

# OPmac – macros for plain $\text{\TeX}$ <sup>1</sup>

*Petr Olsák, 2012 – 2016*

<http://petr.olsak.net/opmac-e.html>

## Contents

Introduction . . . . .	1
0 Using OPmac . . . . .	2
1 Selection of font family . . . . .	2
2 Font sizes . . . . .	3
3 Parts of the document . . . . .	4
4 Another numbered objects . . . . .	4
5 Lists . . . . .	5
6 Table of contents . . . . .	6
7 Making the index . . . . .	6
8 Colors . . . . .	8
9 Hyperlinks, outlines . . . . .	8
10 Verbatim . . . . .	9
11 Tables . . . . .	10
12 Images . . . . .	11
13 PDF transformations . . . . .	12
14 Footnotes and marginal notes . . . . .	12
15 Bib $\text{\TeX}$ ing . . . . .	13
16 Typesetting math . . . . .	14
17 Setting the margins . . . . .	15
18 The last page . . . . .	16
19 Using other language . . . . .	16
20 Summary . . . . .	18

## Introduction

The OPmac package is a set of simple additional macros to plain $\text{\TeX}$ . It enables users to take advantage of basic  $\text{\LaTeX}$  functionality: font size selection, the automatic creation of a table of contents and an index, working with bibliography databases, tables, references with optional hyperlinks, margin settings, etc.

I had been using these macros personally for a long time, but now, after cleaning up the code a bit and providing both user and technical documentation, I'm releasing them to the general public along with the new version of  $\mathcal{C}\text{\S}plain$ .

My main motivation in publishing OPmac is to provide a set of macros with solutions to common tasks for plain $\text{\TeX}$  users. Additionally, I wanted to demonstrate that it is possible to write  $\text{\TeX}$  code in a simple and effective style, something that most  $\text{\LaTeX}$  macro packages lack. All of OPmac's macros are contained within the single file `opmac.tex` in only 1,700 lines. By comparison, the  $\text{\LaTeX}$  code which solves comparable tasks is placed inside its “kernel” along with many  $\text{\LaTeX}$  packages all of which contain tens of thousands of lines.

<sup>1</sup> This text is second revised version. The first version was published in TUGboat 34:1, 2013, pp. 88–96

The main principles which I followed when creating this macro package, are:

- Simplicity is power.
- Macros are not universal, but are readable and understandable.
- User can easily redefine these macros as he/she wishes.

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

OPmac package offers a markup language for authors of texts (like L<sup>A</sup>T<sub>E</sub>X), i.e. the fixed set of tags to define the structure of the document. This markup is different from the L<sup>A</sup>T<sub>E</sub>X markup. It may offer to write the source text of the document somewhat clearer and more attractive. The OPmac package, however, does not care for the typography of the document. The simple sober document is created if no additional macros are used. We assume that the author of additional macros is able to create a look of the document to suit specific requirement.

OPmac has a small number of additional packages: `fontfam`, `pdfuni`, `opmac-xetex` and `opmac-bib` (see the end of <http://petr.olsak.net/opmac-e.html> page). Moreover, there exist many tens of little OPmac tricks comparable with L<sup>A</sup>T<sub>E</sub>X packages mentioned on <http://petr.olsak.net/opmac-tricks-e.html> page.

## 0 Using OPmac

OPmac is not compiled as a format. For using it in plainT<sub>E</sub>X, you can simply `\input opmac` at the beginning of your document. The example of the simple document follows:

```
\input opmac
\typosize[11/13]    % setting the basic font size and the baselineskip
\margins/1 a4 (1,1,1,1)in  % setting 1in margins for A4 paper

Here is a text.
\bye
```

You can use T<sub>E</sub>X, pdfT<sub>E</sub>X, XeT<sub>E</sub>X or luaT<sub>E</sub>X with plain T<sub>E</sub>X format (classical `plain.tex`, `etex.src` or `Csplain`). The `Csplain` is recommended but it is not explicitly requested if you don't need to use Czech/Slovak specific features<sup>1</sup>.

## 1 Selection of font family

OPmac doesn't select a special default font family. The default is the same as in plain T<sub>E</sub>X (CM fonts) or `Csplain` (CS fonts). It is possible to `\input` so called "font-files" for loading a font family, i.e. typically four variants of fonts `\rm`, `\bf`, `\it` and `\bi`. The font-files use primitive command `\font` for loading individual fonts.

You need not to remember names: use `\fontfam[⟨FamilyName⟩]` macro which loads the appropriate font-file. The argument `⟨FamilyName⟩` is case insensitive and spaces are ignored. So, `\fontfam[Times Roman]` is equal to `\fontfam[TimesRoman]` and it is equal to `\fontfam[timesroman]`. Several aliases are prepared, thus `\fontfam[times]` can be used for loading Times Roman family too.

If you write `\fontfam[?]` then all available font families are listed on the terminal and in the log file. The listing looks like:

```
[LM Fonts]  {\caps \sans \ttset ...} {\rm \bf \it \bi } +AMS (8z 8t u)
[TG Heros]  {\caps \cond} {\rm \bf \it \bi } +TX (8z 8t)
...
```

The `⟨FamilyName⟩` is followed by the list of *modifiers* of basic variant selectors, then available basic variant selectors are listed. After plus character, the default set of math fonts

---

<sup>1</sup> Warnings: "falling back to ASCII sorting" or "CZ/SK outline-conversion is off" may occur without `Csplain`.

used together with given family is named. The available font encodings are written in round brackets. More information about `\fontfam` macro can be found in the `fontfam.tex` file.

The variants are selected by basic selectors (`\rm`, `\bf`, `\it`, `\bi`). The modifiers of basic variants (`\caps`, `\cond` for example) can be used immediately before a basic variant selector and they can be (independently) combined: `\caps\it` or `\cond\caps\bf`. The modifiers can be followed by `\fam` command (instead of basic variant selector). Then current variant is kept (but modified) and all consecutive basic variant selectors (in a group) are modified too. If modifiers are followed by `\one` sequence (instead variant selector) then only current variant is modified. More about font modifiers are mentioned in the `cs-heros.tex` file and in the article `kpfonts-plain.pdf`.

The `\fontfam[Catalog]` prints a font catalogue of all configured font families.

## 2 Font sizes

The commands for font size setting described below, for variant selectors and modifiers described above have local validity. If you put them into a group, the font features are selected locally.

The command `\typosize[fontsize/baselineskip]` sets the font size of text and math fonts and baselineskip. If one of these two parameters is empty, the corresponding feature stays unchanged. The metric unit is supposed `pt` and this unit isn't written in parameters. You can change the unit by the command `\ptunit=something-else`, for instance `\ptunit=1mm`. Examples:

```
\typosize[10/12]    % default of plainTeX
\typosize[11/12.5]  % font 11pt, baseline 12.5pt
\typosize[8/]       % font 8pt, baseline unchanged
```

The command `\typoscale[font-factor/baselineskip-factor]` sets the text and math fonts size and baselineskip as a multiple of the current fonts size and baselineskip. The factor is written in `scaled`-like way, it means that 1000 means factor one. The empty parameter is equal to the parameter 1000, i.e. the value stays unchanged. Examples:

```
\typoscale[800/800]    % fonts and baselineskip re-size to 80 %
\typoscale[\magstep2/] % fonts bigger 1,44times
```

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

There are times however, when one would like to make a font change relative to the documents "base" font size instead of the current font (e.g. when typesetting footnotes). The macro `\typobase` provides an easy way to perform such changes. The "base" font size is set with the first use of either `\typosize` or `\typoscale` and this size is restored by `\typobase`. Example:

```
\typosize[12/14]    % if first use of \typosize, then this sets the "base"
...
{\typoscale[16/18]  % bigger size (for example title font)
...
\typobase\typoscale[750/750] % reduced to 75% of [12/14] (i.e. [9/10.5])
...                      % this is calculated from "base", not from actual title font
}
```

The size of the current font can be changed by the command `\thefontsize[font-size]` or can be rescaled by `\thefontscale[factor]`. These macros don't change math fonts sizes nor baselineskip.

The `\resizefont`, `\regfont` and `\resizeall` commands (documented in `Csplain`) can be used even if the used format is not `Csplain`. The best design size of the font for desired size is selected. For example `\typosize[18/]` selects the font `cmr17` at 18pt.

The `\em` macro acts as `\it` if the current font is `\rm`, acts as `\rm` if the current font is `\it`, acts as `\bi` if the current font is `\bf` and acts as `\bf` if the current font is `\bi`. The `\ /` spaces are inserted automatically. Example:

```
This is {\em important} text.      % = This is {\it important\/} text.
\it This is {\em important} text. % = This is\/ {\rm important} text.
\bf This is {\em important} text. % = This is {\bi important\/} text.
\bi This is {\em important} text. % = This is\/ {\bf important} text.
```

### 3 Parts of the document

The document can be divided into chapters, sections and subsections and titled by `\tit` command. The parameters are separated by the end of current line (no braces are used):

```
\tit Document title <end of line>
\chap Chapter title <end of line>
\sec Section title <end of line>
\secc Subsection title <end of line>
```

The chapters are numbered by one number, sections by two numbers (chapter.section) and subsections by three numbers. If there are no chapters then section have only one number and subsection two.

The implicit design of the titles of chapter etc. are implemented in the macros `\printchap`, `\printsec` and `\printsecc`. User can simply change these macros if he/she needs another behavior.

The first paragraph after the title of chapter, section and subsection is not indented but you can type `\let\firstnoindent=\relax` if you need all paragraphs indented.

If a title is so long then it breaks to more lines. It is better to hint the breakpoints because `TeX` does not interpret the meaning of the title. User can put the `\nl` (it means newline) macro to the breakpoints.

The chapter, section or subsection isn't numbered if the `\nonum` precedes. And the chapter, section or subsection isn't delivered to the table of contents if `\notoc` precedes.

### 4 Another numbered objects

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

If user write the `\eqmark` as the last element of the display mode then this equation is numbered. The format is one number in brackets. This number is reset in each section.

If the `\eqalignno` is used, then user can put `\eqmark` to 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 caption which is tagged by `\caption/t` for tables and `\caption/f` for figures. Example:

```
\hfil\table{r1}{
  age   & value \crl\noalign{\smallskip}
  0--1   & unmeasured \cr
  1--6   & observable \cr
  6--12  & significant \cr
  12--20 & extremal \cr
  20--40 & normal \cr}
```

```

40--60 & various \cr
60--$\infty$ & moderate}
\par\nobreak\medskip
\caption/t The dependency of the computer-dependency on the age.

```

This example produces:

age	value
0–1	unmeasured
1–6	observable
6–12	significant
12–20	extremal
20–40	normal
40–60	various
60– $\infty$	moderate

**Table 2.3** The dependency of the computer-dependency on the 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 more lines then the last line is centered.

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

If you wish to make the table or figure as floating object, you need to use plain $\TeX$  macros `\midinsert`, `\topinsert` and `\endinsert`.

Each automatically numbered object can be referenced, if the `\label[⟨label⟩]` command precedes. The reference commands are `\ref[⟨label⟩]` and `\pgref[⟨label⟩]`. Example:

```

\label[beatle] \sec About Beatles

\label[comp-dependence]
\hfil\table{rl}{...} % the table
\caption/t The dependency of the computer-dependency on the 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]`.

If there are forward referenced objects then user have to run  $\TeX$  twice. During each pass, the working `*.ref` file (with refereces data) is created and this file is used (if it exists) at the begin of the document.

You can create a reference to whatever else by commands `\label[⟨label⟩]\wlabel{⟨text⟩}`. The connection between `⟨label⟩` and `⟨text⟩` is established. The `\ref[⟨label⟩]` will print `⟨text⟩`.

## 5 Lists

The list of items is surrounded by `\beginitems` 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 `\beginitems`:

```

\style o % small bullet
\style O % big bullet (default)
\style - % hyphen char
\style n % numbered items 1., 2., 3., ...
\style N % numbered items 1), 2), 3), ...
\style i % numbered items (i), (ii), (iii), ...
\style I % numbered items I, II, III, IV, ...
\style a % items of type a), b), c), ...
\style A % items of type A), B), C), ...
\style x % small rectangle
\style X % big rectangle

```

Another style can be defined by the command `\sdef{item:⟨style⟩}{⟨text⟩}`. Default item can be redefined by `\def\normalitem{⟨text⟩}`. The list environments can be nested. Each new level of item is indented by next multiple of `\iindent` which is set to `\parindent` by default. The vertical space at begin and end of the environment is inserted by the macro `\iiskip`.

## 6 Table of contents

The `\maketoc` command prints the table of contents of all `\chap`, `\sec` and `\secc` used in the document. These data are read from external `*.ref` file, so you have to run  $\mathrm{T\!E\!X}$  more than once (typically three times if the table of contents is at the beginning of the document).

The name of the section with table of contents is not printed. The direct 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 unreferenced title of the section by the code:

```
\nonum\notoc\sec Table of Contents
```

The title of chapters etc. are written into the external file and they are read from this file in a next run of  $\mathrm{T\!E\!X}$ . This technique can induce some problems when a somewhat complicated macro is used in the title. OPmac solves this problem by different way than  $\mathrm{L\!A\!T\!E\!X}$ . User can set the problematic macro as “robust” by `\addprotect\macro` declaration. The `\macro` itself cannot be redefined. The common macros used in OPmac which can be occur in the titles are declared by this way. For example:

```
\addprotect~ \addprotect\TeX \addprotect\thefontsize \addprotect\em
```

## 7 Making the index

The index can be included into document by `\makeindex` macro. No external program is needed, the alphabetical sorting are done inside  $\mathrm{T\!E\!X}$  at macro level.

The `\ii` command (insert to index) declares the word separated by the space as the index item. This declaration is represented as 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 cannot double the word if you use the `\iid` instead `\ii`:

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

or:

```
Now we'll deal with the \iid resistor .
```

Note that the dot or comma have to be separated by space when `\iid` is used. This space (before dot or comma) is removed by the macro in the current text.

The multiple-words entries are commonly organized in the index by the format (for example):

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

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

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

The number of the parts of one index entry is unlimited. Note, that you can spare your typing by the comma in the \ii parameter. The previous example is equivalent to \ii linear/space \ii vector/space.

Maybe you need to propagate to the index the similar entry to the linear/space in the form space/linear. You can do this by the shorthand ,@ at the end of the \ii parameter. Example:

```
\ii linear/space,vector/space,@
is equivalent to:
\ii linear/space,vector/space \ii space/linear,space/vector
```

If you really need to insert the space into the index entry, write “~”.

The \makeindex creates the list of alphabetically sorted index entries without the title of the section and without creating more columns. OPmac provides another macros for more columns:

```
\begmulti <number of columns>
<text>
\endmulti
```

The columns will be balanced. The Index can be printed by the following code:

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

Only “pure words” can be propagated to the index by the \ii command. It means that there cannot be any macro, T<sub>E</sub>X primitive, math selector etc. But there is another possibility to create such complex index entry. Use “pure equivalent” in the \ii parameter and map this equivalent to the real word which is printed in the index by \iis command. 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 the <equivalent> but the <text> is printed in the index entry list.

The special sorting by the Czech or Slovak standard of alphabetical sorting is activated if C<sub>S</sub>plain is used and if \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.



## 8 Colors

The colors selection macros are working only if pdfTeX-like engine (or XeTeX) is used. OPmac provides a small number of color selectors: `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow`, `\Cyan`, `\Magenta`, `\White`, `\Grey`, `\LightGrey` and `\Black`. User can define more such selectors by setting the CMYK components. For example

```
\def\Orange{\setcmykcolor{0 0.5 1 0}} yes see petr-olsak.mac line 870
```

The current color in CMYK format is saved in the `\currentcolor` macro, thus you can save it to your macro `\let\yourmacro=\currentcolor` and you can return to this color by `\setcmykcolor\yourmacro`.

The color selectors work globally by default. It means that colors don't respect the TeX groups and you have to return back to the black typesetting explicitly by the `\Black` selector.

OPmac provides the macro `\localcolor`. If it is used then the colors return back to the original value after TeX groups automatically. The macro has local validity. You can use it at begin of your document (for all TeX groups) or only in selected TeX group (for this group and nested groups). Example:

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

The more usable example follows. It defines a macro which creates the **colored text on the colored background**. Usage: `\coloron<background><foreground>{<text>}`

The `\coloron` can be defined as follows:

```
\def\coloron#1#2#3{%
  \setbox0=\hbox{#3}\leavevmode
  {\localcolor\rlap{#1\strut \vrule width\wd0}#2\box0}%
}
\coloron\Yellow\Brown{The brown text on the yellow background}
```

**The watermark** is grey text on the background of the page. OPmac offers an example: the macro `\draft` which creates grey scaled and rotated text DRAFT on the background of every page.

## 9 Hyperlinks, outlines

If the command `\hyperlinks{<color-in>}{<color-out>}` is used at the beginning of the file, then the following objects 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-out>`. Others links are internal and they are colored by `<color-in>`. Example:

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

You can use another marking of active links: by frames which are visible in the PDF viewer but invisible when the document is printed. The way to do it is to define the macros `\pgborder`, `\tocborder`, `\citeborder`, `\refborder` and `\urlborder` as the triple of RGB components of the used color. Example:



```

\def\tocborder {1 0 0} % links in table of contents: 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. It means that no frames are created.

There are “low level” commands to create the links. You can specify the destination of the internal link by `\dest[⟨type⟩:⟨label⟩]`. The active text linked to the `\dest` can be created by `\link[⟨type⟩:⟨label⟩]{⟨color⟩}{⟨text⟩}`. The `⟨type⟩` parameter is one of the `toc`, `pg`, `cite`, `ref` or another special for your purpose.

The `\url` macro prints its parameter in `\tt` font and creates a potential breakpoints in it (after slash or dot, for example). If `\hyperlinks` declaration is used then the parameter of `\url` is treated as an external URL link. An example: `\url{http://www.olsak.net}` creates <http://www.olsak.net>. The characters `%`, `\`, `#`, `$`, `{` and `}` have to be protected by backslash in the `\url` argument, the other special characters `~`, `_`, `^`, `&` can be written as single character. You can insert the `\|` command in the `\url` argument as a potential breakpoint.

If the linked text have to be different than the URL, you can use `\ulink[⟨url⟩]{text}` macro. For example:

```

\ulink[http://petr.olsak.net/opmac-e.html]{OPmac page}

```

creates [OPmac page](http://petr.olsak.net/opmac-e.html).

The PDF format provides “outlines” which are notes placed in the special frame of the PDF viewer. These notes can be managed as structured and hyperlinked table of contents of the document. The command `\outlines{⟨level⟩}` creates such outlines from data used for table of contents in the document. The `⟨level⟩` parameter gives the level of opened sub-outlines in the default view. The deeper levels can be open by mouse click on the triangle symbol after that.

The strings used in PDF outlines are converted if `ℒSplain` is used: the accents are stripped off because they can make problems in outlines. But user can use `\input pdfuni` in order to convert these strings to internal UNICODE representation.

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

## 10 Verbatim

The display verbatim text have to be surrounded by the `\begtt` and `\endtt` couple. The inline verbatim have to be tagged (before and after) by a character which is declared by `\activettchar⟨char⟩`. For example `\activettchar"` declares the `"` for inline verbatim markup.

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

The indentation of each line in display verbatim is controlled by `\ttindent` register. This register is set to the `\parindent` when `opmac.tex` is read. User have to change its value if the `\parindent` is changed after reading of `opmac.tex`.

The `\begtt` starts internal group in which the catcodes are changed. Then the `\tthook` macro is run. This macro is empty by default and user can control fine behavior by it. For example the cactodes can be reset here. If you need to define active character in the `\tthook`, use `\adef` as in the following 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 can't worry about active categories.

There are tips for global `\tthook` definitions here:

```
\def\tthook{\typosize[9/11]}      % setting font size for verbatim
\def\tthook{\ttline=0}             % each listing will be numbered from one
\def\tthook{\adef{ }\char'\ }    % visualisation of spaces
```

You can print verbatim listing from external files by `\verbatiminput` command. Examples:

```
\verbatiminput (12-42) program.c % listing from program.c, only lines 12-42
\verbatiminput (-60) program.c    % print from begin to the line 60
\verbatiminput (61-) program.c    % from line 61 to the end
\verbatiminput (-) program.c      % whole file is printed
\verbatiminput (70+10) program.c  % from line 70, only 10 lines printed
\verbatiminput (+10) program.c    % from the last line read, print 10 lines
\verbatiminput (-5+7) program.c   % from the last line read, skip 5, print 7
\verbatiminput (+) program.c      % from the last line read to the end
```

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

The `\verbatiminput` can be controlled by `\tthook`, `\ttindent` just like in `\begtt...\endtt`.

## 11 Tables

The macro `\table{<declaration>}{<data>}` provides similar `<declaration>` as in `LATEX`: you can use letters `l`, `r`, `c`, each letter declares one column (aligned to left, right, center respectively). These letters can be combined by the “|” character (vertical line). Example

```
\table{||lcr||}{
  Month      & commodity & price      \crl
  January    & notebook  & \$ 700  \crli \tskip.2em
  February   & skateboard & \$ 100  \cr
  July       & yacht      & k\$ 170  \crl}
```

generates the following result:

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

Apart from `l`, `r`, `c` declarators, you can use the `p{<size>}` declarator which declares the column of given width. More precisely, a long text in the table cell is printed as an paragraph with given width. To avoid the problems with narrow left-right aligned paragraphs you can write `p{<size>\raggedright}`, then the paragraph will be only left aligned.

An arbitrary part of the `<declaration>` can be repeated by a `<number>` prefixed. For example “`3c`” means “`ccc`” or “`c 3{|c}`” means “`c|c|c|c`”. Note that spaces in the `<declaration>` are ignored and you can use them in order to more legibility.

The command `\cr` used in the `<data>` part of the table (the end row separator) is generally known. Moreover `OPmac` defines following similar commands:

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

- `\crlp{<list>}` ... like `\crli` but the lines are drawn only in the columns mentioned in comma separated `<list>` of their numbers. The `<list>` can include `<from>-<to>` declarators, for example `\crlp{1-3,5}` is equal to `\crlp{1,2,3,5}`.

The `\tskip<dimen>` command works like the `\noalign{\vskip<dimen>}` after `\cr*` commands but it doesn't interrupt the vertical lines.

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

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

If you do `\def\tabiteml{${\enspace}}\def\tabitemr{{\enspace$}}` then the `\table` acts like L<sup>A</sup>T<sub>E</sub>X's array environment.

If there is an item which spans to more than one column in the table then you can use `\multispan{<number>}` macro from plain T<sub>E</sub>X or `\mspan{<number>}[<declaration>]{<text>}` from O<sub>P</sub>mac, which spans `<number>` columns and formats the `<text>` by the `<declaration>`. The `<declaration>` must include exactly one letter “c” or “l” or “r” and may include characters “|” for vertical rules. If your table includes vertical rules and you want to create continuous vertical rules by `\mspan`, then use rules in only after “c”, “l” or “r” letter in `\mspan <declaration>`. The exception is only in the case when `\mspan` includes first column and the table have rules on the left side. The example of `\mspan` usage is below.

The `\frame{<text>}` makes a frame around `<text>`. You can put the whole `\table` into `\frame` if you need double-ruled border of the table. Example:

```
\frame{\table{|c||l||r|}{ \crl
\mspan3[|c|]{\bf Title} \crl \noalign{\kern\hhkern}\crl
first & second & third \crl
seven & eight & nine \crl}}
```

creates the following result:

Title		
first	second	third
seven	eight	nine

The c, l, r and p are default `<declaration>` letters but you can define more such letters by `\def\tabdeclare<letter>{<left>##<right>}`. The technical documentation `opmac-d.pdf` describes this feature more precisely.

The rule width of tables (and implicit width of all `\vrules` and `\hrules`) can be set by the command `\rulewidth=<dimen>`. The default value given by T<sub>E</sub>X is 0.4pt.

Many tips about tables can be seen on <http://petr.olsak.net/opmac-tricks-e.html>.

## 12 Images

The `\inspic <filename>.<extension><space>` inserts the picture stored in the graphics file with the name `<filename>.<extension>`. You can set the picture width by `\picw=<dimen>` before first `\inspic` command which declares the width of the picture. The files can be in the PNG, JPG, JBIG2 or PDF format. The `\inspic` command works with pdfT<sub>E</sub>X/XeT<sub>E</sub>X only.

The `\picwidth` is an equivalent register to `\picw`. Moreover there is an `\picheight` register which denotes the height of the picture. If both registers are set then the picture will be (probably) deformed.

The file is searched in `\picdir`. This macro is empty by default, this means that the file is searched in current directory.

## 13 PDF transformations

All typesetting elements are transformed in pdfTeX by linear transformation given by the current transformation matrix. The `\pdfsetmatrix {⟨a⟩ ⟨b⟩ ⟨c⟩ ⟨d⟩}` command makes the internal multiplication with the current matrix so linear transformations can be composed. The stack-oriented commands `\pdfsave` and `\pdfrestore` gives a possibility of storing and restoring the current transformation matrix and current point. The position of current point have to be the same from TeX's point of view as from transformation point of view when `\pdfrestore` is processed. Due to this fact the `\pdfsave\rlap{⟨transformed text⟩}\pdfrestore` or something similar is recommended.

OPmac provides the macros

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

These macros simply calls the properly `\pdfsetmatrix` primitive command.

It is known that the composition of transformations is not commutative. It means that the order is important. You have to read the transformation matrices from right to left. Example:

```
First: \pdfsave \pdfrotate{30}\pdfscale{-2}{2}\rlap{text1}\pdfrestore
      % text1 is scaled two times and it is reflected about vertical axis
      % and next it is rotated by 30 degrees left.
second: \pdfsave \pdfscale{-2}{2}\pdfrotate{30}\rlap{text2}\pdfrestore
      % text2 is rotated by 30 degrees left then it is scaled two times
      % and reflected about vertical axis.
third: \pdfsave \pdfrotate{-15.3}\pdfsetmatrix{2 0 1.5 2}\rlap{text3}%
      \pdfrestore % first slanted, then rotated by 15.3 degrees right
```

This gives the following result. First: second: third:

*text1* *text2* *text3*

## 14 Footnotes and marginal notes

The plainTeX'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<sup>1</sup>. The `⟨text⟩` is scaled by `\typoscale[800]`. The implicit visual aspect of the footnote mark is defined by `\def\thefnote{$~{\locfnum}$}`. User can redefine it, for example:

```
\def\thefnote{\ifcase\locfnum\or *\or**\or***\or$~{\dag}$\or
  $\{ \ddag \}$\or$~{\dag}\dag \}$\fi}
\hbox{abc .\fnotemark 1}
NOT {1}
```

The `\fnote` macro is fully applicable only in "normal outer" paragraph. It doesn't work inside boxes (tables for example). If you are solving such case you can use `\fnotemark⟨number⟩` inside the box (only the footnote mark is generated). When the box is finished you can use `\fnotetext{⟨text⟩}`. This macro puts the `⟨text⟩` to the footnote. The `⟨number⟩` after `\fnotemark` have to be 1 if only one such command is in the box. Second `\fnotemark` inside the same box have to have the parameter 2 etc. The same number of `\fnotetexts` have to be written after the box as the number of `\fnotemarks` inserted inside the box.

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

<sup>1</sup> This behavior is changed if `\runningfnotes` is used: the footnotes are numbered from one in whole document in such case. Alternatives are possible, see OPmac tricks or technical documentation.

The  $\langle text \rangle$  is formatted as a little paragraph with the maximal width `\mnotesize` ragged left on the left margins or ragged right on 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. The exceptions are possible by `\mnoteskip` register. You can implement such exceptions to each `\mnote` manually in final printing in order to margin notes do not overlap. The positive value of `\mnoteskip` shifts the note up and negative value shifts it down. For example `\mnoteskip=2\baselineskip \mnote{\langle text \rangle}` shifts this (and only this) note two lines up.

## 15 BibTeXing

The command `\cite[\langle label \rangle]` or its variations of the type `\cite[\langle label-1 \rangle, \langle label-2 \rangle, \langle label-3 \rangle]` create the citations in the form [42] or [15, 19, 26]. If `\shortcitations` is declared at the beginning of the document then continuous sequences of numbers are re-printed like this: [3–5, 7, 9–11]. If `\sortcitations` is declared then numbers generated by one `\cite` command are sorted upward.

If `\nonumcitations` is used then the marks instead numbers are generated depending on the used bibTeX style. For example the citations look like [Now08] when `alpha` style is used and like [Nowak, 2008] when `apalike` style is used.

The `\rcite[\langle labels \rangle]` creates the same list as `\cite[\langle labels \rangle]` but without the outer brackets. Example: `[\rcite[tbn], pg.~13]` creates [4, pg.13].

The `\ecite[\langle label \rangle]{\langle text \rangle}` prints the  $\langle text \rangle$  only, but the entry labeled  $\langle label \rangle$  is decided as to be cited. If `\hyperlinks` is used then  $\langle text \rangle$  is linked to the references list.

You can define alternative formatting of `\cite` command. Example:

```
\def\cite[#1]{(\rcite[#1])} % \cite[\langle label \rangle] creates (27)
\def\cite[#1]{${\rcite[#1]}$} % \cite[\langle label \rangle] creates^{27}
```

The numbers printed by `\cite` correspond to the same numbers generated in the list of references. There are four possibilities to generate this references list:

- Manually using `\bib[\langle label \rangle]` commands.
- Using bibTeX and `\usebibtex{\langle bib-base \rangle}{\langle bib-style \rangle}` command.
- Using pregenerated `*.bbl` file and `\usebbl/\langle type \rangle \langle bbl-base \rangle` command.
- By `\usebib/\langle type \rangle (\langle style \rangle) \langle bbl-base \rangle` command which reads `*.bib` databases directly.

These possibilities are documented here in detail:

### References created manually using `\bib[\langle label \rangle]` command.

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

If you are using `\nonumcitations` then you need to declare the  $\langle marks \rangle$  used by `\cite` command. To do it you have to use long form of the `\bib` command which is in the format `\bib[\langle label \rangle] = {\langle mark \rangle}`. The spaces around equal sign are mandatory. Example:

```
\bib [tbn] = {Olšák, 2001}
P. Olšák. {\it\TeX{}}book naruby.} 468~s. Brno: Konvoj, 2001.
```

**References using bibTeX.** The command `\usebibtex{\langle bib-base \rangle}{\langle bst-style \rangle}` creates the list of cited entries and entries indicated by `\nocite[\langle label \rangle]`. After first TeX run the `*.aux` file is created, so user have to run the bibTeX by the command `bibtex \langle document \rangle`. After second TeX run the bibTeX's output is read and after third TeX run all references are properly created.

The  $\langle bib-base \rangle$  is one or more `*.bib` database source files (separated by spaces and without extension) and the  $\langle bst-style \rangle$  is the style used by bibTeX. The common styles are `plain`, `alpha`, `apalike`, `ieeetr`, `unsrt`.

**Using pregenerated \*.bbl file by bibTeX.** You can create the temporary file (`mybase.tex`, for example) which looks like:

```
\input opmac
\genbbl{<bib-base>}{<bst-style>}
\end
```

After first T<sub>E</sub>X run the `mybase.aux` is generated. Then you can run `bibtex mybase` which generates the `.bbl` file with all entries from the `<bib-base> *.bib` file(s). Second T<sub>E</sub>X run on the file `mybase.tex` generates the printed form of the list of all bib entries with labels. This is usable printed matter, you can place it to your notice board when you create your document. Finally you can insert to your real document one of the following commands:

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

Sometimes the pure L<sup>A</sup>T<sub>E</sub>X command occurs (unfortunately) in the `.bib` database or bibT<sub>E</sub>X style. User can define such commands in the `\bibtexhook` macro which is a hook started inside the group before `.bbl` file is read. Example:

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

**Direct reading of .bib files** is possible by `\usebib` macro. This macro reads macro package `opmac-bib.tex` (on demand) which uses the external package `librarian.tex` by Paul Isambert. The usage is similar to previous case:

```
% print only \cited and \nocited entries
\usebib/c (<style>) <bib-base> % sorted by \cite-order (c=cite),
\usebib/s (<style>) <bib-base> % sorted by style (s=style).
```

The `<bib-base>` is one or more `*.bib` database source files (separated by spaces and without extension) and the `<style>` is the part of the filename `opmac-bib-<style>.tex` where the formatting of the references list is defined. Possible styles are `simple` or `iso690`. The behavior of `opmac-bib.tex` and `opmac-bib-iso690.tex` is full documented in these files (after `\endinput` command).

**Formatting of the references list** is controlled by the `\printb` macro. It is called at the begin of each entry. The default `\printb` macro prints the number of the entry in square brackets. If the `\nonumcitations` is set then no numbers are printed, only all lines (but no first one) are indented. The `\printb` macro can use the following values: `\the\bibnum` (the number of the entry) and `\the\bibmark` (the mark of the entry used when `\nonumcitations` is set). Examples:

```
% The numbers are without square brackets:
\def\printbib{\hangindent=\parindent \indent \llap{\the\bibnum. }}
% Printing of <marks> when \nonumcitations is set:
\def\printbib{\hangindent=\parindent \noindent [\the\bibmark]\quad}
```

Next examples can be found on the [OPmac tricks WWW page](#).

## 16 Typesetting math

There are two files for math typesetting prepared in C<sub>S</sub>plain:

- `ams-math.tex` loads the AMS math fonts visual compatible with Computer modern.
- `tx-math.tex` loads the TX fonts visual compatible with Times.



OPmac reads the first file `ams-math.tex` by default. If you are using font files from `Csplain` (`ctimes.tex`, `cbookman.tex`, `cs-termes.tex` etc.) then the second math-file `tx-math.tex` is loaded.

This section describes the features of the macros from `ams-math.tex` or `tx-math.tex`. More documentation is written in these files themselves.

Hundreds math symbols and operators like in AMST<sub>EX</sub> are accesible. For example `\alpha`  $\alpha$ , `\geq`  $\geq$ , `\sum`  $\sum$ , `\sphericalangle`  $\sphericalangle$ , `\bumpeq`  $\bumpeq$ ,  $\simeq$ . See AMST<sub>EX</sub> manual (or TX-fonts manual) for complete list of symbols.

The following math alphabets are available:

<code>\mit</code>	% mathematical variables	<i>abc-xyz, ABC-XYZ</i>
<code>\it</code>	% text italics	<i>abc-xyz, ABC-XYZ</i>
<code>\rm</code>	% text roman	abc-xyz, ABC-XYZ
<code>\cal</code>	% normal calligraphics	<i>ABC-<math>\mathcal{XYZ}</math></i>
<code>\script</code>	% script	<i><math>\mathcal{ABC}-\mathcal{XYZ}</math></i>
<code>\frak</code>	% fracture	<i>abc-frak, ABC-frak</i>
<code>\bbchar</code>	% double stroked letters	<b>ABC-XYZ</b>
<code>\bf</code>	% sans serif bold	<b>abc-xyz, ABC-XYZ</b>
<code>\bi</code>	% sans serif bold slanted	<b><i>abc-xyz, ABC-XYZ</i></b>

The last two selectors `\bf` and `\bi` select the sans serif fonts regardless current text font family. The reason is that these shapes are used for vectors and matrices in Czech math typesetting.

The math fonts are scaled by `\typothesize` and `\typoscale` macros. Two math fonts collections are prepared: `\normalmath` for normal weight and `\boldmath` for bold. The first one is set by default. There is an example for math typesetting in titles:

```
\def\title#1\par{\centerline{\typothesize[17/]\bf\boldmath #1}}
\title The title with math $\int_a^b f(x) \, dx$ is here
```

Variables are printed by special math italics when `ams-math.tex` is loaded and by text italics of the current text font when `tx-math.tex` is loaded. You can change this behavior by following commands:

```
\itvariables % variables typeset by text italics.
\mitvariables % variables typeset by math italics.
```

**Note:** Due to the features described in this section (AMS symbols, `\bbchar`, `\script`, `\frak` characters and sans serif bold in math) more special fonts are loaded (from AMS package and from EC fonts). If you dislike such dependency, you can follow the [OPmac trick 0111](#).

## 17 Setting the margins

OPmac declares paper formats a4, a4l (landscape a4), a5, a5l, b5, letter and user can declare another own format by `\sdef`:

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

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

```
\margins/<pg> <fmt> (<left>,<right>,<top>,<bot>)<unit>
example:
\margins/1 a4 (2.5,2.5,2,2)cm
```

Parameters are:

- `<pg>` ... 1 or 2 specifies one-page or two-pages design.



- $\langle fmt \rangle$  ... paper format (a4, a4l, a5, letter, etc. or user defined).
- $\langle left \rangle$ ,  $\langle right \rangle$ ,  $\langle top \rangle$ ,  $\langle bot \rangle$  ... gives the amount of left, right, top and bottom margins.
- $\langle unit \rangle$  ... unit used for values  $\langle left \rangle$ ,  $\langle right \rangle$ ,  $\langle top \rangle$ ,  $\langle bot \rangle$ .

Each of the parameters  $\langle left \rangle$ ,  $\langle right \rangle$ ,  $\langle top \rangle$ ,  $\langle bot \rangle$  can be empty. If both  $\langle left \rangle$  and  $\langle right \rangle$  are nonempty then `\hsize` is set. Else `\hsize` is unchanged. If both  $\langle left \rangle$  and  $\langle right \rangle$  are empty then typesetting area is centered in the paper format. The analogical rule works when  $\langle top \rangle$  or  $\langle bot \rangle$  parameter is empty (`\vsize` instead `\hsize` is used). Examples:

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

If  $\langle pg \rangle=1$  then all pages have the same margins. If  $\langle pg \rangle=2$  then the declared margins are true for odd pages. The margins at the even pages are mirrored in such case, it means that  $\langle left \rangle$  is replaced by  $\langle right \rangle$  and vice versa.

The  $\langle fmt \rangle$  can be in the form  $(\langle width \rangle, \langle height \rangle) \langle unit \rangle$  where  $\langle unit \rangle$  is optional. If it is missing then  $\langle unit \rangle$  after margins specification is used. For example:

```
\margins/1 (100,200) (7,7,7,7)mm
```

declares the paper 100×200 mm with all four margins 7 mm. The spaces before and after  $\langle fmt \rangle$  parameter are necessary.

The command `\magscale[ $\langle factor \rangle$ ]` scales the whole typesetting area. The fixed point of such scaling is the so called the “Knuth’s point”: 1in below and 1in right of paper sides. Typesetting (breakpoints etc.) is unchanged. All units are relative after such scaling. Only paper formats dimensions stays 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 breaking points for paragraphs and pages are unchanged. It may be usable for review printing. After review is done, the second line can be commented out.

## 18 The last page

The number of the last page (it may be different from number of pages) is stored in the `\lastpage` register after first T<sub>E</sub>X run if the working `*.ref` file is open. This file isn’t open for every documents; only for those where the forward references are printed or table of contents is created. If the `*.ref` file isn’t open for your document and you need to use the `\lastpage` register then you have to write the command `\openref`. This command opens the `*.ref` file immediately.

There is an example for footlines in the format “current page / last page”:

```
\footline={\hss \rm \thefontsize[10]\the\pageno/\the\lastpage \hss}
```

## 19 Using other language

OPmac supports all languages but only English, Czech and Slovak languages are immediately ready for use. When using another language, you have to do little more work. First of all, use T<sub>E</sub>X engine and T<sub>E</sub>X format ready for such language (with hyphenation patterns preloaded). In order to not fall to the “encoding hell”, I recommend to use Unicode engine (`xetex` or `luatex`) and to use Unicode font family where all characters and letters needed for your language are prepared.

The `xetex` and `luatex` run with eTeX plain format (generated from `etex.src` sources) by default. This format includes hyphenation tables for all languages. For example, you can select Spanish language by `\uselanguage{espanol}`.

OPmac needs to know the translation of three auto-generated words “Chapter”, “Table” and “Figure” for selected language. And this language is marked by its ISO 639-1 code, `es` for Spanish, for example.

If you are using eTeX plain format you must set the connection between long language name and short ISO code by the macro `\isolangset{<long-name>}{<iso-code>}`. This command does nothing if `CSplain` is used, because `CSplain` sets ISO codes during format generation.

The full example with Spanish language follows. Use `xetex example` or `luatex example` for compiling this.

```
\input opmac

\ifx\directlua\undefined \else \ifx\luafonts\undefined \input luafonts
\fi\fi % lua code to re-define \font primitive, irrelevant in XeTeX

\font\tenrm="DejaVu Serif" % Unicode font family,
\font\tenbf="DejaVu Serif/B" % where all needed characters are ready
\font\tenit="DejaVu Serif/I"
\font\tenbi="DejaVu Serif/IB"
\tenrm

\input tx-math % Math setting using TX fonts

\isolangset {espanol}{es}
\sdef{mt:chap:es}{Capítulo} % Chapter in es
\sdef{mt:t:es}{Cuadro} % Table in es
\sdef{mt:f:es}{Figura} % Figure in es

\uselanguage{espanol} % Spanish hyphenation + autogenerated words
\typsize[12/14.4]

\tit Mañana

\sec Mañana

Mañana.

\caption/f Test % generates the text "Figura 1.1 Test"

\bye
```

When sorting the index by `\makeindex` in the non-`CSplain` format, OPmac writes a warning: “falling back to ASCII sorting”. If you want to use sorting rules given for your language, you must define the macro `\sortingdata<iso-code>`. And you can optionally define the `\specsortingdata<iso-code>` macro. Example:

```
\def\sortingdataes {aAäÄáÁ, bB, cCçÇ, ^P^Q^R, dD, ..., zZ, .}
\def\specsortingdataes {ch: ^P Ch: ^Q CH: ^R}
```

There are groups of letters separated by comma and ended by comma-dot in the macro `\sortingdata<iso-code>`. (In the example above, three dots must be replaced by real data by the user.) All letters in one group are not distinguished during first step of sorting (primary

sorting). If some items are equal from this point of view then the secondary sorting is processed for such items where all mentioned letters are distinguished in the order given in the macro.

Sorting algorithm can treat couple of letters (like Dz, Ch etc.) as one letter if the macro `\specsortdata<iso-code>` is defined. There is a space separated list of items in the form `<couple>:<one-token>`. The replacing from `<couple>` to `<one-token>` is done before sorting, so you can use `<one-token>` in the `\sortdata<iso-code>` macro. The `<one-token>` must be something special not used as the letter of the alphabet. The usage of `^^A`, `^^B` etc. is recommended but you must avoid the `^^I` and `^^M` because these characters have special catcode.

The macro `\sortdata<iso-code>` (if it is defined and the hyphenation of given language is selected) has precedence before an internal `\sortdata` defined in OPmac. The internal `\sortdata` are optimized for Czech, Slovak and English languages.

The list of ignored characters for sorting point of view is defined in the `\setignoredchars` macro. OPmac defines this macro like:

```
\def\setignoredchars{\setlccodes ,.;?!.:.'".|.(.)[.].<.>.=.+.{}}
```

It means that comma, semicolon, question mark, ..., plus mark are treated as dot and dot is ignored by sorting algorithm. You can redefine this macro, but you must keep the format, keep `\setlccodes` in the front and `{}` in the end.

## 20 Summary

```
\tit Title (terminated by end of line)
\chap Chapter Title (terminated by end of line)
\sec Section Title (terminated by end of line)
\secc Subsection Title (terminanted by end of line)

\maketoc           % table of contents generation
\ii item1,item2    % insertion the items to the index
\makeindex         % the index is generated

\label [labname]   % link target location
\ref [labname]     % link to the chapter, section, subsection, equation
\pgref [labname]   % link to the page of the chapter, section, ...

\caption/t         % a numbered table caption
\caption/f         % a numbered caption for the picture
\eqmark            % a numbered equation

\beginitems        % start list of the items
\enditems          % end of list of the items
\begtt            % start verbatim text
\endtt            % end verbatim text
\activettchar X    % initialization character X for in-text verbatim
\verbinput         % verbatim extract from the external file
\begmulti num      % start multicolumn text (num columns)
\endmulti          % end multicolumn text

\cite [labnames]   % refers to the item in the lits of references
\rcite [labnames]  % similar to \cite but [] are not printed.
\sortcitations \shortcitations \nonumcitations % cite format
\bib [labname]     % an item in the list of references
\usebibtex {bib-base}{bst-style} % use BibTeX for bibliography
```

```

\genbbl {bib-base}{bst-style}      % prepare the bbl file generation
\usebbl/? bbl-base    % use pre-generated bbl file, ? in {a,b,c}
\usebib/? (style) bib-base % direct using of .bib file, ? in {s,c}

\fontfam [FamilyName] % selection of font family
\typosize [font-size/baselineskip] % size setting of typesetting
\typoscale [factor-font/factor-baselineskip] % size scaling
\thefontsize [size] \thefontscale [factor]    % current font size

\inspic file.ext      % insert a picture, extensions: jpg, png, pdf
\table {rule}{data} % simple macro for the tables like in LaTeX

\fnote      % footnote (local numbering on each page)
\mnote      % note in the margin (left or right by page number)

\hyperlinks {color-in}{color-out} % PDF links activate as clickable
\outlines {level}    % PDF will have a table of contents in the left tab

\magscale[factor] % resize typesetting, line/page breaking unchanged
\margins/pg format (left, right, top, bottom)unit % margins setting

```