be used with OPmac alone; 𝒞𝒮plain need not be the format. The best design size of the font for desired size is used. For example, with Computer Modern, \typosize[18/] selects the font cmr17 at 18pt.

Parts of the document

A document can be titled and divided into chapters, sections and subsections, The parameters have to be ended with an empty line (no braces are used):

```
\tit  Document title    ⟨empty line⟩
\chap Chapter title     ⟨empty line⟩
\sec  Section title     ⟨empty line⟩
\secc Subsection title  ⟨empty line⟩
```

Chapters are numbered with one number, sections by two numbers (⟨*chapter*⟩.⟨*section*⟩) and subsections by three numbers (similarly). If there are no chapters then sections have only one number and subsections two.

The design of the chapter etc. titles are implemented in the macros \printchap, \printsec and \printsecc. Users can simply change these macros to get their desired output.

The first paragraph after the title of chapter, section and subsection is not indented by default; giving \let\firstnoindent=\relax makes all paragraphs indented.

If a title is long enough, it breaks across multiple lines. It is better to explicitly give the breakpoints because TEX cannot interpret the meaning of the title. Users can insert the \nl (meaning newline) macro to specify the breakpoints.

Other numbered objects

Apart from chapters, sections and subsections, there are other automatically-numbered objects: equations and captions for tables and figures.

If \eqmark is given as the last element in a math display then this equation is numbered. The format is one number in brackets. This number is reset in each section.

In displays using \eqalignno, \eqmark can be given in the last column before \cr. For example:

```
\eqalignno{
    a^2+b^2 &= c^2 \cr
          c &= \sqrt{a^2+b^2} & \eqmark \cr}
```

The next numbered object is captions; these are tagged with \caption/t for tables and \caption/f for figures. Example:

```
\hfil\table{rl}{
    age   & value \crl\noalign{\smallskip}
    0--1  & unmeasurable \cr
    1--6  & observable \cr
    6--12 & significant \cr
   12--20 & extreme \cr
   20--40 & normal \cr
   40--60 & various \cr
   60--$\infty$ & moderate}
\par\nobreak\medskip
\caption/t The relationship of
   computer-dependency to age.
```

This example produces:

age	value
0–1	unmeasurable
1–6	observable
6–12	significant
12–20	extreme
20–40	normal
40–60	various
60–∞	moderate

Table 2.3 The relationship of computer-dependency to age.

The word "Table" followed by a number is added by the macro \caption/t. The macro \caption/f creates the word figure. The caption text is centered. If it occupies multiple lines then the last line is centered.

The added word (table, figure) depends on the value of the \language register. OPmac implements the mapping from \language numbers to languages and the mapping from languages to the generated words.

To make the table or figure a floating object, you can use the plain TEX macros \midinsert, \topinsert and \endinsert.

A \label[⟨*label*⟩] command preceding the automatically-numbered object allows symbolic referencing to the object. The reference commands are `\ref[`⟨*label*⟩`]` (for the value of the number) and `\pgref[`⟨*label*⟩`]` (for the page number). Example:

```
\label[beatle] \sec About The Beatles
...
\label[comp-dependence]
\hfil\table{rl}{...} % the table
\caption/t The relationship of
    computer-dependency to age.
...
\label[pythagoras]
$$ a^2 + b^2 = c^2 \eqmark $$

Now we can point to the section~\ref[beatle] on
the page~\pgref[beatle] or write about the
equation~\ref[pythagoras]. Finally there
is an interesting Table~\ref[comp-dependence].
```

Lists

A list of items is surrounded by `\begitems` and `\enditems` commands. The asterisk (`*`) is active within this environment and it starts one item. The item style can be chosen by `\style` parameter written after `\begitems`:

```
\style o  % small bullet
\style O  % big bullet (default)
\style -  % hyphen char
\style n  % numbered 1., 2., 3., ...
\style N  % numbered 1), 2), 3), ...
\style i  % roman numerals (i), (ii), (iii), ...
\style I  % Roman numerals I, II, III, ...
\style a  % lettered a), b), c), ...
\style A  % Lettered A), B), C), ...
\style x  % small rectangle
\style X  % big rectangle
```

Another style can be defined with the command `\sdef{item:`⟨*style*⟩`}{`⟨*text*⟩`}`. The default style can be redefined with `\def\normalitem{`⟨*text*⟩`}`. List environments can be nested. Each new level of item is indented by next multiple of `\iindent` which is set to `\parindent` by default.

Table of contents

The `\maketoc` command prints a table of contents of all `\chap`, `\sec` and `\secc` titles used in the document. The text is read from an external file, so you have to run TeX more than once (typically three times if the table of contents is at the beginning of the document).

A section name for the table of contents itself is not printed. The usage of `\chap` or `\sec` isn't recommended here because the table of contents is typically not referenced to itself. You can print the unnumbered (and unreference-able) title with the code:

```
\def\thesecnum{}
\printsec{\unskip Table of Contents}
\maketoc
```

The titles of chapters etc. are written to an external file and then read from this file in a subsequent run of TeX. This technique can create problems when a somewhat complicated macro is used in a title. OPmac solves this problem in a different way than LaTeX: users declare the problematic macro as "robust" via an `\addprotect\macro` declaration. The `\macro` itself cannot be redefined. The common macros used in OPmac which are likely to occur in titles are already declared in this way.

Making an index

An index can be included in a document with the `\makeindex` macro. No external program is needed: the alphabetical sorting is done inside TeX at the macro level.

The `\ii` command (insert to index) declares the following word, terminated by a space, as the index item. This declaration is represented as an invisible atom on the page connected to the next visible word. The page number of the page where this atom occurs is listed in the index entry. So you can type:

```
The \ii resistor resistor is a passive
electrical component ...
```

You can avoid doubling the word by using `\iid` instead `\ii`:

```
The \iid resistor is a passive
electrical component ...
Now we'll deal with the \iid resistor .
```

As shown, a period or comma has to be separated from the word by a space when `\iid` is used. This space (before the punctuation) is removed by the macro in the current text.

If you need to have an actual space in an index entry, use "`~`". For example:

```
\ii linear~dependency Linear dependency of ...
```

Multiple-word entries are often organized in the index in the format (for example):

linear dependency 11, 40–50
— independence 12, 42–53
— space 57, 76
— subspace 58

To do this you have to declare the parts of the words with the `/` separator. Example:

```
{\bf Definition.}
\ii linear/space,vector/space
{\em Linear space} (or {\em vector space}) is ...
```

The number of parts in one index entry is unlimited. You can save typing via commas in the `\ii` parameter: the previous example is equivalent to `\ii linear/space \ii vector/space`.

Another need is to propagate to the index the "reversed" terms; e.g. given `linear/space`, you also want to index `space/linear`. You can do this conveniently with the shorthand `,@` at the end of the `\ii` parameter. For example:

```
\ii linear/space,vector/space,@
```

is equivalent to:

```
\ii linear/space,vector/space
\ii space/linear,space/vector
```

The `\makeindex` macro creates the list of alphabetically sorted index entries with no section title and without using multiple columns. OPmac provides another macro for multi-column typesetting:

```
\begmulti ⟨number of columns⟩
⟨text⟩
\endmulti
```

The columns will be balanced. The index title can be printed with `\sec`. So an index in an OPmac document might look like this:

```
\sec Index\par
\begmulti 3 \makeindex \endmulti
```

Only "pure words" can be propagated to the index with the `\ii` command; there cannot be any macros, TEX primitives, math selectors etc. OPmac provides another way for create such complex index entries: use a "plain text equivalent" as the `\ii` parameter, and map this equivalent to the desired TEX word which is printed in the index with the `\iis` command. Here's an example:

```
The \ii chiquadrat $\chi$-quadrat method is
...
If the \ii relax |\relax| command is used
then \TeX\ is relaxing.
...
\iis chiquadrat {$\chi$-quadrat}
\iis relax {{\tt \char`\\relax}}
...
```

The `\iis` ⟨*equivalent*⟩ `{`⟨*text*⟩`}` creates one entry in the "dictionary of the exceptions". The sorting is done by ⟨*equivalent*⟩, while ⟨*text*⟩ is printed in the index entry list.

Czech/Slovak standard alphabetical sorting is used if the `\language` register is set to the Czech or Slovak hyphenation patterns when `\makeindex` is in progress. (The main difference from English sorting is that "ch" is treated as one character between "h" and "i".)

Colors

The color selection macros work only if a pdfTEX-like engine is used. OPmac provides a small number of color selectors: `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow`, `\White`, `\Grey`, `\LightGrey` and `\Black`. Users can define more such selectors by setting the CMYK components. For example:

```
\def\Orange{\setcmykcolor{0 0.5 1 0}}
```

The selectors change the color of the text and of lines with a thickness larger than 1bp. If `\linecolor` immediately precedes the color selector then the lines with a thickness less than or equal to 1bp are colored. This is a second independent color setting.

The color selectors work globally starting on the current page. If the colored text continues to the next page, the color is correctly set on the following page(s) after a second run of TEX, because this event is implemented via external file. Users can also write `\localcolor` inside a group. This command saves the current color and restores it after the group is completed. By default, it is assumed that the group corresponds to the boundary of a box which cannot break across pages. If this is not true, `\longlocalcolor` can be used instead of `\localcolor`. A basic example:

```
\Red the text is red
\hbox{\localcolor \Blue here is blue
  {\localcolor \Green and green}
  restored blue \Brown and brown}
now the text is red again.
```

A more usable example follows. Let's define a macro which creates colored text on a colored background, to be used like this:

```
\coloron⟨background⟩⟨foreground⟩{⟨text⟩}
```

Such a macro can be defined and used like this:

```
\def\coloron#1#2#3{%
  \setbox0=\hbox{#3}\leavevmode
  {\localcolor
   \rlap{#1\strut\vrule width\wd0}%
   #2\box0}}

\coloron\Yellow\Brown{Brown text
  on a yellow background}
```

PDF hyperlinks and outlines

If the command

```
\hyperlinks{⟨color-int⟩}{⟨color-ext⟩}
```

is used at the beginning of the file, then the following are hyperlinked when PDF output is used:

- numbers generated by `\ref` or `\pgref`,
- numbers of chapters, sections and subsections in the table of contents,
- numbers or marks generated by `\cite` command (bibliography references),
- texts printed by `\url` command.

The last object is an external link and it is colored by ⟨*color-ext*⟩. Others links are internal and they are colored by ⟨*color-int*⟩. Example:

```
\hyperlinks \Blue \Green % internal links blue,
                         % URLs green.
```

You can use another method of marking active links: frames which are visible in the PDF viewer but invisible when the document is printed. To do this, define the macros `\pgborder`, `\tocborder`, `\citeborder`, `\refborder` and `\urlborder` to be the RGB color value (a triple) to use. Examples:

```
\def\tocborder{1 0 0}  % links in toc:
                       %   red frame
\def\pgborder{0 1 0}   % links to pages:
                       %   green frame
\def\citeborder{0 0 1} % links to references:
                       %   blue frame
```

By default these macros are not defined, so no frames are created.

There are "low level" commands to create the links. You can specify the destination of an internal link with `\dest[`⟨*type*⟩`:`⟨*label*⟩`]{`⟨*height*⟩`}`. Active text linked to the `\dest` can be created with `\link[`⟨*type*⟩`:`⟨*label*⟩`]{`⟨*color*⟩`}{`⟨*text*⟩`}`. The ⟨*type*⟩ parameter is one of `toc`, `pg`, `cite`, `ref` or one user-defined for your purposes. The ⟨*height*⟩ parameter gives the vertical distance between the actual destination point and the current baseline.

The `\url` macro prints its parameter in the `\tt` font and inserts potential breakpoints (after slash or dot, for example). If the `\hyperlinks` declaration is used then the parameter is treated as an external url link. An example: `\url{http://www.olsak.net}`.

The PDF format also provides for "outlines" which are notes placed in a special frame of a PDF viewer. These notes are usually managed as a structured and hyperlinked table of contents of the document. The command `\outlines{`⟨*level*⟩`}` creates such an outline from the table of contents data in the document. The ⟨*level*⟩ parameter gives the default level of opened outlines. Deeper levels can be opened by (typically) clicking on the triangle symbol after that.

The command `\insertoutline{`⟨*text*⟩`}` inserts next entry into "outlines" at the main level 0. This entry can be placed before table of contents (created by `\outlines`) or after it.

Verbatim

Display verbatim text in OPmac is surrounded by the `\begtt` and `\endtt` pair. Inline verbatim is tagged (before and after) by a character declared with `\activettchar`⟨*char*⟩. For example `\activettchar|` makes the `|` character do inline verbatim markup, as in the *TUGboat* style.

If the numerical register `\ttline` is set to a non-negative value then display verbatim numbers the lines. The first line is numbered `\ttline`+1 and when the verbatim display ends, the `\ttline` value is equal to the number of last line printed. The next `\begtt...\endtt` environment will continue the line numbering. OPmac sets `\ttline=-1` by default.

The indentation of lines in display verbatim is controlled by the `\ttindent` register. This register is set to `\parindent` at the time `opmac.tex` is read. Users should change its value as desired, e.g. if `\parindent` is changed after `opmac.tex` is read.

The `\begtt` starts a group in which the catcodes are changed. Then the `\tthook` macro is run. This macro is empty by default; users can control fine behavior with it. For example, more catcodes can be reset here. To define an active character in `\tthook`, you can use `\adef` as in this example:

```
\def\tthook{\adef!{?}\adef?{!}}
\begtt
Each occurrence of the exclamation mark
will be changed to the question mark
and vice versa. Really? You can try it!
\endtt
```

The `\adef` command sets its parameter as active *after* the body of `\tthook` is read. So you need not worry about active definitions beforehand.

Here are some tips for global `\tthook` definitions:

```
     % setting font size for verbatim:
\def\tthook{\typosize[9/11]}
     % each listing is numbered from 1:
\def\tthook{\ttline=0}
     % visible spaces:
\def\tthook{\adef{ }{\char`\ }}
```

You can print a verbatim listing of an external file with the `\verbinput` command. Examples:

```
    % whole file program.c is printed:
\verbinput (-) program.c
    % only lines 12-42:
\verbinput (12-42) program.c
    % from beginning to line 60:
\verbinput (-60) program.c
    % from line 61 to the end:
\verbinput (61-) program.c
    % starting at line 70, only 10 lines printed:
```

```
\verbinput (70+10) program.c
    % from last line read, print 10 more lines:
\verbinput (+10) program.c
    % from last line read, skip 5, print 7:
\verbinput (-5+7) program.c
    % from last line read to the end:
\verbinput (+) program.c
```

The `\ttline` influences the line numbering in the same way as the `\begtt...\endtt` environment. If `\ttline=-1` then real line numbers are printed; this is the default. If `\ttline` < -1 then no line numbers are printed.

The `\verbinput` output can be controlled by `\tthook` and `\ttindent`, also just as with `\begtt...\endtt`.

Tables

The macro `\table{`⟨*declaration*⟩`}{`⟨*data*⟩`}` provides ⟨*declaration*⟩ similar to LaTeX: you can use the letters `l`, `r`, and `c`, with each letter declaring one column aligned to left, right, center respectively. These letters can be combined with the "`|`" character to create a vertical line.

The command `\cr` ends a row as usual. OPmac defines the following similar commands:

- `\crl` ends the row, with a horizontal line after.
- `\crli` is like `\crl`, but the horizontal line doesn't intersect any vertical double lines.
- `\crlli` is like `\crli`, but horizontal line is doubled.

Basic example:

```
\table{||lc|r||}{                    \crl
   Month    & commodity & price    \crli
                                   \tskip.5ex
   January  & notebook   & \$ 700  \cr
   February & skateboard & \$ 100  \cr
   July     & yacht      & k\$ 170 \crl}
```

which generates the following result:

Month	commodity	price
January	notebook	$ 700
February	skateboard	$ 100
July	yacht	k$ 170

The `\tskip`⟨*dimen*⟩ command adds ⟨*dimen*⟩ vertical space after the current row, more or less like `\noalign{\vskip`⟨*dimen*⟩`}` but without creating interruptions in vertical lines.

The configuration macros for `\table` are shown in the following, with their default values:

```
    % left material in each column:
\def\tabiteml{\enspace}
    % right material in each column:
\def\tabitemr{\enspace}
    % strut inserted in each line:
\def\tabstrut{\strut}
    % space between double vertical line:
\def\vvkern{1pt}
    % space between double horizontal line:
\def\hhkern{1pt}
```

If you do

```
\def\tabiteml{$\enspace}\def\tabitemr{\enspace$}
```

then `\table` acts like LaTeX's array environment.

The command `\frame{`⟨*text*⟩`}` makes a frame around ⟨*text*⟩. You can put the whole `\table` into `\frame` to get a double-ruled border for a table. Example:

```
\frame{\table{|c||l||r|}{\crl
  \multispan3\vrule\hss\bf Title\hss
                  \vrule\tabstrut \crl
                  \noalign{\kern\hhkern}\crli
  first & second & third  \crlli
  seven & eight  & nine   \crli}}
```

creates the following result:

Title		
first	second	third
seven	eight	nine

The rule width of tables (and the implicit width of all `\vrule`s and `\hrule`s) can be set by the command `\rulewidth=`⟨*dimen*⟩. The default value set by TeX is 0.4pt.

Images

The command

`\inspic` ⟨*filename*⟩`.`⟨*extension*⟩⟨*space*⟩

inserts the image in the file ⟨*filename*⟩`.`⟨*extension*⟩. Before the first `\inspic` command, you have to set the picture width with `\picw=`⟨*dimen*⟩. Images can be in PNG, JPG, JBIG2 or PDF format. The `\inspic` command works with pdfTeX only.

PDF transformations

All typesetting elements are transformed in pdfTeX by a linear transformation given by the current transformation matrix. The `\pdfsetmatrix {`⟨*a*⟩ ⟨*b*⟩ ⟨*c*⟩ ⟨*d*⟩`}` command creates an internal multiplication with the current matrix, so linear transformations can be composed. The commands `\pdfsave` and `\pdfrestore` allow for storing and restoring the current transformation matrix.

OPmac provides the macros

`\pdfscale{`⟨*horizontal-factor*⟩`}{`⟨*vertical-factor*⟩`}`
`\pdfrotate{`⟨*angle-in-degrees*⟩`}`

These macros simply expand to the proper `\pdfsetmatrix` command.

Footnotes and marginal notes

Plain TeX's macro `\footnote` is not redefined, but a new macro `\fnote{`⟨*text*⟩`}` is defined. The footnote mark is added automatically and it is numbered on each page from one. The ⟨*text*⟩ is scaled by `\typoscale[800/800]`. The footnote mark is typeset with `\def\thefnote{$^{\locfnum}$)}` by default. Users can redefine this; for example:

```
\def\thefnote{\ifcase\locfnum\or
   *\or**\or***\or$^{\dag}$\or
   $^{\ddag}$\or$^{\dag\dag}$\fi}
```

The `\fnote` macro is fully applicable only in "normal outer" paragraphs. It doesn't work inside boxes (tables for example). If you are in such a case, you can use `\fnotemark`⟨*number*⟩ inside the box (only the footnote mark is generated). When the box is finished you then use `\fnotetext{`⟨*text*⟩`}` to define the text for footnote ⟨*number*⟩. The ⟨*number*⟩ after `\fnotemark` has to be `1` if only one such command is in the box. The second `\fnotemark` inside the same box have to use the value `2` etc. The same number of `\fnotetext`s have to be defined after the box as the number of `\fnotemark`s inserted inside the box.

Marginal notes can be printed by the macro `\mnote{`⟨*text*⟩`}`. The ⟨*text*⟩ is placed in the right margin on odd pages and the left margin on even pages. This is done after a second TeX run because the relevant information is stored in an external file. If you want to place the notes only to a fixed margin, write `\fixmnotes\right` or `\fixmnotes\left`.

The ⟨*text*⟩ is formatted as a little paragraph with maximal width `\mnotesize`, ragged right in the left margins and ragged left in the right margins. The first line of this little paragraph is at the same height as the invisible mark created by `\mnote` in the current paragraph. Exceptions are possible via the `\mnoteskip` register. You can implement such exceptions to each `\mnote` manually, e.g., in a final printing so that margin notes do not overlap.

BibTeXing

The command `\cite[`⟨*label*⟩`]` makes citations of the form [42]. Multiple citation labels are also allowed, as in `\cite[`⟨*label1*⟩`,`⟨*label2*⟩`,`⟨*label3*⟩`]` producing [15, 19, 26]. If `\shortcitations` is given at the beginning of the document then continuous sequences of numbers are collapsed: [3–5, 7, 9–11].

The printed numbers correspond to the same numbers generated in the list of references. This list can be created manually by `\bib[`⟨*label*⟩`]` command for each entry. Example:

```
\bib[tbn] P. Olšák. {\it\TeX{}book naruby.}
          468~p. Brno: Konvoj, 1997.
\bib[tst] P. Olšák.
          {\it Typografický systém \TeX.}
          269~p. Praha: CSTUG, 1995.
```

There are two other possibilities which use BibTeX. The first is based to the command

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

which creates the list of cited entries and entries indicated by `\nocite[`⟨*label*⟩`]`. After the first TeX run, `\jobname.aux` is created, so users have to run BibTeX with the command `bibtex` ⟨*document*⟩. After a second TeX run, BibTeX's output is read, and after a third run all references are properly created.

The second possibility is based on a pre-generated `.bbl` file by BibTeX. You can create the temporary file (`mybase.tex`, let's say) which looks like this:

```
\input opmac
\genbbl{⟨bib-base⟩}{⟨bst-style⟩}
\end
```

After a first TeX run, `mybase.aux` is generated. Then you can run `bibtex mybase` which generates the `.bbl` file with all entries from the ⟨*bib-base*⟩`.bib` file. The second TeX run on the file `mybase.tex` generates the printed form of the list of all bib entries with labels. Finally you can insert to your real document one of the following commands:

```
    % print all entries from mybase.bbl (a=all):
\usebbl/a mybase
    % print only \cited and \nocited entries
    % sorted by mybase.bbl  (b=bbl):
\usebbl/b mybase
    % print only \cited and \nocited entries
    % sorted by \cite-order (c=cite):
\usebbl/c mybase
```

Sometimes a pure LaTeX command occurs (unfortunately) in a `.bib` database or BibTeX style. OPmac users can define such commands in the `\bibtexhook` macro, which is expanded inside the group before the `.bbl` file is read. Example:

```
\def\bibtexhook{
   \def\emph##1{{\em##1}}
   \def\frac##1##2{{##1\over##2}}
}
```

Setting the margins

OPmac declares common paper formats: `a4`, `a4l` (landscape a4), `a5`, `a5l`, `b5`, and `letter`; users can declare their own format using `\sdef`:

```
\sdef{pgs:b5l}{(250,176)mm}
\sdef{pgs:letterl}{(11,8.5)in}
```

The `\margins` command declares the margins of the document. This command has the following parameters:

`\margins/`⟨*pg*⟩ ⟨*fmt*⟩ `(`⟨*left*⟩`,`⟨*right*⟩`,`⟨*top*⟩`,`⟨*bot*⟩`)`⟨*unit*⟩

For example:

```
\margins/1 a4 (2.5,2.5,2,2)cm
```

These parameters are:

- ⟨*pg*⟩: `1` or `2` specifies single-page or double-page (spread) design.
- ⟨*fmt*⟩: paper format (`a4`, `a4l`, etc.).
- ⟨*left*⟩, ⟨*right*⟩, ⟨*top*⟩, ⟨*bot*⟩: specifies the left, right, top and bottom margins.
- ⟨*unit*⟩: unit used for the ⟨*left*⟩, ⟨*right*⟩, ⟨*top*⟩, ⟨*bot*⟩ values.

Any of the parameters ⟨*left*⟩, ⟨*right*⟩, ⟨*top*⟩, ⟨*bot*⟩ can be empty. If both ⟨*left*⟩ and ⟨*right*⟩ are nonempty then `\hsize` is set. Else `\hsize` is unchanged. If both ⟨*left*⟩ and ⟨*right*⟩ are empty then typesetting area is centered in the paper format. The analogous case holds when ⟨*top*⟩ or ⟨*bot*⟩ parameter is empty (for `\vsize` instead of `\hsize`). Examples:

```
   % \hsize, \vsize untouched,
   % typesetting area centered:
\margins/1 a4 (,,,)mm
   % right margin set to 2cm
   % \hsize, \vsize untouched,
   % vertically centered:
\margins/1 a4 (,2,,)cm
```

If ⟨*pg*⟩`=1` then all pages have the same margins. If ⟨*pg*⟩`=2` then the declared margins are used for odd pages, and the margins of even pages are mirrored, i.e. ⟨*left*⟩ is replaced by ⟨*right*⟩ and vice versa.

The command `\magscale[`⟨*factor*⟩`]` scales the whole typesetting area. The fixed point of such scaling is the so-called “Knuthian origin”: 1in below and 1in right of paper sides. Typesetting (breakpoints etc.) is unchanged. Almost all units are relative after such scaling; only paper format dimensions remain unscaled. Example:

```
\margins/2 a5 (22,17,19,21)mm
\magscale[1414] \margins/1 a4 (,,,)mm
```

The first line sets the `\hsize` and `\vsize` and margins for final printing at a5 format. The setting on the second line centers the scaled typesetting area to the true a4 paper while breakpoints for paragraphs and pages are unchanged. It may be useful for a proof copy printed at a larger size. After the review is done, the second line can be commented out.

◇ Petr Olšák
Czech Technical University
in Prague,
Czech Republic