OpTEX

Format Based on Plain TEX and OPmac[1]

Version 0.07

Petr Olšák, 2020

http://petr.olsak.net/optex

Contents

---

[1] The OPmac package is a set of simple additional macros to Plain TEX. It enables users to take advantage of basic LATEX functionality but keeps Plain TEX simplicity. See http://petr.olsak.net/opmac-e.html for more information about it. For OPmac users: the red triangle ◀ in the right margin means that there is a difference from standard OPmac features.

Introduction

OpTEX is LuaTEX format with Plain TEX and OPmac. Only LuaTEX engine is supported. The main goal of OpTEX is:

- OpTEX keeps the simplicity (like in Plain TEX and OPmac macros).
- There is no old obscurities concerning with various 8-bit encodings and various engines.
- OpTEX provides a powerful font selection system (for Unicode font families, of course).
- OpTEX supports hyphenations of all languages installed in your TEX system.
- All features from OPmac macros are copied.
- Macros are documented in the same place where code is (macros for printing such documentation will come in the future).
- User name space of control sequences is separated from internal name space of OpTEX and primitives (`\foo` versus `\_foo`).

OpTEX should be a modern Plain TEX with power from OPmac (fonts selection system, colors, external graphics, references, hyperlinks, indexing, bibliography, ...) with preferred Unicode fonts.

If you need to customize your document or you need to use something very specific, then you can copy relevant parts of OpTEX macros into your macro file and do changes of these macros here. This is significant difference from LaTEX or ConTeXt, which are an attempt to create a new user level with a plenty of non-primitive parameters and syntax hiding TEX internals. The macros from OpTEX are simple and straightforward because they solve only what is explicitly needed, they does not create a new user level for controlling your document over TEX. And you can use OpTEX macros, understand them an modify them.

OpTEX offers a markup language for authors of texts (like LaTEX), i.e. the fixed set of tags to define the structure of the document. This markup is different from the LaTEX markup. It may offer to write the source text of the document somewhat clearer and more attractive.

Disclaimer: This software is under construction. It is possible that some features documented ◀ here will be changed in future.

This manual desribes OpTEX features only. We suppose that user knows TEX basic principles. They are described in many books. You can see a short document TEX in nutshell too.

1 Starting with OpTEX

OpTEX is compiled as a format for LuaTEX. Maybe there is a command `optex` in your TEX distribu- ◀ tion. Then you can write into command line

```
optex document
```

You can try to process `optex op-demo` or `optex optex-doc`.

If there is no `optex` command, see more information about installation OpTEX at http://petr. olsak.net/optex.

A minimal document should be

```
\fontfam[LMfonts]
Hello World! \bye
```

The first line `\fontfam[LMfonts]` tells that Unicode Latin Modern fonts (derived from Computer Modern) are used. If you omit this line then preloaded Latin Modern fonts are used but preloaded fonts cannot be in Unicode[2]. So the sentence `Hello World` will be OK without the first line, but you cannot print such sentence in another languages (for example `Ahoj světe!`) where Unicode fonts are needed because of the characters like ě are not mapped correctly in preloaded fonts.

A somewhat larger example with common settings should be:

```
\fontfam[Termes]    % selecting Unicode font family Termes (section 3.1)
\typosize[11/13]    % setting the basic font size and the baselineskip (sec. 3.2)
\margins/1 a4 (1,1,1,1)in  % setting 1in margins for A4 paper (section 2.1)
\cslang             % Czech hyphenation patterns (section 7.1)
```

---

[2] This is a technical limitation of LuaTEX for fonts downloaded in formats: only 8bit fonts can be preloaded.

```
   Tady je text.
   \bye
```

You can look at `op-demo.tex` file for more examples.

2 Page layout

2.1 Setting the margins

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

Each 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 analogical rule works when ⟨top⟩ or ⟨bot⟩
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 ⟨pg⟩=1 then all pages have the same margins. If ⟨pg⟩=2 then the declared margins are true for
odd pages. The margins at the even pages are automaticaly mirrored in such case, it means that
⟨left⟩ is replaced by ⟨right⟩ and vice versa.

The ⟨fmt⟩ can be also in the form (⟨width⟩,⟨height⟩)⟨unit⟩ where ⟨unit⟩ is optional. If it is
missing then ⟨unit⟩ 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 ⟨fmt⟩
parameter are necessary.

The command `\magscale[⟨factor⟩]` scales the whole typesetting area. The fixed point of such ◀
scaling is the upper left corner of the paper sheet. 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.

2.2 Concept of default page

OpTeX uses for page design very similar "output routine" like Plain TeX. There is `\headline` followed by "page body" followed by `\footline`. The `\headline` is empty by default and it can be used for running headers repeated on each page. The `\footline` prints centered page number by default. You can set the `\footline` to empty using `\nopagenumbers` macro.

The margins declared by `\margins` macro (documented in the previous section 2.1) is concerned to the page body, i.e. the `\headline` and `\footline` are placed to the top and bottom margins.

The distance between the `\headline` and the top of the page body is given by `\headlinedist` register. The distance between bottom of the page body and the `\footline` is given by `\footlinedist`. The default values are:

```
\headline = {}
\footline = {\_hss\_rmfixed \_folio \_hss} % \folio expands to page number
\headlinedist = 14pt % from baseline of \headline to top of page body
\footlinedist = 24pt % from last line in pagebody to baseline of footline
```

The page body should be divided to top insertions (floating tables and figures) followed by a real text and followed by footnotes. Typically, only real text is here.

The `\pgbackground` tokens list is empty by default but it can be used for creating background of each page (colors, picture, watermark for example). The macro `\draft` uses this register and puts big text DRAFT as watermark to each page. You can try it.

More about the page layout is documented in files `parameters.opm` and `output.opm`.

2.3 Footnotes and marginal notes

The Plain TeX's macro `\footnote` can be used as usual. But a new macro `\fnote{⟨text⟩}` is defined. The footnote mark is added automatically and it is numbered on each chapter from one[3]. The ⟨text⟩ ◀ is scaled to 80 %. User can redefine footnote mark or scaling, as shown in the file `fnotes.opm`.

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⟨numeric-label⟩` inside the box: only the footnote mark is generated here. When the box is finished you can use `\fnotetext{⟨text⟩}`. This macro puts the ⟨text⟩ to the footnote. The ⟨numeric-label⟩ 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 `\fnotetext`s have to be written after the box as the number of `\fnotemark`s inserted inside the box. Example:

```
Text in a paragraph\fnote{First notice}...    % a "normal" footnote
\table{...}{...\fnotemark1...\fnotemark2...}  % two footnotes in a box
\fnotetext{Second notice}
\fnotetext{Third notice}
...
\table{...}{...\fnotemark1...}                % one footnote in a box
\fnotetext{Fourth notice}
```

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 and read from it again. If you need to place the notes only to the fixed margin write `\fixmnotes\right` or `\fixmnotes\left`.

The ⟨text⟩ 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 has its vertical position given by the position of `\mnote` in the text. The exceptions are possible by setting the `\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{⟨text⟩}` shifts this (and only this) note two lines up.

---

[3] You can declare `\fnotenumglobal` if you want footnotes numbered in whole document from one or `\fnotenumpages` if you want footnotes numbered at each page from one. Default setting is `\fnotenumchapters`

3 Fonts

3.1 Font families

You can select the font family by `\fontfam[`⟨*Family-name*⟩`]`. The argument ⟨*Family-name*⟩ is case insensitive and spaces are ignored. So, `\fontfam[LM Fonts]` is equal to `\fontfam[LMfonts]` and it is equal to `\fontfam[lmfonts]`. Several aliases are prepared, thus `\fontfam[Latin Modern]` can be used for loading Latin Modern family too.

If you write `\fontfam[?]` then all font families registered in OpTEX are listed on the terminal and in the log file. If you write `\fontfam[catalog]` then a catalog of all fonts registered in OpTEX and available in your TEX system is printed. The instructions how to register your own font family is appended in such catalog.

If the family is loaded then *font modifiers* applicable in such font family are listed on the terminal: (`\caps`, `\cond` for example). And there are four basic *variant selectors* (`\rm`, `\bf`, `\it`, `\bi`). The usage of variant selectors is the same as in Plain TEX: `{\it italics text}`, `{\bf bold text}` etc.

The font modifiers (`\caps`, `\cond` for example) can be used before a basic variant selector and they can be (independently) combined: `\caps\it` or `\cond\caps\bf`. The modifiers keeps their internal setting until group ends or until another modifier which negates the previous feature is used. So `\caps \rm text \it text` uses normal and italics in Caps and SmallCaps.

There is one special variant selector `\currvar` which does not change the selected variant but ◀ reloads the font due to the (maybe newly specified) font modifier(s).

The context between variants `\rm`–`\it` and `\bf`–`\bi` is kept by the `\em` macro (emphasize text). It switches from current `\rm` to `\it`, from current `\it` to `\rm`, from current `\bf` to `\bi` and from current `\bi` to `\bf`. The italics correction `\/` is 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.
```

More about the OpTEX font selection system is written the file `fonts-select.opm`. You can mix ◀ more font families in your document, you can declare your own variant selectors or modifiers etc.

3.2 Font sizes

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. Don't write the unit of these parameters. The unit is internally set to `\ptunit` which is 1pt by default. You can change the unit by the command `\ptunit=`⟨*something-else*⟩, for instance `\ptunit=1mm` enlarges all font sizes declared by `\typosize`. Examples:

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

The commands for font size setting described in this section have local validity. If you put them into a group, the settings are lost when the group is finished. If you set something relevant with paragraph shape (baselineskip given by `\typosize` for example) then you must first finalize the paragraph and second to close the group: `{\typosize[12/14] ...`⟨*text of paragraph*⟩`... \par}`.

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 (\magstep2 expands to 1440)
```

First usage of `\typosize` or `\typoscale` macro in your document sets so called *main values*, i.e. main font size and main baselineskip. They are internally saved in registers `\mainfosize` and `\mainbaselineskip`.

The `\typoscale` command does scaling in respect to current values by default. If you want to ◀
do it in respect to main values, type `\scalemain` immediately before `\typoscale` command.

```
\typosize[12/14.4] % first usage in document, sets main values internally
\typosize[15/18]   % bigger font
\scalemain \typoscale[800/800] % reduces from main values, no from current.
```

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.

There is "low level" `\setfontsize{⟨size-spec⟩}` command which behaves like a font modifier and ◀
sets given font size used by next variant selectors. For example `\setfontsize{at15pt}\currvar` sets current variant to 15pt.

If you are using a font family with "optical sizes feature" (i. e. there are more recommended sizes of the same font which are not scaled linearly; good example is Computer Modern aka Latin Modern fonts) then the recommended size is selected by all mentioned commands automatically.

More information about resizing of fonts is documented in `fonts-resize.opm` file.

3.3 Typesetting math

OpTEX preloads a collection of 7bit Computer Modern math fonts and AMS fonts in its format.
You can use them in any size and in the `\boldmath` variant. Most declared text font families (see ◀
`\fontfam` in the section 3.1) are configured with recommended Unicode math font. This font is automaticlaly loaded unless you specify `\noloadmath` before first `\fontfam` command. See log file for more information about loading text font family and Unicode math fonts. If you prefer another Unicode math font, specify it by `\loadmath{[⟨font-file⟩]}` or `\loadmath{font-name}` before first `\loadfam` command.

Hundreds math symbols and operators like in AMSTEX are accessible. For example `\alpha` α, `\geq` $\geq$, `\sum` $\sum$, `\sphericalangle` $\sphericalangle$, `\bumpeq`, $\bumpeq$. See AMSTEX manual for complete list of symbols.

The following math alphabets are available:

```
\mit      % mathematical variables    abc−xyz, ABC−XYZ
\it       % text italics              abc−xyz, ABC−XYZ
\rm       % text roman                abc−xyz, ABC−XYZ
\cal      % normal calligraphics      ABC−XYZ
\script   % script                    ABC−XYZ
\frak     % fracture                  abc−xyz, ABC−XYZ
\bbchar   % double stroked letters    ABC−XYZ
\bf       % sans serif bold           abc−xyz, ABC−XYZ
\bi       % sans serif bold slanted   abc−xyz, ABC−XYZ
```

The last two selectors `\bf` and `\bi` select the sans serif fonts in math regardless the current text font family. This is common notation for vectors and matrices. You can re-declare it, see the file `unimath-codes.opm` where math variants of `\bf` and `\bi` selectors are defined.

The math fonts can be scaled by `\typosize` and `\typoscale` macros. Two math fonts collections are prepared: `\normalmath` for normal weight and `\boldmath` for bold. The first one is set by default.

You can use `\mathbox{⟨text⟩}` inside math mode. It behaves as `{\hbox{⟨text⟩}}` (i.e. the ⟨text⟩ ◀
is printed in horizontal non-math mode) but the size of the ⟨text⟩ is adapted to the context of math size (text or script or scriptscript).

4 Typical elements of document

4.1 Chapters and sections

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⟩
```

If you want to write a title to more lines in your source file then you can use percent character before ⟨*end of line*⟩. Such ⟨*end of line*⟩ is not scanned and reading of the title continues at next line.

The chapters are automatically 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`. A designer 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.

4.2 Another numbered objects

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

If user writes the `\eqmark` as the last element of the display mode then this equation is numbered. The equation number is printed in brackets. This number resets in each section by default.

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 automatically numbered objects are captions which is tagged by `\caption/t` for tables and `\caption/f` for figures. The caption text follows. The `\cskip` can be used between `\caption` text and the real object (table or figure). You can use two orders: ⟨*caption*⟩`\cskip` ⟨*object*⟩ or ⟨*object*⟩`\cskip` ⟨*caption*⟩. The `\cskip` creates appropriate vertical space between them. Example:

```
\caption/t The dependency of the computer-dependency on the age.
\cskip
\hfil\table{rl}{
    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}
```

This example produces:

Table 4.1 The dependency of the computer-dependency on the age.

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

You can see that the word "Table" followed by a number is added by the macro `\caption/t`. The caption text is centered. If it occupies more lines then the last line is centered.

The macro `\caption/f` behaves like `\caption/t` but it is intended for figure captions with independent numbering. The word (Table, Figure) depends on the actual selected language (see section 7.1 about languages).

If you wish to make the table or figure as floating object, you need to use Plain TeX macros `\midinsert` or `\topinsert` terminated by `\endinsert`. Example:

```
\topinsert  % table and its caption is printed at the top of the current page
    ⟨caption and table⟩
\endinsert
```

4.3 References

Each automatically numbered object documented in sections 4.1 and 4.2 can be referenced if optional ◀ parameter [⟨*label*⟩] is appended to `\chap`, `\sec`, `\secc`, `\caption/t`, `\caption/f` or `\eqmark`. The alternative syntax is to use `\label[⟨*label*⟩]` before mentioned commands (not necessarily directly before). The reference is realized by `\ref[⟨*label*⟩]` or `\pgref[⟨*label*⟩]`. Example:

```
\sec[beatle] About Beatles

\hfil\table{rl}{...} % the table
\cskip
\caption/t [comp-depend] 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 something about the equation~\ref[pythagoras]. Finally there
is an interesting Table~\ref[comp-depend].
```

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

By default, labels are not printed, of course. But if you are preparing a draft version of document, you can declare `\showlabels`. The labels are printed at their destination places after such declaration.

4.4 Hyperlinks, outlines

If the command `\hyperlinks` ⟨*color-in*⟩ ⟨*color-out*⟩ is used at the beginning of the document, then the following objects are hyperlinked in the PDF output:

- 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` or `\ulink` commands.

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 `\ilink[⟨type⟩:⟨label⟩]{⟨text⟩}`. The ⟨type⟩ parameter is one of the `toc`, `pg`, `cite`, `ref` or another special for your purpose. These commands create internal links only when `\hyperlinks` is decared.

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[4]. 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/optex]{\OpTeX/ page}` creates OpTeX page.

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.

If you are using a special macro in section titles then `\outlines` macro may crash. You must ◀ declare variant of the macro for outlines case which is expandable. Use `\regmacro` in such case. See the section 5.1 for more information about `\regmacro`.

The command `\insertoutline{⟨text⟩}` inserts next entry into PDF outlines at the main level 0. These entries can be placed before table of contents (created by `\outlines`) or after it. Theirs hyperlink destination is in the place where `\insertoutline` macro is used.

4.5 Lists

The 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 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
```

For example:

```
\begitems
* First idea
* Second idea in subitems:
  \begitems \style i
  * First sub-idea
  * Second sub-idea
  * Last sub-idea
  \enditems
* Finito
\enditems
```

produces:

- First idea
- Second idea in subitems:

---

[4] More exactly, there are the same rules as for `\code` command, see section 4.7.

    (i) First sub-idea

   (ii) Second sub-idea

  (iii) Last sub-idea

- Finito

Another style can be defined by the command `\sdef{_item:⟨style⟩}{⟨text⟩}`. Default item can be set by `\defaultitem={⟨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. More information about design of lists of items are in the `lists.opm` file.

4.6 Tables

The macro `\table{⟨declaration⟩}{⟨data⟩}` provides similar ⟨declaration⟩ of tables 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{||lc|r||}{                      \crl
   Month     & commodity  & price    \crli \tskip2pt
   January   & notebook   & \$ 700   \cr
   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 problems with narrow left-right aligned paragraphs you can write `p{⟨size⟩\raggedright}`, then the paragraph will be only left aligned.

You can use (⟨text⟩) in the ⟨declaration⟩. Then this text is applied in each line of the table. For example `r(\kern10pt)l` adds more 10 pt space between `r` and `l` rows.

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 is generally known. It marks end row of the table. Moreover OpTeX 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.

You can use following parameters for the `\table` macro. Default values are listed too.

```
\everytable={}       % code used after settings in \vbox before table processing
\thistable={}        % code used when \vbox starts, is is removed after using it
\tabiteml={\enspace} % left material in each column
\tabitemr={\enspace} % right material in each column
\tabstrut={\strut}   % strut which declares lines distance in the table
\tablinespace=2pt    % additional vertical space before/after horizontal lines
\vvkern=1pt          % space between lines in double vertical line
\hhkern=1pt          % space between lines in double horizontal line
```

Example: if you do `\tabiteml={$\enspace}\tabitemr={\enspace$}` then the `\table` acts like LaTeX's array environment.

If there is an item which spans to more than one column in the table then `\multispan{⟨number⟩}` macro from Plain TeX can help you or, you can use `\mspan⟨number⟩[⟨declaration⟩]{⟨text⟩}` which spans ⟨number⟩ columns and formats the ⟨text⟩ by the ⟨declaration⟩. The ⟨declaration⟩ must include a declaration of only one column with the same syntax as common `\table` ⟨declaration⟩. If your table includes vertical rules and you want to create continuous vertical rules by `\mspan`, then use rule declarators `|` 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}\crli
  first & second & third  \crlli
  seven & eight  & nine   \crli}}
```

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⟩}`. More about it is in technical documentation in the file `table.opm`.

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

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

4.7 Verbatim

The display verbatim text have to be surrounded by the `\begtt` and `\endtt` couple. The in-line verbatim have to be tagged (before and after) by a character which is declared by `\activettchar⟨char⟩`. For example `\activettchar`` declares the character `` ` `` for in-line verbatim markup. And you can use `` `\relax` `` for verbatim `\relax` (for example). Another alternative of printing in-line verbatim text is ◀ `\code{⟨text⟩}` (see below).

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. OpTeX 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` by default. User can change values of the `\parindent` and `\ttindent` independently.

The `\begtt` command starts internal group in which the catcodes are changed. Then the ◀ `\everytt` tokens register is run. It is empty by default and user can control fine behavior by it. For example the catcodes can be reseted here. If you need to define active character in the `\everytt`, use `\adef` as in the following example:

```
\everytt={\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 value of `\everytt` is read. So you don't have to worry about active categories.

There is an alternative to `\everytt` named `\everyintt` which is used for in-line verbatim surrounded by an `\activettchar` or processed by the `\code` command.

The `\everytt` is applied to all `\begtt...\endtt` environments (if it is not decared in a group). There are tips for such global `\everytt` definitions here:

```
\everytt={\typosize[9/11]}      % setting font size for verbatim
\everytt={\ttline=0}            % each listing will be numbered from one
\everytt={\adef{ }{\char9251 }} % visualization of spaces (Unicode fonts)
```

If you want to apply a special code only for one `\begtt...\endtt` environment then don't set ◀ any `\everytt` but put desired material at the same line where `\begtt` is. For example:

```
\begtt   \adef!{?}\adef?{!}
Each occurrence of ? will be changed to ! and vice versa.
\endtt
```

The in-line verbatim surrounded by an `\activettchar` doesn't work in parameter of macros and macro definitions, especially in titles declared by `\chap`, `\sec` etc. You can use more robust command ◀ `\code{⟨text⟩}` in such situations, but you must escape following characters in the ⟨text⟩: `\`, `#`, `%`, braces (if the braces are unmatched in the ⟨text⟩), and space or `^` (if there are more than one subsequent spaces or `^` in the ⟨text⟩). Examples:

```
\code{\\text, \%\#} ... prints \text, %#
\code{@{..}*&^$ $}  ... prints @{..}*&^$ $ without escaping, but you can
                        escape these characters too, if you want.
\code{a \ b}        ... two spaces between a  b, the second one must be escaped
\code{xy\{z}        ... xy{z ... unbalanced brace must be escaped
\code{^\^M}         ... prints ^^M, the second ^ must be escaped
```

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

```
\verbinput (12-42) program.c  % listing from program.c, only lines 12-42
\verbinput (-60) program.c    % print from begin to the line 60
\verbinput (61-) program.c    % from line 61 to the end
\verbinput (-) program.c      % whole file is printed
\verbinput (70+10) program.c  % from line 70, only 10 lines printed
\verbinput (+10) program.c    % from the last line read, print 10 lines
\verbinput (-5+7) program.c   % from the last line read, skip 5, print 7
\verbinput (+) 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 `\verbinput` can be controlled by `\everytt`, `\ttindent` just like in `\begtt...\endtt`.

5   Autogenerated lists

5.1   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 TeX 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
```

If you are using a special macro in section titles or chapter titles and you need different behavior ◀ of such macro in other cases then use `\regmacro{⟨case-toc⟩}{⟨case-mark⟩}{⟨case-outline⟩}`. The parameters are applied locally in given cases. The `\regmacro` can be used repeatedly: the parameters

are accumulated (for more macros). If a parameter is empty then original definition is used in given case. For example:

```
% default value of \mylogo macro used in text and in the titles:
\def\mylogo{\leavevmode\hbox{\Red{\it My}\Black{\setfontsize{mag1.5}\rm Lo}Go}}
% another variants:
\regmacro {\def\mylogo{\hbox{\Red My\Black LoGo}}} % used in TOC
         {\def\mylogo{\hbox{{\it My}\/LoGo}}}      % used in running heads
         {\def\mylogo{MyLoGo}}                     % used in outlines
```

5.2   Making the index

The index can be included into document by `\makeindex` macro. No external program is needed, the alphabetical sorting are done inside TeX 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 index entries 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 `~`.

If the `\ii` or `\iid` command is preceded by `\iitype` ⟨*letter*⟩, then such reference (or more references generated by one `\ii`) has specified type. The page numbers of such references should be formatted specially in the index. OpTeX implements only `\iitype b`, `\iitype i` and `\iitype u`. The page number in bold or in italics or underlined is printed in the index when these types are used. Default index type is empty, which prints page numbers in normal font. See TeXbook index as an example.

The `\makeindex` creates the list of alphabetically sorted index entries without the title of the section and without creating more columns. OpTeX 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
\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, TeX 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 {\code{\\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 sorting rules when `\makeindex` runs depends on the current language. See section 7.1 about languages selection.

5.3   BibTeXing

The command `\cite[`⟨*label*⟩`]` (or its variants of the type `\cite[`⟨*label-1*⟩`,`⟨*label-2*⟩`,...,`⟨*label-n*⟩`]`) creates the citation 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 declared then the marks instead numbers are generated depending on the used bib-style. For example the citations look like [Now08] or [Nowak, 2008].

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

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

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

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

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

- Manually using `\bib[`⟨*label*⟩`]` commands.
- By `\usebib/`⟨*type*⟩ (⟨*style*⟩) ⟨*bib-base*⟩ command which reads `*.bib` files directly.

Note that another two possibilities documented in OPmac (using external BibTeX program) isn't ◀ supported because BibTeX is old program which does not support Unicode. And Biber seems to be not compliant with Plain TeX.

References created manually using `\bib[`⟨*label*⟩`]` 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 ⟨*marks*⟩ used by `\cite` command. To do it you must use long form of the `\bib` command in the format `\bib[`⟨*label*⟩`] = {`⟨*mark*⟩`}`. The spaces around equal sign are mandatory. Example:

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

Direct reading of `.bib` files is possible by `\usebib` macro. This macro reads and uses macro package `librarian.tex` by Paul Isambert. The usage is:

```
\usebib/c (⟨style⟩) ⟨bib-base⟩ % sorted by \cite-order (c=cite),
\usebib/s (⟨style⟩) ⟨bib-base⟩ % sorted by style (s=style).
% example:
\usebib/s (simple) op-example
```

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 `bib-`⟨*style*⟩`.opm` where the formatting of the references list is defined. OpTₑX supports `simple` or `iso690` styles. The behavior of `\usebib` is documented in `usebib.opm` and `bib-iso690.opm` files in detail.

Not all records are printed from ⟨*bib-base*⟩ files: the command `\usebib` selects only such bib-records which were used in `\cite` or `\nocite` commands in your document. The `\nocite` behaves as `\cite` but prints nothing. It only tels that mentioned bib-record should be printed in the reference list. If `\notcite[*]` is used then all records from ⟨*bib-base*⟩ are printed.

6 Graphics

6.1 Colors

OpTₑX 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 four CMYK components or three RGB components. For example ◀

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

The command `\morecolors` reads more definitions of color selectors from LaTeX file `x11nam.def`. ◀ There is about 300 color names like `\DeepPink`, `\Chocolate` etc. If there are numbered variants of the same name, then you can append letters B, C, etc. to the name in OpTₑX. For example `\Chocolate` is Chocolate1, `\ChocolateB` is Chocolate2 etc.

The color selectors work locally in groups by default but with limitations. See the file `colors.opm` ◀ for more information.

The colors `\Blue`, `\Cyan` etc. are defined with CMYK components using `\setcmykcolor`. But you can define a color with three RGB components too and `\morecolors` defines such RGB colors. By default, the color model isn't converted but only stored to PDF output for each used color by default. Thus, there may be a mix of color models in the PDF output which is not good idea. You can overcome this problem by declaration `\onlyrgb` or `\onlycmyk`. Then only selected color model is used for PDF output and if an used color is declared by another color model then it is converted. The `\onlyrgb` creates colors more bright (usable for computer presentations). On the other hand CMYK makes colors more true[5] for printing.

You can define your color by a linear combination of previously defined colors using `\colordef`. ◀ For example:

```
\colordef \myCyan {.3\Green + .5\Blue}  % 30 % green, 50 % blue, rest is white
\colordef \DarkBlue {\Blue + .4\Black}  % Blue mixed with 40 % of black
\colordef \myGreen{\Cyan+\Yellow}       % exact the same as \Green
\colordef \MyColor {.3\Orange+.5\Green+.2\Yellow}
```

The linear combination is done in CMYK substractive color space by default (RGB colors used in `\colordef` argument are converted first). If the resulting component is greater than 1 then it is truncated to 1. If a convex linear combination (as in the last example above) is used then it emulates color behavior on a painter's palette. You can use `\rgbcolordef` instead `\colordef` if you want to mix colors in additive RGB color space.

More about colors, about CMYK versus RGB and about the `\colordef` command is written in the `colors.opm` file.

The following example defines a macro which creates the <mark>colored text on the colored background</mark>. Usage: `\coloron`⟨*background*⟩⟨*foreground*⟩`{`⟨*text*⟩`}`

The `\coloron` can be defined as follows:

---

[5] Printed output is more equal to the monitor preview especialy if you are using ICC profile for your printer.

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

6.2   Images

The `\inspic` `{⟨filename⟩.⟨extension⟩}` or `\inspic` `⟨filename⟩.⟨extension⟩⟨space⟩` inserts the picture
stored in the graphics file with the name ⟨*filename*⟩.⟨*extension*⟩ to the document. You can set the
picture width by `\picw=⟨dimen⟩` before `\inspic` command which declares the width of the picture
The image files can be in the PNG, JPG, JBIG2 or PDF format.

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 image files are searched in `\picdir`. This token list is empty by default, this means that
the image files are searched in the current directory. Example: `\picdir={img/}` supposes that image
files are in `img` subdirectory. Note: the directory name must end by `/` in `\picdir` declaration.

Inkscape[6] is able to save a picture to PDF and labels of the picture to another file[7]. This second
file should be read by TeX in order to print labels in the same font as document font. OpTeX supports
this feature by `\inkinspic` `{⟨filename⟩.pdf}` command. It reads and displays both: PDF image and
labes generated by Inkscape.

If you want to create a vector graphics (diagrams, schema, geometry skicing) then you can do it
in Wysiwyg graphics editor (Inkscape, Geogebra for example), export the result to PDF and include
it by `\inspic`. If you want to "programm" such pictures then Tikz package is recommended. It works
in Plain TeX.

6.3   PDF transformations

All typesetting elements are transformed 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. One linear transformation given by
the `\pdfsetmatrix` above transforms the vector $[0, 1]$ to $[⟨a⟩, ⟨b⟩]$ and $[1, 0]$ to $[⟨c⟩, ⟨d⟩]$. The stack-
oriented commands `\pdfsave` and `\pdfrestore` gives a possibility of storing and restoring the current
transformation matrix and the current point. The position of the 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.

OpTeX provides two special transformation macros:

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

These macros simply calls the properly `\pdfsetmatrix` 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:

_____

[6] A powerfull and free wysiwyg editor for creating vector graphics.
[7] Chose "Omit text in PDF and create LaTeX file" option.

You can see that TeX knows nothing about dimensions of transfomed material, it treats it as with a zero dimension object. This problem is solved by the `\transformbox{⟨transformation⟩}{⟨text⟩}` macro which puts the transformed material to a box with relevant dimension. The ⟨transfromation⟩ parameter includes one or more transfromation commands `\pdfsetmatrix`, `\pdfscale`, `\pdfrotate` with their parameters. The ⟨text⟩ is transformed text.

Example: `\frame{\transformbox{\pdfscale{1}{1.5}\pdfrotate{-10}}{moj}}` creates ⧄moj⧄.

The `\rotbox{⟨deg⟩}{⟨text⟩}` is a shortcut for `\transformbox{\pdfrotate{⟨deg⟩}}{⟨text⟩}`.

7 Others

7.1 Using more languages

OpTeX prepares hyphenation patterns for all languages if such patterns are available in your TeX system. Only USenglish patterns (original from Plain TeX) are preloaded. Hyphenation patterns of all another languages are loaded on demand when you first use the `\⟨iso-code⟩lang` command in your document. For example `\delang` for German, `\cslang` for Czech, `\pllang` for Polish. The ⟨iso-code⟩ is a shortcut of the language (mostly from ISO 639-1). You can list all available languages by `\langlist` macro. This macro prints now:

en(USenglish) enus(USenglishmax) engb(UKenglish) it(Italian) ia(Interlingua) id(Indonesian) cs(Czech) sk(Slovak) de(nGerman) fr(French) pl(Polish) cy(Welsh) da(Danish) es(Spanish) sl(Slovenian) fi(Finnish) hy(Hungarian) tr(Turkish) et(Estoniak) eu(Basque) ga(Irish) nb(Bokmal) nn(Nynorsk) nl(Dutch) pt(Portuguese) ro(Romanian) hr(Croatian) zh(Pinyin) is(Icelandic) hsb(Uppersorbian) af(Afrikaans) gl(Galician) kmr(Kurmanji) tk(Turkmen) la(Latin) lac(classicLatin) lal(liturgicalLatin) elm(monoGreek) elp(Greek) grc(ancientGreek) ca(Catalan) cop(Coptic) mn(Mongolian) sa(Sanskrit) ru(Russian) uk(Ukrainian) hy(Armenian) as(Assamese) hi(Hindi) kn(Kannada) lv(Latvian) lt(Lithuanian) ml(Malayalam) mr(Marathi) or(Oriya) pa(Panjabi) ta(Tamil) te(Telugu)

For compatibility with e-plain macros, there is the command `\uselanguege{⟨language⟩}`. The parameter ⟨language⟩ is long form of language name, i.e. `\uselanguage{Czech}` works the same as `\cslang`. The `\uselanguage` parameter is case insensitive.

For compatibility with $\mathcal{C_S}$plain, there are macros `\ehyph`, `\chyph`, `\shyph` which are equivalent to `\enlang`, `\cslang` and `\sklang`.

You can switch between language patterns by `\⟨iso-code⟩lang` commands mentioned above. Default is `\enlang`.

OpTeX generates three words used for captions and titles in technical articles or books: "Chapter", "Table" and "Figure". These words need to be known in used language and it depends on the previously used language selectors `\⟨iso-code⟩lang`. OpTeX declares these words only for few languages: Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English, If you need to use these words in another languages or you want to auto-generate more words in your macros, then you can declare it by `\sdef` or `\_langw` commands as shown in the file `languages.opm`.

The `\makeindex` command needs to know the sorting rules used in your language. OpTeX defines only few language rules for sorting: Czech, Slovak and English. How to declare sorting rules for more languages are described in the file `makeindex.opm`.

If you declare `\⟨iso-code⟩quotes`, then the control sequences `\"` and `\'` should be used like this: `\"⟨quoted text⟩"` or `\'⟨quoted text⟩'` (note that the terminating character is the same but it isn't escaped). This prints language dependent normal or alternative quotes around ⟨quoted text⟩. The language is specified by ⟨iso-code⟩. OpTeX declares quotes only for Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English (`\csquotes`, `\dequotes`, ..., `\enquotes`). You can simply define your own quotes as shown in `languages.opm` file. The `\"` is used for quotes visualy more similar to the `"` character which can be primary quotes or secondary quotes depending on the language rules. May be you want to alternate meaning of these two types of quotes. Use `\⟨isocode⟩quotes\atlquotes` in such case.

7.2 Pre-defined styles

OpTeX defines three style-declaration macros `\report`, `\letter` and `\slides`. You can use them at the beginning of your document if you are preparing these types of document and you don't need to create your own macros.

The `\report` declaration is intended to create reports. It sets default font size to 11 pt and `\parindent` (paragraph indentation) to 1.2 em. The `\tit` macro uses smaller font because we assume that "chapter level" will be not used in reports. The first page has no page number, but next pages are numbered (from number 2). Footnotes are numbered from one in whole document. The macro `\author` ⟨*authors*⟩⟨*end-line*⟩ can be used when `\report` is declared. It prints ⟨*authors*⟩ in italics at center of the line. You can separate authors by `\nl` to more lines.

The `\letter` declaration is intended to create letters. See an example in the file `op-letter.tex`. The `\letter` style sets default font size to 11 pt and `\parindent` to 0 pt. It sets half-line space between paragraphs. The page numbers are not printed. The `\subject` macro can be used, it prints the word "Subject:" or "Věc" (or something else depending on current language) in bold. Moreover, the `\address` macro can be used when `\letter` is declared. The usage of the `\address` macro looks like:

> `\address`
> ⟨*first line of address*⟩
> ⟨*second line of address*⟩
> ⟨*etc.*⟩
> ⟨*empty line*⟩

It means that you need not to use any special mark at the end of lines: end of lines in the source file are the same as in printed output. The `\address` macro creates `\vtop` with address lines. The width of such `\vtop` is equal to the most wide line used in it. So, you can use `\hfill\address...` in order to put the address box to the right side of the document. Or you can use ⟨*prefixed text*⟩`\address...` to put ⟨*prefixed text*⟩ before first line of the address.

The `\slides` style ceates a simple presentation slides. See an example in the file `op-slides.tex`. Run `optex op-slides.tex` and see the usage of `\slides` style in the file `op-slides.pdf`.

Analogical declaration macros `\book` is not prepared. Each book needs an individual typographical care so you need to create specific macros for design.

7.3 Lorem ipsum dolor sit

A designer needs to concentrate to the design of the output and maybe he/she needs a material ◀ for testing macros. There is the possibility to generate a neutral text for such experiments. Use `\lorem[`⟨*number*⟩`]` or `\lorem[`⟨*from*⟩`-`⟨*to*⟩`]`. It prints a paragraph (or paragraphs) with neutral text. The numbers ⟨*number*⟩ or ⟨*from*⟩, ⟨*to*⟩ must be in the range 1 to 150 because there are 150 paragraphs with neutral text prepared for you. The `\lipsum` macro is equivalent to `\lorem`. Example `\lipsum[1-150]` prints all prepared paragraphs.

7.4 Logos

The control sequences for typical logos can be terminated by optional `/` which is ignored when printing. ◀ This makes logos more legible in source file:

> `We are using \TeX/ because it is cool. \OpTeX/ is better than \LaTeX.`

7.5 The last page

The number of the last page (it may be different from number of pages) is expanded by `\lastpage` macro. It expands to `?` in first TeX run and to the last page in next TeX runs.

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

> `\footline={\hss \fixedrm \folio/\lastpage \hss}`

The `\lastpage` expands to the last `\folio` which is a decimal number or roman nummeral (when ◀ `\pageno` is negative). If you need to know total pages used in the document, use `\totalpages` macro. It expands to zero (in first TeX run) or to the number of all pages in the document (in next TeX runs).

7.6 Use OpTeX

The command `\useOpTeX` (or `\useoptex`) does nothing in OpTeX but it causes an error (undefined ◀ control sequence) when another format is used. You can put it as the first command in your document:

> `\useOpTeX % we are using OpTeX format, no LaTeX :)`

8   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 (terminated 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

\begitems         % 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
\code             % another alternative 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
\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 {text}   % footnote (local numbering on each page)
\mnote {text}   % 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

\report \letter \slides  % style declaration macros
```

9   Compatibility with Plain TeX

All macros of Plain TeX are re-written in OpTeX. Common macros should work in the same sense as in original Plain TeX. Internal control sequences like `\p@` or `\f@@t` are removed and mostly replaced by control sequences prefixed by _ (like `\_this`). All primitives and common macros have two control sequences with the same meaning: in prefixed and unprefixed form. For example `\hbox` is equal to

`\_hbox`. Internal macros of OpTeX have and use only prefixed form. User should use unprefixed forms, but prefixed forms are accessible too, because the `_` is set as a letter category code globally (in macro files and in users document too). User should re-define unprefixed forms of control sequences with no worries that something internal will be broken (only the sequence `\par` cannot be re-defined without internal change of TeX behavior because it is hard-coded in TeXs tokenization processor).

The Latin Modern 8bit fonts instead Computer Modern 7bit fonts are preloaded in the format, ◀ but only few ones. The full family set is ready to use after the command `\fontfam[LMfonts]` which reads the fonts in OTF format.

The text accents macros like `\'`, `\v`, `\"` are undefined[8] in OpTeX. Use real letters like á, ř, ž ◀ in your source document instead these old accents macros. If you really want to use them, you can initialize them by `\oldaccents` command. But we don't recommend it.

The default paper size is not set as letter with 1 in margins but as A4 with 2.5 cm margins. You ◀ can change it, for example by `\margins/1 letter (1,1,1,1)in`. This example sets the classical Plain TeX page layout.

The origin for typographical area is not at top left 1 in 1 in coordinates but at top left paper ◀ corner exactly. For example, `\hoffset` includes directly left margin.

---

[8] The math accents macros like `\acute`, `\bar`, `\dot`, `\hat` still work.