

# OpTeX

## Format Based on Plain TeX and OPmac<sup>1</sup>

Version 0.13

*Petr Olšák, 2020*

<http://petr.olsak.net/optex>

OpTeX is LuaTeX format with Plain TeX and OPmac. Only LuaTeX engine is supported.

OpTeX should be a modern Plain TeX with power from OPmac (Fonts Selection System, colors, graphics, references, hyperlinks, indexing, bibliography, ...) with preferred Unicode fonts.

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 Fonts 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. For example sorting words in the Index<sup>2</sup>, reading .bib files directly<sup>2</sup>, syntax highlighting<sup>2</sup>, colors, graphics, hyperlinks, references).
- Macros are documented in the same place where code is.
- User name space of control sequences is separated from internal name space of OpTeX and primitives (\foo versus \\_foo). The name spaces for macro writers are designed too.

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 L<sup>A</sup>TeX 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. We have TeX. You can use OpTeX macros, understand them and modify them.

OpTeX offers a markup language for authors of texts (like L<sup>A</sup>TeX), i. e. the fixed set of tags to define the structure of the document. This markup is different from the L<sup>A</sup>TeX markup. It may offer to write the source text of the document somewhat clearer and more attractive.

The manual includes two parts: user documentation and technical documentation. The second part is generated directly from the sources of OpTeX. There are many hyperlinks from one part to second and vice versa.

This manual describes OpTeX features only. We suppose that user knows TeX basics. They are described in many books. You can see a short document [TeX in nutshell](#) too.

---

<sup>1</sup> The OPmac package is a set of simple additional macros to Plain TeX. It enables users to take advantage of L<sup>A</sup>TeX functionality but keeps Plain TeX simplicity. See <http://petr.olsak.net/opmac-e.html> for more information about it.

<sup>2</sup> All these features are implemented by TeX macros, no external program is needed.

# Contents

<b>1</b>	<b>User documentation</b>	<b>5</b>
1.1	Starting with OpTeX . . . . .	5
1.2	Page layout . . . . .	5
1.2.1	Setting the margins . . . . .	5
1.2.2	Concept of default page . . . . .	6
1.2.3	Footnotes and marginal notes . . . . .	7
1.3	Fonts . . . . .	7
1.3.1	Font families . . . . .	7
1.3.2	Font sizes . . . . .	8
1.3.3	Typesetting math . . . . .	9
1.4	Typical elements of document . . . . .	10
1.4.1	Chapters and sections . . . . .	10
1.4.2	Another numbered objects . . . . .	10
1.4.3	References . . . . .	11
1.4.4	Hyperlinks, outlines . . . . .	12
1.4.5	Lists . . . . .	13
1.4.6	Tables . . . . .	14
1.4.7	Verbatim . . . . .	16
1.5	Autogenerated lists . . . . .	18
1.5.1	Table of contents . . . . .	18
1.5.2	Making the index . . . . .	18
1.5.3	BibTeXing . . . . .	19
1.6	Graphics . . . . .	20
1.6.1	Colors . . . . .	20
1.6.2	Images . . . . .	21
1.6.3	PDF transformations . . . . .	22
1.6.4	Ovals, circles . . . . .	23
1.6.5	Putting images and texts wherever . . . . .	23
1.7	Others . . . . .	23
1.7.1	Using more languages . . . . .	23
1.7.2	Pre-defined styles . . . . .	24
1.7.3	Loading other macro packages . . . . .	25
1.7.4	Lorem ipsum dolor sit . . . . .	25
1.7.5	Logos . . . . .	25
1.7.6	The last page . . . . .	25
1.7.7	Use OpTeX . . . . .	26
1.8	Summary . . . . .	26
1.9	Compatibility with Plain T <sub>E</sub> X . . . . .	27
<b>2</b>	<b>Technical documentation</b>	<b>28</b>
2.1	The main initialization file . . . . .	28
2.2	Concept of name spaces of control sequences . . . . .	30
2.2.1	Prefixing internal control sequences . . . . .	30
2.2.2	Name space of control sequences for users . . . . .	30
2.2.3	Macro files syntax . . . . .	31
2.2.4	Name spaces for package writers . . . . .	31
2.2.5	Summary about rules for external macro files published for OpTeX . . . . .	31
2.2.6	The implementation of the name spaces . . . . .	32
2.3	pdfTeX initialization . . . . .	33

2.4	Basic macros . . . . .	35
2.5	Allocators for T <sub>E</sub> X registers . . . . .	36
2.6	If-macros, loops, is-macros . . . . .	38
2.6.1	Classical <code>\newif</code> . . . . .	38
2.6.2	Loops . . . . .	38
2.6.3	Is-macros . . . . .	39
2.7	Setting parameters . . . . .	41
2.7.1	Primitive registers . . . . .	41
2.7.2	Plain T <sub>E</sub> X registers . . . . .	42
2.7.3	Different settings than in plain T <sub>E</sub> X . . . . .	42
2.7.4	OpT <sub>E</sub> X parameters . . . . .	43
2.8	More OpT <sub>E</sub> X macros . . . . .	47
2.9	Plain T <sub>E</sub> X macros . . . . .	49
2.10	Preloaded fonts for text mode . . . . .	54
2.11	Scaling fonts in text mode (low-level macros) . . . . .	54
2.11.1	The <code>\fontdef</code> declarator . . . . .	55
2.11.2	The <code>\fontlet</code> declarator . . . . .	55
2.11.3	Optical sizes . . . . .	55
2.11.4	Implementation notes . . . . .	56
2.12	The Font Selection System . . . . .	58
2.12.1	Terminology . . . . .	58
2.12.2	Font families, selecting fonts . . . . .	59
2.12.3	Math Fonts . . . . .	59
2.12.4	Declaring font commands . . . . .	59
2.12.5	Modifying font features . . . . .	60
2.12.6	Special font modifiers . . . . .	61
2.12.7	How to create the font family file . . . . .	61
2.12.8	How to write the font family file with optical sizes . . . . .	64
2.12.9	How to register the font family in the Font Selection System . . . . .	64
2.12.10	Implementation of the Font Selection System . . . . .	65
2.13	Preloaded fonts for math mode . . . . .	70
2.14	Math macros . . . . .	72
2.15	Unicode-math fonts . . . . .	81
2.15.1	Unicode-math macros preloaded in the format . . . . .	81
2.15.2	Macros and codes set when <code>\loadmatfont</code> is processed . . . . .	83
2.15.3	A few observations . . . . .	88
2.16	Scaling fonts in document (high-level macros) . . . . .	88
2.17	Output routine . . . . .	91
2.18	Margins . . . . .	93
2.19	Colors . . . . .	94
2.20	The <code>.ref</code> file . . . . .	99
2.21	References . . . . .	101
2.22	Hyperlinks . . . . .	102
2.23	Making table of contents . . . . .	103
2.24	PDF outlines . . . . .	105
2.24.1	Nesting PDF outlines . . . . .	105
2.24.2	Strings in PDF outlines . . . . .	106
2.25	Chapters, sections, subsections . . . . .	107
2.26	Lists, items . . . . .	112
2.27	Verbatim, listings . . . . .	113
2.27.1	Inline and “display” verbatim . . . . .	113

2.27.2	Listings with syntax highlighting . . . . .	117
2.28	Graphics . . . . .	120
2.29	The <code>\table</code> macro . . . . .	125
2.29.1	The boundary declarator : . . . . .	125
2.29.2	Usage of the <code>\tabskip</code> primitive . . . . .	125
2.29.3	Tables to given width . . . . .	126
2.29.4	Implemetation of the <code>\table</code> macro . . . . .	126
2.30	Balanced multi-columns . . . . .	131
2.31	Citations, bibliography . . . . .	132
2.31.1	Macros for citations and bibliography preloaded in the format . . . . .	132
2.31.2	The <code>\usebib</code> command . . . . .	135
2.31.3	The <code>usebib.opm</code> macro file loaded when <code>\usebib</code> is used . . . . .	136
2.31.4	Usage of the <code>bib-iso690</code> style . . . . .	139
2.31.5	Implementation of the <code>bib-iso690</code> style . . . . .	146
2.32	Sorting and making Index . . . . .	150
2.33	Footnotes and marginal notes . . . . .	156
2.34	Styles . . . . .	158
2.34.1	<code>\report</code> and <code>\letter</code> styles . . . . .	158
2.34.2	<code>\slides</code> style for presentations . . . . .	159
2.35	Logos . . . . .	162
2.36	Multilingual support . . . . .	162
2.36.1	Lowercase, uppercase codes . . . . .	162
2.36.2	Hyphenations . . . . .	163
2.36.3	Multilingual phrases and quotation marks . . . . .	166
2.37	Other macros . . . . .	167
2.38	Printing documentation . . . . .	168
	<b>Index</b>	<b>172</b>

# Chapter 1

## User documentation

### 1.1 Starting with OpTeX

OpTeX is compiled as a format for LuaTeX. Maybe there is a command `optex` in your TeX distribution. 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<sup>1</sup>. 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 1.3.1)
\typoysize[11/13] % setting default font size and baselineskip (sec. 1.3.2)
\margins/1 a4 (1,1,1,1)in % setting A4 paper, 1 in margins (section 1.2.1)
\cslang           % Czech hyphenation patterns (section 1.7.1)
```

```
Tady je zkušební textík v českém jazyce.
\bye
```

You can look at `op-demo.tex` file for more complex, but still simple example.

### 1.2 Page layout

#### 1.2.1 Setting the margins

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>`.

---

<sup>1</sup> This is a technical limitation of LuaTeX for fonts downloaded in formats: only 8bit fonts can be preloaded.

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 automatically mirrored in such case, it means that  $\langle left \rangle$  is replaced by  $\langle right \rangle$  and vice versa.

OpTeX declares following 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  $\langle fmt \rangle$  can be also 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 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.

## 1.2.2 Concept of default page

OpTeX uses for page design very similar to Plain TeX “output routine”. 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 1.2.1) 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 the `\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`.

## 1.2.3 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<sup>2</sup>. 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 then you can use the command `\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 `\fnotetexts` have to be written after the box as the number of `\fnotemarks` 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.

## 1.3 Fonts

### 1.3.1 Font families

You can select the font family by `\fontfam[<Family-name>]`. The argument `<Family-name>` is case insensitive and spaces are ignored in it. For example, `\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

---

<sup>2</sup> 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`



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 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 First text \it Second text` gives `FIRST TEXT SECOND TEXT`.

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

The context between variants `\rm ↔ \it` and `\bf ↔ \bi` is kept by the `\em` macro (emphasis 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, if needed. 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 technical documentation in the section 2.12. You can mix more font families in your document, you can declare your own variant selectors or modifiers etc.

### 1.3.2 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 before closing 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 `\mainfontsize` 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.



```
\typsize[12/14.4] % first usage in document, sets main values internally
\typsize[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. It doesn't change the font size immediately, but following variant selector does it. 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 the section 2.11.

### 1.3.3 Typesetting math

OpTeX preloads a collection of 7bit Computer Modern math fonts and AMS fonts in its format for math typesetting. You can use them in any size and in the `\boldmath` variant. Most declared text font families (see `\fontfam` in the section 1.3.1) are configured with recommended Unicode math font. This font is automatically 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 `\fontfam` command.

Hundreds math symbols and operators like in AMSTeX are accessible. For example `\alpha`  $\alpha$ , `\geq`  $\geq$ , `\sum`  $\sum$ , `\sphericalangle`  $\sphericalangle$ , `\bumpeq`  $\bumpeq$ ,  $\simeq$ . See AMSTeX 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-XYZ</i>
<code>\script</code>	% script	<i>ABC-XYZ</i>
<code>\frak</code>	% fracture	<i>abc-xyz, ABC-XYZ</i>
<code>\bbchar</code>	% double stroked letters	<i>ABC-XYZ</i>
<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 in math regardless the current text font family. This is common notation for vectors and matrices. You can re-declare them, see the file `unimath-codes.opm` where Unicode math variants of `\bf` and `\bi` selectors are defined.

The math fonts can be scaled by `\typsize` and `\typoscale` macros. Two math fonts collections are prepared: `\normalmath` for normal weight and `\boldmath` for bold. The first one is set by default, the second one is usable for math formulae in titles typeset in bold, for example.

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). Moreover, there is the macro `\mathstyles{<math list>}` which depends on the current math style. It is documented at the end of the section 2.14.

The `\eqalign` macro is more flexible in OpTeX, see technical documentation about it.

## 1.4 Typical elements of document

### 1.4.1 Chapters and sections

The document can be divided into chapters (`\chap`), sections (`\sec`), subsections (`\secc`) and they can be 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 the 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  $\text{\TeX}$  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. You can combine both prefixes.

### 1.4.2 Another numbered objects

Apart from chapters, sections and subsections, there are another automatically numbered objects: equations, captions for tables and figures. The user can declare more numbered object.

If the 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}
```

Another automatically numbered object is a caption 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
\noindent\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 1.4.1** 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– $\infty$	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 1.7.1 about languages).

If you wish to make the table or figure as floating object, you need to use Plain T<sub>E</sub>X macros `\midinsert` or `\topinsert` terminated by `\endinsert`. Example:

```

\topinsert % table and its caption printed at the top of the current page
<caption and table>
\endinsert

```

There are five prepared counters A, B, C, D and E. They are reset in each chapter and section<sup>3</sup>. They can be used in context of `\numberedpar <letter>{<text>}` macro. For example:

```

\def\theorem    {\numberedpar A{Theorem}}
\def\corollary  {\numberedpar A{Corollary}}
\def\definition {\numberedpar B{Definition}}
\def\example    {\numberedpar C{Example}}

```

Three independent numbers are used in this example. One for Theorems and Corollaries second for Definitions and third for Examples. The user can write `\theorem Let  $M$  be...` and the new paragraph is started with the text: **Theorem 2.3.1.** Let  $M$  be... You can add an optional parameter in brackets. For example, `\theorem [(L'Hôpital's rule)] Let  $f, g$  be...` is printed like **Theorem 2.4.2 (L'Hôpital's rule).** Let  $f, g$  be...

### 1.4.3 References

Each automatically numbered object documented in sections 1.4.1 and 1.4.2 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:

<sup>3</sup> This feature can be changed, see the section 2.25 in the technical documentation.

```
\sec[beatle] About Beatles
```

```
\noindent\hfil\table{rl}{...} % the table
```

```
\cskip
```

```
\caption/t [comp-depend] The dependency of the comp-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 use the `\label[<label>]` before the `\theorem`, `\definition` etc. (macros defined by `\numberedpar`) if you want to reference these numbered objects. You can't use `\theorem[<label>]` because the optional parameter is reserved to another purpose here.

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 your document then you can declare `\showlabels`. The labels are printed at their destination places after such declaration.

#### 1.4.4 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.

The hyperlinked footnotes can be activated by `\fnotelinks <color-fnt> <color-fnf>` where footnote marks in text have `<color-fnt>` and the same footnote marks in footnotes have `<color-fnf>`. You can define relevant borders `\fntborder` and `\fnfborder` analogically as `\pgborder` (for example).

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 declared.

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<sup>4</sup>. 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}` outputs to the text `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 as 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 unprotected 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 1.5.1 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 the `\insertoutline` macro is used.

### 1.4.5 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 the `\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
```

---

<sup>4</sup> More exactly, there are the same rules as for `\code` command, see section 1.4.7.

```
* Finito
\enditems
```

produces:

- First idea
- Second idea in subitems:
  - (i) First sub-idea
  - (ii) Second sub-idea
  - (iii) Last sub-idea
- Finito

Another style can be defined by the command `\sdef{<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` value which is set to `\parindent` by default. The `\ilevel` register says what level of items is currently processed. Each `\begitems` starts `\everylist` tokens register. You can set, for example:

```
\everylist={\ifcase\ilevel\or \style X \or \style x \else \style - \fi}
```

You can say `\begitems \novspaces` if you don't want vertical spaces above and below the list. The nested item list are without vertical spaces automatically. More information about design of lists of items should be found in the section [2.26](#).

### 1.4.6 Tables

The macro `\table{<declaration>}{<data>}` provides similar `<declaration>` of tables as in  $\text{\LaTeX}$ : 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||}{
  Month      & commodity  & price  \crl
  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 with paragraphs of given width. More precisely, a long text in the table cell is printed as an paragraph with given width. By default, the paragraph is left-right justified. But there are alternatives:

- `p{<size>\fL}` fit left, i.e. left justified, ragged right,
- `p{<size>\fR}` fit right, i.e. right justified, ragged left,
- `p{<size>\fC}` fit center, i.e. ragged left plus right,
- `p{<size>\fS}` fit special, short one-line pararaph centered, long paragraph normal,
- `p{<size>\fX}` fit extra, left-right justified but last line centered.

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

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

The command `\cr` used in the  $\langle data \rangle$  part of the table is generally known. It marks the end of each row in 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{\langle list \rangle}` ... like `\crli` but the lines are drawn only in the columns mentioned in comma separated  $\langle list \rangle$  of their numbers. The  $\langle list \rangle$  can include  $\langle from \rangle$ - $\langle to \rangle$  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>}` immediately 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 in \vbox before table processing
\thistable={}           % code used in \vbox, it 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 vert. space before/after horizontal lines
\vvkern=1pt             % space between lines in double vertical line
\hhkern=1pt             % space between lines in double horizontal line
\tabskip=0pt            % space between columns
\tabskipl=0pt \tabskipr=0pt % space before first and after last column
```

Example: if you do `\tabiteml={\enspace}\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 the macro `\multispan{\langle number \rangle}` (from Plain T<sub>E</sub>X) can help you. Another alternative is the command `\mspan{\langle number \rangle}[\langle declaration \rangle]{\langle text \rangle}` which spans  $\langle number \rangle$  columns and formats the  $\langle text \rangle$  by the  $\langle declaration \rangle$ . The  $\langle declaration \rangle$  must include a declaration of only one column with the same syntax as common `\table \langle declaration \rangle`. If your table includes vertical rules and you want to create continuous vertical rules by `\mspan`, then use rule declarators `|` after `c`, `l` or `r` letter in `\mspan \langle declaration \rangle`. 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{\langle text \rangle}` makes a frame around  $\langle text \rangle$ . 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 `\vspan<number>\{<text>\}` shifts the `<text>` down in order it looks like to be in the center of the `<number>` lines (current line is first). You can use this for creating tables like in the following example:

```
\thistable{\tabstrut={\vrule height 20pt depth10pt width0pt}
\baselineskip=20pt \tablinespace=0pt \rulewidth=.8pt}
\table{\8{c|}}{\crlp{3-8}
\mspan2[c|]{ } & \mspan3[c|]{Singular} & \mspan3[c|]{Plural} \crlp{3-8}
\mspan2[c|]{ } & Neuter & Masculine & Feminine & Masculine & Feminine & Neuter \crl
\vspan2{I} & Inclusive & \mspan3[c|]{\vspan2{0}} & \mspan3[c|]{X} \crlp{2,6-8}
& Exclusive & \mspan3[c|]{ } & \mspan3[c|]{X} \crl
\vspan2{II} & Informal & \mspan3[c|]{X} & \mspan3[c|]{X} \crlp{2-8}
& Formal & \mspan6[c|]{X} \crl
\vspan2{III} & Informal & \vspan2{0} & X & X & \mspan2[c|]{X} & \vspan2{0} \crlp{2,4-7}
& Formal & & & & \mspan4[c|]{X} & \crl
}
```

The `<number>` parameter of `\vspan` must be one-digit number. If you want to set more digits then use braces. You can use non-integer values too if you feel that the result is better, for example `\vspan{2.1}\{text\}`.

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.

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 section 2.29.4. See the definition of the `\tabdeclarec` macro, for example.

The `:` columns boundary declarator is described in section 2.29.1. The tables with given width can be declared by `to<size>` or `xto<size>`. More about it is in section 2.29.3 Many tips about tables can be seen on the site <http://petr.olsak.net/optex/optex-tricks.html>.

		Singular			Plural		
		Neuter	Masculine	Feminine	Masculine	Feminine	Neuter
I	Inclusive	O			X		
	Exclusive				X		
II	Informal	X			X		
	Formal	X					
III	Informal	O	X	X	X		O
	Formal		X				

## 1.4.7 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. OpT<sub>E</sub>X 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 parameter of `\everytt` is read. So you don't have to worry about active categories in this parameter.

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 declared 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={\visiblesp}      % visualization of spaces
```

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 have to 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, 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 the `\verbininput` command. Examples:

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

You can insert additional commands for the `\verbininput` before first opening bracket. They are processed in the local group. For example, `\verbininput \hsize=20cm (-) program.c`.

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 `\verbininput` can be controlled by `\everytt`, `\ttindent` just like in `\begtt... \endtt`.

The `\begtt... \endtt` pair or `\verbininput` can be used for listings of codes. Automatic syntax highlighting is possible, for example `\begtt \hisyntax{C}` activates colors for C programs. Or `\verinput \hisyntax{HTML} (-) file.html` can be used for HTML or XML codes. OpTeX implements C, Python, TeX, HTML and XML syntax highlighting. More languages can be declared, see the section [2.27.2](#).

## 1.5 Autogenerated lists

### 1.5.1 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  $\text{\TeX}$  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 need a customization of the design of the TOC, read the section 2.23.

If you are using a special macro in section 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: then 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}{\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
```

### 1.5.2 Making the index

The index can be included into document by the `\makeindex` macro. No external program is needed, the alphabetical sorting are done inside  $\text{\TeX}$  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 as follows:

```
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 `~`.

The `\ii` or `\iid` commands can be 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. The T<sub>E</sub>Xbook index is good 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 `\begmulti` and `\endmulti` 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, 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 the `\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 1.7.1 about languages selection.

### 1.5.3 BibT<sub>E</sub>Xing

The command `\cite[<label>]` (or `\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 formatting 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 features of the `iso690` style is documented in the section 2.31.4 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 tells that mentioned bib-record should be printed in the reference list. If `\nocite[*]` is used then all records from `⟨bib-base⟩` are printed.

## 1.6 Graphics

### 1.6.1 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. There is about 300 color names like `\DeepPink`, `\Chocolate` etc. If there are numbered variants of the same name, then the letters B, C, etc. are appended 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 technical documentation, section 2.19 for more information.

The basic colors `\Blue`, `\Red`, `\Cyan`, `\Yellow` etc. are defined with CMYK components using `\setcmykcolor`. On the other hand, you can define a color with three RGB components 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<sup>5</sup> 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, 20% 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 subtractive 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 the additive RGB color space.

The following example defines the macro for 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{#2#3}%
  \leavevmode \rlap{#1\strut \vrule width\wd0}\box0
}
\coloron\Yellow\Brown{The brown text on the yellow background}
```

## 1.6.2 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 the `\picdir` declaration.

---

<sup>5</sup> Printed output is more equal to the monitor preview specially if you are using ICC profile for your printer.



Inkscape<sup>6</sup> is able to save a picture to PDF and labels of the picture to another file<sup>7</sup>. This second file should be read by T<sub>E</sub>X in order to print labels in the same font as document font. OpT<sub>E</sub>X supports this feature by `\inkinspic {<filename>.pdf}` command. It reads and displays both: PDF image and labels generated by Inkscape.

If you want to create a vector graphics (diagrams, schema, geometry skicing) then you can do it by 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 T<sub>E</sub>X.

### 1.6.3 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  $[\langle a \rangle, \langle b \rangle]$  and  $[1, 0]$  to  $[\langle c \rangle, \langle d \rangle]$ . 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 T<sub>E</sub>X’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.

OpT<sub>E</sub>X provides two special transformation macros `\pdfscale` and `\pdfrotate`:

```
\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:

*text1* *text2* *text3*

You can see that T<sub>E</sub>X knows nothing about dimensions of transformed material, it treats it as with a zero dimension object. It can be solved by the `\transformbox{<transformation>}{<text>}` macro. This macro 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 shortcut for `\transformbox{\pdfrotate{<deg>}}{<text>}`.

<sup>6</sup> A powerfull and free wysiwyg editor for creating vector graphics.

<sup>7</sup> Chose “Omit text in PDF and create LaTeX file” option.




## 1.6.4 Ovals, circles

The `\inoval{<text>}` creates a box like this: . Multiline text can be put in an oval by the command `\inoval{\vbox{<text>}}`. Local settings can be set by `\inoval[<settings>]{<text>}` or you can re-declare global settings by `\ovalparams={<settings>}`. The default settings are:

```
\ovalparams={\roundness=2pt          % diameter of circles in the corners
              \fcolor=\Yellow          % color used for filling oval
              \lcolor=\Red             % line color used in the border
              \linewidth=0.5bp        % line width in the border
              \shadow=N               % use a shadow effect
              \overlapmargin=N        % ignore margins by surrounding text
              \hhkern=0pt \vbkern=0pt % left-right margin, top-bottom margin}
```

The total distance from text to oval boundary is `\hhkern+\roundness` at the left and right sides and `\vbkern+\roundness` at the top and bottom sides of the text.

If you need to set a parameters for the `<text>` (color, size, font etc.), put such setting right in front of the `<text>`: `\inoval{<text settings><text>}`.

The `\incircle[<ratio=1.8>]{<text>}` creates a box like this . The `\ratio` parameter means width/height. The usage is analogical like for oval. The default parameters are

```
\circleparams={\ratio=1 \fcolor=\Yellow \lcolor=\Red \linewidth=0.5bp
                \shadow=N \ignoremargin=N \hhkern=2pt \vbkern=2pt}
```

The macros `\clipinoval <x> <y> <width> <height> {<text>}` and `\clipincircle` (with the same parameters) print the `<text>` when a clipping path (oval or circle with given `<with>` and `<height>` shifted its center by `<x>` to right and by `<y>` to up) is used. The `\roundness=5mm` is default for `\clipinoval` and user can change it. Example:

```
\clipincircle 3cm 3.5cm 6cm 7cm {\picw=6cm \inspic{myphoto.jpg}}
```

## 1.6.5 Putting images and texts wherever

The `\puttext <x> <y> {<text>}` puts the `<text>` shifted by `<x>` right and by `<y>` up from current point of typesetting and does'n not change the position of the current point. Assume coordinate system with origin in the current point. Then `\puttext <x> <y> {<text>}` puts the text at the coordinates `<x>`, `<y>`. More exactly the left edge of its baseline is at that position.

The `\putpic <x> <y> <width> <height> {<image>}` puts the `<image>` of given `<width>` and `<height>` at given position (its left-bottom corner). You can write `\nospec` instead `<width>` or `<height>` if this parameter is not given.

## 1.7 Others

### 1.7.1 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(clas-
sicLatin) 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 `\uselanguage{⟨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 `\Cgplain`, 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 phrases used for captions and titles in technical articles or books: “Chapter”, “Table” and “Figure”. These phrases 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 section 2.36.3.

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 section 2.32.

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 visually 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\altquotes` in such case.

## 1.7.2 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 `\n1` to more lines.

The `\letter` declaration is intended to create letters. See the files `op-letter-*.tex` for examples. 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 `\langle prefixed text \rangle\address...` to put `\langle prefixed text \rangle` before first line of the address.

The `\slides` style creates 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 macro `\book` is not prepared. Each book needs an individual typographical care. You need to create specific macros for design.

### 1.7.3 Loading other macro packages

You can load more macro packages by `\input{\langle file-name \rangle}` or by `\load[\langle file-names \rangle]`. The first case is  $\text{\TeX}$  primitive command, it can be used in the alternative old syntax `\input \langle filename \rangle \langle space \rangle` too. The second case allows to specify a comma separated list of included files. Moreover, the `\load` macro loads each macro file only once, it sets temporarily standard category codes during loading and it tries to load `\langle filename \rangle.opm` or `\langle filename \rangle.tex` or `\langle filename \rangle`, first occurrence wins. Example:

```
\load [qrcode, tikz]
```

does `\input qrcode.opm` and `\input tikz.tex` and it saves local information about the fact that these file names `qrcode` and `tikz` were already used, i. e. next `\load` will skip them.

It is strongly recommended to use the `\load` macro for loading external macros, if you need them. On the other hand, if your document is structured to more files, use simply the `\input` primitive.

### 1.7.4 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[\langle number \rangle]` or `\lorem[\langle from \rangle-\langle to \rangle]`. It prints a paragraph (or paragraphs) with neutral text. The numbers `\langle number \rangle` or `\langle from \rangle`, `\langle to \rangle` 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.

### 1.7.5 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.
```

### 1.7.6 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  $\text{\TeX}$  run and to the last page in next  $\text{\TeX}$  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 numeral (when `\pageno` is negative). If you need to know total pages used in the document, use `\totalpages` macro. It expands to zero (in first  $\text{\TeX}$  run) or to the number of all pages in the document (in next  $\text{\TeX}$  runs).

### 1.7.7 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 :)
```

## 1.8 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

\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
\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 lists 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}

\load [<filenames>] % loading macro files
\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} % 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
```

## 1.9 Compatibility with Plain T<sub>E</sub>X

All macros of Plain T<sub>E</sub>X are re-written in OpT<sub>E</sub>X. Common macros should work in the same sense as in original Plain T<sub>E</sub>X. 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 OpT<sub>E</sub>X 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 T<sub>E</sub>X behavior because it is hard-coded in T<sub>E</sub>Xs 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.

Plain T<sub>E</sub>X defines `\newcount`, `\bye` etc. as `\outer` macros. OpT<sub>E</sub>X doesn't set any macro as `\outer`. Macros like `\TeX`, `\rm` are defined as `\protected`.

The text accents macros `\"`, `\'`, `\v`, `\u`, `\=`, `\^`, `\.`, `\H`, `\~`, `\``, `\t` are undefined<sup>8</sup> in OpT<sub>E</sub>X. Use real letters like á, ř, ž in your source document instead of these old accents macros. If you really want to use them, you can initialize them by the `\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 T<sub>E</sub>X 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.

The `\sec` macro is reserved to sections but original Plain T<sub>E</sub>X declares this control sequence for math secans.

---

<sup>8</sup> The math accents macros like `\acute`, `\bar`, `\dot`, `\hat` still work.

# Chapter 2

## Technical documentation

This documentation is written in the source files \*.opm between the `\_doc` and `\_cod` pairs or after the `\_endcode` command. When the format is generated by

```
luatex -ini optex.ini
```

then the text of the documentation is ignored and the format `optex.fmt` is generated. On the other hand, if you run

```
optex optex-doc.tex
```

then the same \*.opm files are read when the second chapter of this documentation is printed.

A knowledge about T<sub>E</sub>X is expected from the reader. You can see a short document [T<sub>E</sub>X in a Nutshell](#) or more detail [T<sub>E</sub>X by topic](#).

Notices about hyperlinks. If a control sequence is printed in red color in this documentation then this denotes its “main documentation point”. Typically, the listing where the control sequence is declared follows immediately. If a control sequence is printed in the blue color in the listing or in the text then it is active link which points (usually) to the main documentation point. The main documentation point can be active link which points to a previous text where the control sequence was mentioned. Such occurrences are active links to the main documentation point.

## 2.1 The main initialization file

The `optex.ini` file is read as main file when the format is generated.

```
1 %% This is part of OpTeX project, see http://petr.olsak.net/optex
2
3 %% OpTeX ini file
4 %% Petr Olsak <project started from: Jan. 2020>
```

Category codes are set first. Note that the `_` is set to category code “letter”, it can be used as a part of control sequence names. Other category codes are set as in the plain T<sub>E</sub>X.

```
6 % Catcodes:
7
8 \catcode \{=1 % left brace is begin-group character
9 \catcode \}=2 % right brace is end-group character
10 \catcode \$=3 % dollar sign is math shift
11 \catcode \&=4 % ampersand is alignment tab
12 \catcode \#=6 % hash mark is macro parameter character
13 \catcode \^=7 %
14 \catcode \^K=7 % circumflex and uparrow are for superscripts
15 \catcode \^A=8 % downarrow is for subscripts
16 \catcode \^I=10 % ascii tab is a blank space
17 \catcode \_ =11 % underline can be used in control sequences
18 \catcode \~=13 % tilde is active
19 \catcode \^a0=13 % non breaking space in Unicode
20 \catcode 127=12 % normal character
```

The `\optexversion` and `\fmtname` are defined.

```
22 % OpTeX version
23
24 \def\optexversion{Beta 0.13 May 2020}
25 \def\fmtname{OpTeX}
```

We check if LuaT<sub>E</sub>X engine is used at `-ini` state. And the `^^J` character is set as `\newlinechar`.

```
27 % Engine testing:
28
29 \newlinechar=^^J
30 \ifx\directlua\undefined
31   \message{This format is based only on LuaTeX, use luatex -ini optex.ini^^J}
32 \endif
```

```

33
34 \ifx\bgroup\undefined \else
35   \message{This file can be used only for format initialisation, use luatex -ini^^J}
36 \endinput \fi

```

The basic macros for macro file syntax is defined, i.e. `\endcode`, `\_doc` and `\_cod`. The `\_codedecl` will be re-defined later.

```

38 % Basic .opm syntax:
39
40 \let\endcode =\endinput
41 \def \_codedecl #1#2{\message{#2^^J}}% information about .opm file
42 \long\def \_doc#1\_cod#2 {} % skip documentation

```

optex.ini

Individual \*.opm macro files are read.

```

44 % Initialization:
45
46 \message{OpTeX (Olsak's Plain TeX) initialization <\optexversion>^^J}
47
48 \input prefixed.opm      % prefixed primitives and code syntax
49 \input luatex-ini.opm   % luaTeX initialization
50 \input basic-macros.opm % basic macros
51 \input alloc.opm        % allocators for registers
52 \input if-macros.opm    % special \if-macros, \is-macros and loops
53 \input parameters.opm   % parameters setting
54 \input more-macros.opm  % OpTeX useful macros (todo: doc)
55 \input plain-macros.opm % plainTeX macros (todo:doc)
56 \input fonts-preload.opm % preloaded Latin Modern fonts
57 \input fonts-resize.opm % font resizing (low-level macros) (todo: texdoc)
58 \input fonts-select.opm % font selection system (todo: texdoc)
59 \input math-preload.opm % math fams CM + AMS preloaded (todo: doc)
60 \input math-macros.opm  % basic macros for math plus mathchardefs (todo: x)
61 \input math-unicode.opm % macros for loading UnicodeMath fonts (todo: x)
62 \input fonts-opmac.opm  % font managing macros from OPmac (todo: doc)
63 \input output.opm       % output routine
64 \input margins.opm      % macros for margins setting (todo: texdoc)
65 \input colors.opm       % colors
66 \input ref-file.opm     % ref file
67 \input references.opm   % references
68 \input hyperlinks.opm  % hyperlinks
69 \input maketoc.opm      % maketoc
70 \input outlines.opm     % PDF outlines (todo: x)
71 \input pdfuni-string.opm % PDFUnicode strings for outlines (todo: x)
72 \input sections.opm     % titles, chapters, sections
73 \input lists.opm        % lists, \begitems, \enditems
74 \input verbatim.opm     % verbatim
75 \input hi-syntax.opm    % syntax highlighting of verbatim listings
76 \input graphics.opm     % graphics
77 \input table.opm        % table macro
78 \input multicolumns.opm % more columns by \begmulti ... \endmulti
79 \input cite-bib.opm     % Bibliography, \cite
80 \input makeindex.opm    % Make index and sorting
81 \input fnotes.opm       % \fnotes, \mnotes
82 \input styles.opm       % styles \report, \letter
83 \input logos.opm        % standard logos
84 \input uni-lcuc.opm     % Setting lccodes and uccodes for Unicode characters
85 \input hyphen-lan.opm   % initialization of hyphenation patterns (todo: doc)
86 \input languages.opm    % languages
87 \input others.opm       % miscenaleous

```

optex.ini

The `\everyjob` register is initialized and the format is saved by the `\dump` command.

```

89 \_everyjob = {%
90   \message{This is OpTeX (Olsak's Plain TeX), version <\optexversion>^^J}%
91   \_mathchardef\_fnotestack=\_pdfcolorstackinit page {0 g 0 G}%
92   \_mathsbon % replaces \int _a^b to \int _a^b
93   \_inputref % inputs \jobname.ref if exists
94 }
95

```

optex.ini



## 2.2 Concept of name spaces of control sequences

### 2.2.1 Prefixing internal control sequences

All control sequences used in OpTeX are used and defined with `_` prefix. The user can be sure that when he/she does `\def\foo` then internal macros of OpTeX nor TeX primitives will be not damaged. For example `\def\if{...}` will not damage macros because OpTeX's macros are using `\_if` instead of `\if`.

All TeX primitives are initialized with two representative control sequences: `\word` and `\_word`, for example `\hbox` and `\_hbox`. The first alternative is reserved for users or such control sequences can be re-defined by user.

OpTeX sets the character `_` as letter, so it can be used in control sequences. When a control sequence begins with this character then it means that it is a primitive or it is used in OpTeX macros as internal. User can redefine such prefixed control sequence only if he/she explicitly know what happens.

We never change catcode of `_`, so internal macros can be redefined by user without problems if it is desired. We need not something like `\makealetter` from L<sup>A</sup>T<sub>E</sub>X.

OpTeX defines all new macros as prefixed. For public usage of such macros we need to set non-prefixed version. This is done by

```
\public <list of control sequences> ;
```

For example `\public \foo \bar ;` does `\let\foo=\_foo, \let\bar=\_bar`.

At the end of each code segment in OpTeX, the `\_public` macro is used. You can see, what macros are defined for public usage in such code segment.

The macro `\private` does a reverse job to `\public` with the same syntax. For example `\private \foo \bar ;` does `\let\_foo=\foo, \let\_bar=\bar`. This should be used when unprefix variant of control sequence is declared already but we need the prefixed variant too.

In this documentation: if both variants of a control sequence are declared (prefixed and unprefix), then the accompanying text mentions only unprefix variant. The code typically defines prefixed variant and then the `\public` (or `\_public`) macro is used.

### 2.2.2 Name space of control sequences for users

User can define or declare any control sequence with a name without any `_`. This does not make any problem. Only one exception is the reserved control sequence `\par`. It is generated by tokenizer (at empty lines) and used as internal in TeX.

User can define or declare control sequences with `_` character, for example `\my_control_sequence`, but with the following exceptions:

- Control sequences which begin with `_` are reserved for TeX primitives, OpTeX internal macros and macro package writers.
- Control sequences (terminated by non-letter) in the form `\<word>_` or `\<word>_<one-letter>`, where `<word>` is a sequence of letters, are inaccessible, because they are interpreted as `\<word>` followed by `_` or as `\<word>` followed by `_<one-letter>`. This is important for writing math, for example:

```
\int_a^b    ... is interpreted as \int_a^b
\max_M      ... is interpreted as \max_M
\alpha_{ij} ... is interpreted as \alpha_{ij}
```

This feature is implemented using lua code at input processor level, see the section 2.14 for more details. You can deactivate this feature by `\mathsboff`. After this, you can still write  $\int_a^b$  (Unicode) or  $\int_a^b$  without problems but `\int_a^b` yields to undefined control sequence `\int_a`. You can activate this feature again by `\mathsbon`. The effect will take shape from next line read from input file.

- Control sequences in the form `\_<pkg>_<word>` is intended for package writers as internal macros for a package with `<pkg>` identifier, see section 2.2.4.

The single letter control sequences like `\%`, `\$`, `\^` etc. are not used in internal macros. User can redefine them, but (of course) some classical features can be lost (printing percent character by `\%` for example).

### 2.2.3 Macro files syntax

Each segment of OpTeX macros is stored in one file with `.opm` extension (means OPTex Macros). Your local macros should be in normal `*.tex` file.

The code in macro files starts by `\_codedec1` and ends by `\_endcode`. The `\_endcode` is equivalent for `\endinput`, so documentation can follow. The `\_codedec1` has syntax:

```
\_codedec1 \sequence {Name <version>}
```

If the mentioned `\sequence` is defined, then `\_codedec1` does the same as `\endinput`: this protect from reading the file twice. We suppose, that `\sequence` is defined in the macro file.

It is possible to use the `\_doc ... \_cod` pair between the macro lines. The documentation text should be here. It is ignored when macros are read but it can be printed using `doc.opm` macros like in this documentation.

### 2.2.4 Name spaces for package writers

Package writer should use internal names in the form `\_<pkg>_<sequence>`, where `<pkg>` is a package label. For example: `\_qr_utfstring` from `qrcode.opm` package.

The package writer needs not write repeatedly `\_pkg_foo \_pkg_bar` etc. again and again in the macro file.<sup>1</sup> When the `\_namespace {<pkg>}` is declared at the beginning of the macro file then all occurrences of `\_foo` will be replaced by `\_<pkg>_foo` at the input processor level. The macro writer can write (and backward can read his/her code) simply with `\_foo`, `\_bar` control sequences and `\_<pkg>_foo`, `\_<pkg>_bar` control sequences are processed internally. The scope of the `\_namespace` command ends at the `\_endcode` command or when another `\_namespace` is used. This command checks if the same package label is not declared by the `\_namespace` twice.

The `\_public` macro does `\let\_foo = \_<pkg>_foo` when `\_namespace{<pkg>}` is declared. And the `\_private` macro does reverse operation to it. Example: you can define `\def\_macro{...}` and then set it to the user name space by `\_public \_macro;`.

If the package writer needs to declare a control sequence by `\_newif`, then there is an exception of the rule described above. Use `\_newifi\_if<pkg>_bar`, for example `\_newifi\_ifqr_incorner`. Then the control sequences `\_qr_incornertrue` and `\_qr_incornerfalse` can be used (or the sequences `\_incornertrue` and `\_incornerfalse` when `\_namespace{qr}` is used).

### 2.2.5 Summary about rules for external macro files published for OpTeX

If you are writting a macro file which is intended to be published for OpTeX, then you are greatly welcome. You should follow these rules:

- Don't use a control sequences from user name space in the macro bodies if there is not explicit and documented reason to do this.
- Don't declare control sequences in the user name space if there is not explicit and documented reason to do this.
- Use control sequences from OpTeX and primitive name space in read only mode if there is not explicit and documented reason to redefine them.
- Use `\_<pkg>_<name>` for your internal macros or `\_<name>` if the `\_namespace{<pkg>}` is declared. See section 2.2.4.
- Use `\_load` (or better: `\_load`) for loading more external macros if you need them. Don't use `\_input` explicitly in such cases. The reason is: the external macro file is not loaded twice if another macro or the user needs it explicitly too.
- Use `\_codedec1` as your first command in the macro file and `\_endcode` to close the text of macros.
- Use `\_doc ... \_cod` pairs for documenting the code pieces and/or write more documentation after the `\_endcode` command.

If the macro file accepts these recommendations then it should be named by `<filename>.opm` where `<filename>` differs from file names used directly in OpTeX and from other published macros. This extension `opm` has a precedence before `.tex` when the `\_load` macro is used.

The `qrcode.opm` is first example how an external macro file for OpTeX can look like.

---

<sup>1</sup> We have not adaptet the idea from expl3 language:)

## 2.2.6 The implementation of the name spaces

```
3 \_codedecl \public {Prefixing and code syntax <2020-02-14>} % preloaded in format
```

prefixed.opm

All TeX primitives have alternative control sequence `\_hbox` `\_string`, ...

prefixed.opm

```
9 \let\_directlua = \directlua
10 \_directlua {
11   % enable all TeX primitives with _ prefix
12   tex.enableprimitives('_', tex.extraprimatives('tex'))
13   % enable all primitives without prefixing
14   tex.enableprimitives('', tex.extraprimatives())
15   % enable all primitives with _ prefix
16   tex.enableprimitives('_', tex.extraprimatives())
17 }
```

`\_ea` is useful shortcut for `\_expandafter`. We recommend to use always the private form of `\_ea` because there is high probability that `\_ea` will be redefined by the user.

`\_public`  $\langle sequence \rangle$   $\langle sequence \rangle$  ... ; does `\_let` `\_` $\langle sequence \rangle$  = `\_` $\langle sequence \rangle$  for all sequences.

`\_private`  $\langle sequence \rangle$   $\langle sequence \rangle$  ... ; does `\_let` `\_` $\langle sequence \rangle$  = `\_` $\langle sequence \rangle$  for all sequences.

`\_xargs`  $\langle what \rangle$   $\langle sequence \rangle$   $\langle sequence \rangle$  ... ; does  $\langle what \rangle \langle sequence \rangle$  for each sequences.

prefixed.opm

```
34 \_let\_ea =\_expandafter % usefull shortcut
35
36 \_long\_def \_xargs #1#2{\_ifx #2;\_else \_ea#1\_ea#2\_ea\_xargs \_ea #1\_fi}
37
38 \_def \_pkglabel{}
39 \_def \_public {\_xargs \_publicA}
40 \_def \_publicA #1{\_ea\_let \_ea#1\_csname \_pkglabel \_csstring #1\_endcsname}
41
42 \_def \_private {\_xargs \_privateA}
43 \_def \_privateA #1{\_ea\_let \_csname \_pkglabel \_csstring #1\_endcsname =#1}
44
45 \_public \public \private \xargs \ea ;
```

Each macro file should begin with `\_codedecl` `\_macro`  $\{\langle info \rangle\}$ . If the `\_macro` is defined already then the `\_endinput` protects to read such file more than one times. Else the  $\langle info \rangle$  is printed to the terminal and the file is read.

The `\_endcode` is defined as `\_endinput` in the `optex.ini` file. `\_wterm`  $\{\langle text \rangle\}$  prints  $\langle text \rangle$  to the terminal and to the `.log` file (as in plain TeX).

prefixed.opm

```
57 \_def \_codedecl #1#2{%
58   \_ifx #1\_undefined \_wterm{#2}%
59   \_else \_expandafter \_endinput \_fi
60 }
61 \_def \_wterm {\_immediate \_write16 }
62
63 \_public \wterm ;
```

The `\_optexversion` and `\_fmtname` are defined in the `optex.ini` file. Maybe, somebody will need a private version of these macros.

prefixed.opm

```
70 \_private \optexversion \fmtname ;
```

The `\_mathsbon` and `\_mathsboff` are defined in `math-macros.opm` file. Now, we define the macro `\_namespace`  $\{\langle pkg label \rangle\}$  for package writers, see section 2.2.4.

prefixed.opm

```
78 \_def \_namespace #1{%
79   \_ifcsname namesp:#1\_endcsname \_errmessage
80     {The name space "#1" is used already, it cannot be used twice}%
81   \_endinput
82   \_else
83     \_ea \_gdef \_csname namesp:#1\_endcsname {}%
84     \_gdef \_pkglabel{#1}%
85     \_directlua{
86       callback.register("process_input_buffer",
87         function (str)
88           return string.gsub(str, "\_nbb[.]( [a-zA-Z])", "\_nbb _#1\_pcent 1")
89         }
90     }
```

```

89     end )
90 }%
91 \_gdef \_endcode {%
92     \_ifmathsb \_mathsbon \_else \_mthsboff \_fi
93     \_gdef \_pkglabel{#1}%
94     \_global \_let \_endcode=\_endinput
95     \_endinput }%
96 \_fi
97 }

```

## 2.3 pdfTeX initialization

Common pdfTeX primitives equivalents are declared here. Initial values are set.

luatex-ini.opm

```

3 \_codedecl \pdfprimitive {LuaTeX initialization code <2020-02-21>} % preloaded in format
4
5 \_let\pdfpagewidth \pagewidth
6 \_let\pdfpageheight \pageheight
7 \_let\pdfadjustspacing \adjustspacing
8 \_let\pdfprotrudechars \protrudechars
9 \_let\pdfnoligatures \ignoreligaturesinfont
10 \_let\pdffontexpand \expandglyphsinfont
11 \_let\pdfcopyfont \copyfont
12 \_let\pdfxform \saveboxresource
13 \_let\pdflastxform \lastsavedboxresourceindex
14 \_let\pdfrefxform \useboxresource
15 \_let\pdfximage \saveimageresource
16 \_let\pdflastximage \lastsavedimageresourceindex
17 \_let\pdflastximagepages \lastsavedimageresourcepages
18 \_let\pdfrefximage \useimageresource
19 \_let\pdfsavepos \savepos
20 \_let\pdflastxpos \lastxpos
21 \_let\pdflastypos \lastypos
22 \_let\pdfoutput \outputmode
23 \_let\pdfdraftmode \draftmode
24 \_let\pdfpxdimen \pxdimen
25 \_let\pdfinsertht \insertht
26 \_let\pdfnormaldeviate \normaldeviate
27 \_let\pdfuniformdeviate \uniformdeviate
28 \_let\pdfsetrandomseed \setrandomseed
29 \_let\pdfrandomseed \randomseed
30 \_let\pdfprimitive \primitive
31 \_let\ifpdfprimitive \ifprimitive
32 \_let\ifpdfabsnum \ifabsnum
33 \_let\ifpdfabsdim \ifabsdim
34
35 \_public
36 \pdfpagewidth \pdfpageheight \pdfadjustspacing \pdfprotrudechars
37 \pdfnoligatures \pdffontexpand \pdfcopyfont \pdfxform \pdflastxform
38 \pdfrefxform \pdfximage \pdflastximage \pdflastximagepages \pdfrefximage
39 \pdfsavepos \pdflastxpos \pdflastypos \pdfoutput \pdfdraftmode \pdfpxdimen
40 \pdfinsertht \pdfnormaldeviate \pdfuniformdeviate \pdfsetrandomseed
41 \pdfrandomseed \pdfprimitive \ifpdfprimitive \ifpdfabsnum \ifpdfabsdim ;
42
43 \_directlua {tex.enableprimitives('pdf',{'tracingfonts'})}
44
45 \_protected\_def \pdftexversion {\_numexpr 140\_relax}
46 \_def \pdftexrevision {7}
47 \_protected\_def \pdflastlink {\_numexpr\pdffeedback lastlink\_relax}
48 \_protected\_def \pdfretval {\_numexpr\pdffeedback retval\_relax}
49 \_protected\_def \pdflastobj {\_numexpr\pdffeedback lastobj\_relax}
50 \_protected\_def \pdflastannot {\_numexpr\pdffeedback lastannot\_relax}
51 \_def \pdfxformname {\_pdffeedback xformname}
52 {\_outputmode=1
53 \_xdef \pdfcreationdate {\_pdffeedback creationdate}
54 }
55 \_def \pdffontname {\_pdffeedback fontname}
56 \_def \pdffontobjnum {\_pdffeedback fontobjnum}

```

```

57         \def \pdffontsize      {\pdffeedback fontsize}
58         \def \pdfpageref       {\pdffeedback pageref}
59         \def \pdfcolorstackinit {\pdffeedback colorstackinit}
60 \protected\def \pdfliteral     {\pdfextension literal}
61 \protected\def \pdfcolorstack {\pdfextension colorstack}
62 \protected\def \pdfsetmatrix   {\pdfextension setmatrix}
63 \protected\def \pdfsave        {\pdfextension save\relax}
64 \protected\def \pdfrestore     {\pdfextension restore\relax}
65 \protected\def \pdfobj         {\pdfextension obj }
66 \protected\def \pdfrefobj      {\pdfextension refobj }
67 \protected\def \pdfannot       {\pdfextension annot }
68 \protected\def \pdfstartlink   {\pdfextension startlink }
69 \protected\def \pdfendlink     {\pdfextension endlink\relax}
70 \protected\def \pdfoutline     {\pdfextension outline }
71 \protected\def \pdfdest        {\pdfextension dest }
72 \protected\def \pdfthread      {\pdfextension thread }
73 \protected\def \pdfstartthread {\pdfextension startthread }
74 \protected\def \pdfendthread   {\pdfextension endthread\relax}
75 \protected\def \pdfinfo        {\pdfextension info }
76 \protected\def \pdfcatalog     {\pdfextension catalog }
77 \protected\def \pdfnames       {\pdfextension names }
78 \protected\def \pdfincludechars {\pdfextension includechars }
79 \protected\def \pdffontattr    {\pdfextension fontattr }
80 \protected\def \pdfmapfile     {\pdfextension mapfile }
81 \protected\def \pdfmapline     {\pdfextension mapline }
82 \protected\def \pdftrailer     {\pdfextension trailer }
83 \protected\def \pdfglyphtounicode {\pdfextension glyphtounicode }
84
85 \protected\edef \pdfcompresslevel {\pdfvariable compresslevel}
86 \protected\edef \pdfobjcompresslevel {\pdfvariable objcompresslevel}
87 \protected\edef \pdfdecimaldigits {\pdfvariable decimaldigits}
88 \protected\edef \pdfgamma {\pdfvariable gamma}
89 \protected\edef \pdfimageresolution {\pdfvariable imageresolution}
90 \protected\edef \pdfimageapplygamma {\pdfvariable imageapplygamma}
91 \protected\edef \pdfimagegamma {\pdfvariable imagegamma}
92 \protected\edef \pdfimagehicolor {\pdfvariable imagehicolor}
93 \protected\edef \pdfimageaddfilename {\pdfvariable imageaddfilename}
94 \protected\edef \pdfpkresolution {\pdfvariable pkresolution}
95 \protected\edef \pdfinclusioncopyfonts {\pdfvariable inclusioncopyfonts}
96 \protected\edef \pdfinclusionerrorlevel {\pdfvariable inclusionerrorlevel}
97 \protected\edef \pdfgentounicode {\pdfvariable gentounicode}
98 \protected\edef \pdfpagebox {\pdfvariable pagebox}
99 \protected\edef \pdfminorversion {\pdfvariable minorversion}
100 \protected\edef \pdfuniqueresname {\pdfvariable uniqueresname}
101 \protected\edef \pdfhorigin {\pdfvariable horigin}
102 \protected\edef \pdfvorigin {\pdfvariable vorigin}
103 \protected\edef \pdflinkmargin {\pdfvariable linkmargin}
104 \protected\edef \pdfdestmargin {\pdfvariable destmargin}
105 \protected\edef \pdfthreadmargin {\pdfvariable threadmargin}
106 \protected\edef \pdfpagesattr {\pdfvariable pagesattr}
107 \protected\edef \pdfpageattr {\pdfvariable pageattr}
108 \protected\edef \pdfpageresources {\pdfvariable pageresources}
109 \protected\edef \pdfxformattr {\pdfvariable xformattr}
110 \protected\edef \pdfxformresources {\pdfvariable xformresources}
111 \protected\edef \pdfpkmode {\pdfvariable pkmode}
112
113 \public
114 \pdftexversion \pdftexrevision \pdflastlink \pdfretval \pdflastobj
115 \pdflastannot \pdfxformname \pdfcreationdate \pdffontname \pdffontobjnum
116 \pdffontsize \pdfpageref \pdfcolorstackinit \pdfliteral \pdfcolorstack
117 \pdfsetmatrix \pdfsave \pdfrestore \pdfobj \pdfrefobj \pdfannot
118 \pdfstartlink \pdfendlink \pdfoutline \pdfdest \pdfthread \pdfstartthread
119 \pdfendthread \pdfinfo \pdfcatalog \pdfnames \pdfincludechars \pdffontattr
120 \pdfmapfile \pdfmapline \pdftrailer \pdfglyphtounicode \pdfcompresslevel
121 \pdfobjcompresslevel \pdfdecimaldigits \pdfgamma \pdfimageresolution
122 \pdfimageapplygamma \pdfimagegamma \pdfimagehicolor \pdfimageaddfilename
123 \pdfpkresolution \pdfinclusioncopyfonts \pdfinclusionerrorlevel
124 \pdfgentounicode \pdfpagebox \pdfminorversion \pdfuniqueresname \pdfhorigin
125 \pdfvorigin \pdflinkmargin \pdfdestmargin \pdfthreadmargin \pdfpagesattr

```

```

126 \pdfpageattr \pdfpageresources \pdfxformattr \pdfxformresources \pdfpkmode ;
127
128 \pdfminorversion = 5
129 \pdfobjcompresslevel = 2
130 \pdfcompresslevel = 9
131 \pdfdecimaldigits = 3
132 \pdfpkresolution = 600

```

## 2.4 Basic macros

We define first bundle of basic macros.

```

3 \codedecl \sdef {Basic macros for OpTeX <2020-02-14>} % loaded in format

```

basic-macros.opm

`\bgroup`, `\egroup`, `\empty`, `\space`, `\null` and `\wlog` are classical macros from plain T<sub>E</sub>X.

```

10 \let\bgroup={ \let\egroup=}
11 \def \empty {}
12 \def \space { }
13 \def \null {\hbox{}}
14 \def \wlog {\immediate\write-1 } % write on log file (only)
15
16 \public \bgroup \egroup \empty \space \null \wlog ;

```

basic-macros.opm

`\bslash` is “normal backslash” with category code 12. `\nbb` and `\pcent` are double backslash and normal %, they should be used in lua codes, for example.

```

24 \edef \bslash {\csstring\}
25 \edef \nbb {\bslash\bslash}
26 \edef \pcent{\csstring\%}
27
28 \public \bslash \nbb \pcent ;

```

basic-macros.opm

`\sdef {<text>}` is equivalent to `\def\<text>`, where `\<text>` is a control sequence. You can use arbitrary parameter mask after `\sdef{<text>}`, don’t put the (unwanted) space immediately after closing brace }.

`\sxdef {<text>}` is equivalent to `\xdef\<text>`.

`\slet {<textA>}{<textB>}` is equivalent to `\let \<textA> = \<textB>`.

```

40 \def \sdef #1{\_ea\_def \_csname#1\_endcsname}
41 \def \sxdef #1{\_ea\_xdef \_csname#1\_endcsname}
42 \def \slet #1#2{\_ea\_let \_csname#1\_ea\_endcsname \_csname#2\_endcsname}
43
44 \public \sdef \sxdef \slet ;

```

basic-macros.opm

`\adef {<char>}{<body>}` puts the `<char>` as active character and defines it as `{<body>}`. You can declare a macro with parameters too. For example `\adef @#1{...$1...}`.

```

52 \def \adef #1{\_catcode`#1=13 \begingroup \lccode\~=`#1\_lowercase{\endgroup\_def~}
53 \public \adef ;

```

basic-macros.opm

`\cs {<text>}` is only a shortcut to `\csname <text>\endcsname`, but you need one more `\_ea` if you need to get the real control sequence `\<text>`.

`\trycs {<csname>}{<text>}` expands to `\<csname>` if it is defined else to the `<text>`.

```

63 \def \cs #1{\_csname#1\_endcsname}
64 \def \trycs#1#2{\_ifcsname #1\_endcsname \_csname #1\_endcsname \_else #2\_fi}
65 \public \cs \trycs ;

```

basic-macros.opm

`\addto \macro{<text>}` adds `<text>` to your `\macro`, which must be defined.

```

71 \long\_def \addto #1#2{\_ea\_def\_ea#1\_ea{#1#2}}
72 \public \addto ;

```

basic-macros.opm

`\opwarning {<text>}` prints warning on the terminal and to the log file.

```

78 \def \opwarning #1{\_wterm{WARNING: #1.}}
79 \public \opwarning ;

```

basic-macros.opm

`\loggingall` and `\tracingall` are defined similarly as in plain TeX, but they print more logging information to the log file and to the terminal.

basic-macros.opm

```

87 \def\loggingall{\tracingcommands=3 \tracingstats=2 \tracingpages=1
88 \tracingoutput=1 \tracinglostchars=1 \tracingmacros=2
89 \tracingparagraphs=1 \tracingrestores=1 \tracingscantokens=1
90 \tracingifs=1 \tracinggroups=1 \tracingassigns=1 }
91 \def\tracingall{\tracingonline=1 \loggingall}
92
93 \public \loggingall \tracingall ;

```

## 2.5 Allocators for TeX registers

Like plain TeX, the allocators `\newcount`, `\newwrite`, etc. are defined. The registers are allocated from 256 to the `\mai<type>` which is 65535 in LuaTeX.

Unlike in Plain TeX, the mentioned allocators are not `\outer`.

User can use `\dimen0` to `\dimen200` and similarly for `\skip`, `\muskip`, `\box` and `\toks` directly. User can use `\count20` to `\count200` directly too. This is the same philosophy like in old plain TeX, but the range of directly used registers is wider.

Inserts are allocated from 254 to 201 using `\newinsert`.

You can define your own allocation concept (for example for allocation of arrays) from top of registers array. The example shows a definition of the array-like declarator of counters.

```

\newcount \_maicount    % redefine maximal allocation index as variable
\_maicount = \maicount  % first value is top of the array

\def\newcountarray #1[#2]{% \newcountarray \foo[100]
  \advance\_maicount by -#2\relax
  \ifnum \_countalloc > \_maicount
    \errmessage{No room for a new array of \string\count}%
  \else
    \chardef#1=\_maicount
  \fi
}
\def\usecount #1[#2]{% \usecount \foo[2]
  \count\numexpr#1+#2\relax
}

```

alloc.opm

```

3 \codedecl \newdimen {Allocators for registers <2020-05-12>} % loaded in format

```

The limits are set first.

alloc.opm

```

9 \chardef\_maicount = 65535    % Max Allocation Index for counts registers in LuaTeX
10 \_let\_maidimen = \_maicount
11 \_let\_maiskip = \_maicount
12 \_let\_maimuskip = \_maicount
13 \_let\_maibox = \_maicount
14 \_let\_maitoks = \_maicount
15 \chardef\_mairead = 15
16 \chardef\_maiwrite = 15
17 \chardef\_maifam = 255

```

Each allocation macro needs its own counter.

alloc.opm

```

23 \countdef\_countalloc=10 \_countalloc=255
24 \countdef\_dimenalloc=11 \_dimenalloc=255
25 \countdef\_skipalloc=12 \_skipalloc=255
26 \countdef\_muskipalloc=13 \_muskipalloc=255
27 \countdef\_boxalloc=14 \_boxalloc=255
28 \countdef\_toksalloc=15 \_toksalloc=255
29 \countdef\_readalloc=16 \_readalloc=-1
30 \countdef\_writealloc=17 \_writealloc=-1
31 \countdef\_famalloc=18 \_famalloc=3

```



The common allocation macro `\_allocator`  $\langle sequence \rangle \{ \langle type \rangle \} \langle primitive declarator \rangle$  is defined. This idea was used in classical plain  $\text{\TeX}$  by Donald Knuth too but the macro from plain  $\text{\TeX}$  seems to be more complicated:).

```
alloc.opm
```

```

41 \_def\_allocator #1#2#3{%
42   \_global\_advance\_cs{#2alloc}by1
43   \_ifnum\_cs{#2alloc}>\_cs{\_mai#2}%
44     \_errmessage{No room for a new \_ea\_string\_csname #2\_endcsname}%
45   \_else
46     \_global#3#1=\_cs{#2alloc}%
47     \_wlog{\_string#1=\_ea\_string\_csname #2\_endcsname\_the\_cs{#2alloc}}%
48   \_fi
49 }
```

The allocation macros `\newcount`, `\newdimen`, `\newskip`, `\newmuskip`, `\newbox`, `\newtoks`, `\newread`, `\newwrite` and `\newfam` are defined here.

```
alloc.opm
```

```

58 \_def\_newcount #1{\_allocator #1{count}\_countdef}
59 \_def\_newdimen #1{\_allocator #1{dimen}\_dimendef}
60 \_def\_newskip #1{\_allocator #1{skip}\_skipdef}
61 \_def\_newmuskip #1{\_allocator #1{muskip}\_muskipdef}
62 \_def\_newbox #1{\_allocator #1{box}\_chardef}
63 \_def\_newtoks #1{\_allocator #1{toks}\_toksdef}
64 \_def\_newread #1{\_allocator #1{read}\_chardef}
65 \_def\_newwrite #1{\_allocator #1{write}\_chardef}
66 \_def\_newfam #1{\_allocator #1{fam}\_chardef}
67
68 \_public \newcount \newdimen \newskip \newmuskip \newbox \newtoks \newread \newwrite \newfam ;
```

The `\newinsert` macro is defined differently than others.

```
alloc.opm
```

```

74 \_newcount\_insertalloc \_insertalloc=255
75 \_chardef\_insertmin = 201
76
77 \_def\_newinsert #1{%
78   \_advance\_insertalloc by-1
79   \_ifnum\_insertalloc <\_insertmin
80     \_errmessage {No room for a new \_string\_insert}%
81   \_else
82     \_global\_chardef#1=\_insertalloc
83     \_wlog {\_string#1=\_string\_insert\_the\_insertalloc}%
84   \_fi
85 }
86 \_public \newinsert ;
```

Other allocation macros `\newattribute` and `\newcatodetable` have their counter allocated by the `\newcount` macro.

```
alloc.opm
```

```

93 \_newcount \_attributealloc \_attributealloc=0
94 \_chardef\_maiaattribute=\_maicount
95 \_def\_newattribute #1{\_allocator #1{attribute}\_attributedef}
96
97 \_newcount \_catcodetablealloc \_catcodetablealloc=10
98 \_chardef\_maicatcodetable=32767
99 \_def\_newcatcodetable #1{\_allocator #1{catcodetable}\_chardef}
100
101 \_public \newattribute \newcatcodetable ;
```

We declare public and private versions of `\tmpnum` and `\tmpdim` registers separately. They are independent registers.

```
alloc.opm
```

```

108 \_newcount \tmpnum \_newcount \_tmpnum
109 \_newdimen \tmpdim \_newdimen \_tmpdim
```

A few registers are initialized like in plain $\text{\TeX}$ . Note that `\z@` and `\z@skip` from plain $\text{\TeX}$  is `\zo` and `\zoskip` because we absolutely don't support the `@` category dance. Note that `\p@` is not defined because we can write `1pt` which is more legible.

```
alloc.opm
```

```

119 \_newdimen\_maxdimen \_maxdimen=16383.99999pt % the largest legal <dimen>
```

```

120 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
121 \newskip\centering \centering=0pt plus 1000pt minus 1000pt
122 \newskip\zskip \zskip=0pt plus 0pt minus 0pt
123 \newbox\voidbox % permanently void box register
124
125 \public \maxdimen \hideskip \centering \zskip \voidbox ;

```

## 2.6 If-macros, loops, is-macros

```

3 \codedecl \newif {Special if-macros, is-macros and loops <2020-05-06>} % preloaded in format

```

### 2.6.1 Classical \newif

The `\newif` macro implements boolean value. It works as in plain TeX. It means that after `\newif\ifxxx` you can use `\xxxtrue` or `\xxxfalse` to set the boolean value and use `\ifxxx true\else false\fi` to test this value. The default value is false.

The macro `\_newif` enables to declare `\_ifxxx` and to use `\_xxxtrue` and `\_xxxfalse`. This means that it is usable for internal name space (`_`-prefixed macros).

```

18 \def\_newif #1{\_ea\_newifA \string #1\_relax#1}
19 \_ea\_def \_ea\_newifA \string\if #1\_relax#2{%
20   \sdef{#1true}{\_let#2=\_iftrue}%
21   \sdef{#1false}{\_let#2=\_iffalse}%
22   \_let#2=\_iffalse
23 }
24 \def\_newifi #1{\_ea\_newifiA \string#1\_relax#1}
25 \_ea\_def \_ea\_newifiA \string\if #1\_relax#2{%
26   \sdef{#1true}{\_let#2=\_iftrue}%
27   \sdef{#1false}{\_let#2=\_iffalse}%
28   \_let#2=\_iffalse
29 }
30 \public \newif ;

```

### 2.6.2 Loops

The `\loop` `<codeA>` `\ifsomething` `<codeB>` `\repeat` loops `<codeA>``<codeB>` until `\ifsomething` is false. Then `<codeB>` is not executed and loop is finished. This works like in plain TeX, but implementation is somewhat better (you can use `\else` clause after the `\ifsomething`).

There are public version `\loop... \repeat` and private version `\_loop... \_repeat`. You cannot mix both versions in one loop.

The `\loop` macro keeps its original plain TeX meaning. It is not expandable and nested `\loops` are possible only in a TeX group.

```

46 \long\_def \_loop #1\_repeat{\_def\_body{#1}\_iterate}
47 \_def \_loop #1\_repeat{\_def\_body{#1}\_iterate}
48 \_let \_repeat=\_fi % this makes \loop... \if... \repeat skippable
49 \_let \_repeat=\_fi
50 \_def \_iterate {\_body \_ea \_iterate \_fi}

```

`\foreach` `<list>` `\do` `{<what>}` repeats `<what>` for each element of the `<list>`. The `<what>` can include `#1` which is substituted by each element of the `<list>`. The macro is expandable.

`\fornum` `<from>..<to>` `\do` `{<what>}` or `\fornumstep` `<num>`: `<from>..<to>` `\do` `{<what>}` repeats `<what>` for each number from `<from>` to `<to>` (with step `<num>` or with step one). The `<what>` can include `#1` which is substituted by current number. The sequence `<from>..<to>` can be decreasing too. The macro is expandable.

Recommendation: it is better to use private variants of `\_foreach` and `\_fornum`. When the user writes `\input tikz` then `\foreach` macro is redefined! The private variants use `\_do` separator instead `\do` separator.

```

71 \newcount\_frnum % the numeric variable used in \fornum
72 \_def\_do{\_doundefined} % we need to ask \_ifx#1\_do ...
73
74 \long\_def\_foreach #1\_do#2{\_putforstack

```

```

75 \_immediateassignment\_def\_fbody##1{#2}%
76 \_foreachA #1\_do}
77 \_long\_def\_foreachA #1{\_ifx\_do#1\_getforstack\_else\_fbody{#1}\_ea\_foreachA\_fi}
78
79 \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
80 \_long\_def\_fornumstep#1:#2..#3\_do#4{\_putforstack
81 \_immediateassigned{%
82 \_def\_fbody##1{#4}%
83 \_def\_fornumsgn{}}%
84 \_def\_fornumrel{<}}%
85 \_frnum=\_numexpr#2\_relax
86 \_ifnum\_numexpr#3<\_frnum \_def\_fornumrel{>}\_fi %decreasing sequence
87 \_ifnum\_numexpr#1\_fornumrel0 \_def\_fornumsgn{-}\_fi % correction
88 }%
89 \_fornumB{#3}{#1}%
90 }
91 \_def\_fornumB #1#2{\_ifnum\_numexpr#1\_fornumrel\_frnum \_getforstack \_else
92 \_ea\_fbody\_ea{\_the\_frnum}%
93 \_immediateassignment\_advance\_frnum by\_numexpr\_fornumsgn#2\_relax
94 \_afterfi{\_fornumB{#1}{#2}}\_fi
95 }
96 \_def\_afterfi#1#2\_fi{\_fi#1}
97
98 \_def\_foreach #1\_do{\_foreach #1\_do}
99 \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
100 \_def\_fornumstep#1:#2..#3\_do{\_fornumstep #1:#2..#3\_do}

```

The `\foreach` and `\fornum` macros can be nested and arbitrary combined. When they are nested then use `##1` for the variable of nested level, `###1` for the variable of second nested level etc. Example:

```
\foreach ABC \do {\fornum 1..5 \do {letter:#1, number: ##1. }}
```

Implementation note: we cannot use  $\TeX$ -groups for nesting levels because we want to do the macros expandable. We must implement a special for-stack which saves the data needed by `\foreach` and `\fornum`. The `\_putforstack` is used when `\for*` is initialized and `\_getforstack` is used when the `\for*` macro ends. The `\_forlevel` variable keeps the current nesting level. If it is zero, then we need not save nor restore any data.

if-macros.opm

```

118 \_newcount\_forlevel
119 \_def\_putforstack{\_immediateassigned{%
120 \_ifnum\_forlevel>0
121 \_sdef\_frnum:\_the\_forlevel\_ea}\_ea{\_the\_frnum}%
122 \_slet{fbody:\_the\_forlevel}{fbody}%
123 \_fi
124 \_advance\_forlevel by1
125 }}
126 \_def\_getforstack{\_immediateassigned{%
127 \_advance\_forlevel by-1
128 \_ifnum\_forlevel>0
129 \_slet{fbody}{fbody:\_the\_forlevel}%
130 \_frnum=\_cs{frnum:\_the\_forlevel}\_space
131 \_fi
132 }}

```

### 2.6.3 Is-macros

There are a collection of macros `\isempty`, `\istoksemt`, `\isequal`, `\ismacro`, `\isdefined`, `\isinlist` and `\isfile` with common syntax:

```

\_issomething <params> \iftrue <codeA> \else <codeB> \fi
or
\_issomething <params> \iffalse <codeB> \else <codeA> \fi

```

The `\else` part is optional. The `<codeA>` is processed if `\_issomething<params>` generates true condition. The `<codeB>` is processed if `\_issomething<params>` generates false condition.

The `\iftrue` or `\iffalse` is an integral part of this syntax because we need to keep skippable nested `\if` conditions.

Implementation note: we read this `\iftrue` or `\iffalse` into unseparated parameter and repeat it because we need to remove an optional space before this command.

`\isempty`  $\langle text \rangle$  `\iftrue` is true if the  $\langle text \rangle$  is empty. This macro is expandable.

`\istokseempty`  $\langle tokens\ variable \rangle$  `\iftrue` is true if the  $\langle tokens\ variable \rangle$  is empty. It is expandable.

if-macros.opm

```
162 \def \isempty #1#2{\if\relax\detokenize{#1}\relax \else \ea\unless \fi#2}
163 \def \istokseempty #1#2{\ea\isempty\ea{\the#1}#2}
164 \public \isempty \istokseempty ;
```

`\isequal`  $\langle textA \rangle$   $\langle textB \rangle$  `\iftrue` is true if the  $\langle textA \rangle$  and  $\langle textB \rangle$  are equal, only from strings point of view, category codes are ignored. The macro is expandable.

if-macros.opm

```
173 \def\isequal#1#2#3{\directlua{%
174   if "\luaescapestring{\detokenize{#1}}"=="\luaescapestring{\detokenize{#2}}"
175   then else tex.print("\nbb unless") end}#3}
176 \public \isequal ;
```

`\ismacro`  $\langle macro \{ text \} \rangle$  `\iftrue` is true if macro is defined as  $\langle text \rangle$ . Category codes are ignored in this testing. The macro is expandable.

if-macros.opm

```
183 \def\ismacro#1{\ea\isequal\ea{#1}}
184 \public \ismacro ;
```

`\isdefined`  $\langle csname \rangle$  `\iftrue` is true if  $\langle csname \rangle$  is defined. The macro is expandable.

if-macros.opm

```
191 \def\isdefined #1#2{\ifcsname #1\endcsname \else \ea\unless \fi #2}
192 \public \isdefined ;
```

`\isinlist`  $\langle list \{ \langle text \rangle \} \rangle$  `\iftrue` is true if the  $\langle text \rangle$  is included the macro body of the `\list`. The category code are relevant here. The macro is not expandable.

if-macros.opm

```
200 \long\def\isinlist#1#2{\begingroup
201   \long\def\tmp##1#2##2\end/_%
202   {\endgroup\if\relax\detokenize{##2}\relax \ea\unless\fi}%
203   \ea\tmp#1\endlistsep#2\end/_%
204 }
205 \public \isinlist ;
```

`\isfile`  $\langle filename \rangle$  `\iftrue` is true if the file  $\langle filename \rangle$  exists and are readable by T<sub>E</sub>X.

if-macros.opm

```
212 \newread \_testin
213 \def\isfile #1{%
214   \openin\_testin ={#1}\relax
215   \ifeof\_testin \ea\unless
216   \else \closein\_testin
217   \fi
218 }
219 \public \isfile ;
```

`\isfont`  $\langle fontname\ or\ [fontfile] \rangle$  `\iftrue` is true if given font exists. The result of this testing is saved to the `\_ifexistfam`.

if-macros.opm

```
227 \newif \_ifexistfam
228 \def\isfont#1#2{%
229   \begingroup
230     \suppressfontnotfounderror=1
231     \font\_testfont={#1}\relax
232     \ifx\_testfont\_nullfont \def\tmp{\_existfamfalse \unless}
233     \else \def\tmp{\_existfamtrue}\fi
234   \ea \endgroup \tmp #2%
235 }
236 \public \isfont ;
```

The last macro `\isnextchar`  $\langle char \rangle$   $\langle codeA \rangle$   $\langle codeB \rangle$  has different syntax than all others is-macros. It executes  $\langle codeA \rangle$  if next character is equal to  $\langle char \rangle$ . Else the  $\langle codeB \rangle$  is executed. The macro is not expandable.

if-macros.opm

```
245 \long\def\isnextchar#1#2#3{\begingroup\_toks0={\_endgroup#2}\_toks1={\_endgroup#3}%
```

```

246 \let\tmp= #1\futurelet\next\isnextcharA
247 }
248 \def\isnextcharA{\the\toks\ifx\tmp\next0\else1\fi\space}
249
250 \public \isnextchar ;

```

## 2.7 Setting parameters

The behavior of document processing by OpTeX is controlled by *parameters*. The parameters are

- primitive registers used in build-in algorithms of T<sub>E</sub>X,
- registers declared and used by OpTeX macros.

Both groups of registers have their type: number, dimension, skip, token list.

The registers are represented by their names (control sequences). If the user re-defines such control sequence then the appropriate register exists steadily and build-in algorithms are using it without change. But user cannot access its value in such case. OpTeX declares two control sequences for each register: prefixed and unprefixed. OpTeX macros use only prefixed variants of control sequences. The user should use unprefixed variant with the same meaning and set or read values of registers using the unprefixed variant. If the user re-defines the unprefixed control sequence of a register then OpTeX macros still work without change.

```

3 \codeldecl \normalbaselineskip {Parameter settings <2020-03-17>} % preloaded in format

```

parameters.opm

### 2.7.1 Primitive registers

The primitive registers with the same default value as in plain T<sub>E</sub>X follow:

```

10 \parindent=20pt      % indentation of paragraphs
11 \pretolerance=100    % parameters used in paragraph breaking algorithm
12 \tolerance=200
13 \hbadness=1000
14 \vbadness=1000
15 \doublehyphendemerits=10000
16 \finalhyphendemerits=5000
17 \adjdemerits=10000
18 \uchyph=1
19 \defaultthyphenchar=-
20 \defaultskewchar=-1
21 \hfuzz=0.1pt
22 \vfuzz=0.1pt
23 \overfullrule=5pt
24 \linepenalty=10      % penalty between lines inside the paragraph
25 \hyphenpenalty=50    % when a word is broken
26 \exhyphenpenalty=50 % when the hyphenmark is used explicitly
27 \binoppenalty=700    % between binary operators in math
28 \relpenalty=500      % between relations in math
29 \brokenpenalty=100   % after lines if they end by a broken word.
30 \displaywidowpenalty=50 % before last line of paragraph if display math follows
31 \predisplaypenalty=10000 % above display math
32 \postdisplaypenalty=0 % below display math
33 \delimiterfactor=901 % parameter for scaling delimiters
34 \delimitershortfall=5pt
35 \nulldelimiterspace=1.2pt
36 \scriptspace=0.5pt
37 \maxdepth=4pt
38 \splitmaxdepth=\maxdimen
39 \boxmaxdepth=\maxdimen
40 \parskip=0pt plus 1pt
41 \abovedisplayskip=12pt plus 3pt minus 9pt
42 \abovedisplayshortskip=0pt plus 3pt
43 \belowdisplayskip=12pt plus 3pt minus 9pt
44 \belowdisplayshortskip=7pt plus 3pt minus 4pt
45 \parfillskip=0pt plus 1fil
46 \thinmuskip=3mu
47 \medmuskip=4mu plus 2mu minus 4mu

```

parameters.opm

```
48 \_thickmuskip=5mu plus 5mu
```

Note that `\topskip` and `\splittopskip` are changed when first `\typosize` sets the main values (default font size and default `\baselineskip`).

parameters.opm

```
56 \_topskip=10pt      % top edge of page-box to first baseline distance
57 \_splittopskip=10pt
```

## 2.7.2 Plain T<sub>E</sub>X registers

Declared registers used in plain T<sub>E</sub>X

parameters.opm

```
64 % We also define special registers that function like parameters:
65 \_newskip\_smallskipamount \_smallskipamount=3pt plus 1pt minus 1pt
66 \_newskip\_medskipamount \_medskipamount=6pt plus 2pt minus 2pt
67 \_newskip\_bigskipamount \_bigskipamount=12pt plus 4pt minus 4pt
68 \_newskip\_normalbaselineskip \_normalbaselineskip=12pt
69 \_newskip\_normallineskip \_normallineskip=1pt
70 \_newdimen\_normallineskiplimit \_normallineskiplimit=0pt
71 \_newdimen\_jot \_jot=3pt
72 \_newcount\_interdisplaylinepenalty \_interdisplaylinepenalty=100
73 \_newcount\_interfootnotelinepenalty \_interfootnotelinepenalty=100
74
75 \_def\_normalbaselines{\_lineskip=\_normallineskip
76   \_baselineskip=\_normalbaselineskip \_lineskiplimit=\_normallineskiplimit}
77
78 \_def\_frenchspacing{\_sfcode\`.=1000 \_sfcode`\?=1000 \_sfcode`\!=1000
79   \_sfcode\`:=1000 \_sfcode\`:=1000 \_sfcode\`=1000 }
80 \_def\_nonfrenchspacing{\_sfcode\`.=3000 \_sfcode`\?=3000 \_sfcode`\!=3000
81   \_sfcode\`:=2000 \_sfcode\`:=1500 \_sfcode\`=1250 }
82
83 \_public \normalbaselines \frenchspacing \nonfrenchspacing
84 \smallskipamount \medskipamount \bigskipamount
85 \normalbaselineskip \normallineskip \normallineskiplimit
86 \jot \interdisplaylinepenalty \interfootnotelinepenalty ;
```

## 2.7.3 Different settings than in plain T<sub>E</sub>X

Default “baseline setting” is for 10 pt fonts (like in plain T<sub>E</sub>X). But `\typosize` and `\typoscale` macros re-declare it if another font size is used.

The `\nonfrenchspacing` is not set by default because the author of OpT<sub>E</sub>X is living in the Europe. If you set `\enlang` hyphenation patterns then `\nonfrenchspacing` is set.

parameters.opm

```
100 \_normalbaselines % baseline setting, 10 pt font size
```

Different values than in plain T<sub>E</sub>X have following primitive registers. We prohibit orphans, set more information for tracing boxes, set page origin to upper left corner of the paper (no at 1 in, 1 in coordinates) and set default page dimensions as A4, no letter.

parameters.opm

```
109 \_emergencystretch=20pt % we want to use third pass of a paragraph building algorithmh
110                          % we need not to keep the compatibility with old documents
111
112 \_clubpenalty=10000      % after first line of paragraph
113 \_widowpenalty=10000    % before last line of paragraph
114
115 \_showboxbreadth=150    % for tracing boxes
116 \_showboxdepth=7
117 \_errorcontextlines=15
118 \_tracinglostchars=2    % missing chracter warnings on terminal too
119
120 \_outputmode=1          % PDF ouput
121 \_pdfvorigin=0pt        % orgin is exatly at left upper corner
122 \_pdfhorigin=0pt
123 \_hoffset=25mm         % margins are 2.5cm, no 1in
124 \_voffset=25mm
125 \_hsize=160mm           % 210mm (from A4 size) - 2*25mm (default margins)
126 \_vsize=244mm           % 297mm (from A4 size) - 2*25mm (default margins) -3mm baseline correction
127 \_pagewidth=210 true mm
```

```
128 \_pageheight=297 true mm
```

If you insist on plain T<sub>E</sub>X values of these parameters then you can call the `\plaintexsetting` macro.

parameters.opm

```
135 \_def\_plaintexsetting{%
136   \_emergencystretch=0pt
137   \_clubpenalty=150
138   \_widowpenalty=150
139   \_pdfvorigin=1in
140   \_pdfhorigin=1in
141   \_hoffset=0pt
142   \_voffset=0pt
143   \_hsize=6.5in
144   \_vsize=8.9in
145   \_pagewidth=8.5 true in
146   \_pageheight=11 true in
147   \_nonfrenchspacing
148 }
149 \_public \plaintexsetting ;
```

## 2.7.4 OpT<sub>E</sub>X parameters

The main principle how to configure OpT<sub>E</sub>X is not to use only parameters. A designer can copy macros from OpT<sub>E</sub>X and re-define them as required. This is a reason why we don't implement dozens of parameters, but we keep OpT<sub>E</sub>X macros relatively simple. Example: do you want another design of section titles? Copy macros `\_printsec` and `\_printsecc` from `sections.opm` file to your macro file and re-define them.

Notice for OPmac users: there is important difference: all "string-like" parameters are token lists in OpT<sub>E</sub>X (OPmac uses macros for them). The reason of this difference: if user sets parameter by unprotected control sequence, an OpT<sub>E</sub>X macro can read *the same data* using protected control sequence. If user re-defines such unprotected control sequence (because he/she does know about it) then nothing bad happens.

The `\picdir` tokens list can include a directory where image files (loaded by `\inspic`) are saved. Empty `\picdir` (default value) means that image files are in the current directory (or somewhere in the T<sub>E</sub>X system where LuaT<sub>E</sub>X is able to find them). If you set non-empty value to the `\picdir`, then it must end by / character, for example `\picdir={img/}` means that there exists a directory `img` in your current directory and the image files are stored here.

parameters.opm

```
177 \_newtoks\_picdir
178 \_public \picdir ;
```

You can control the dimensions of included images by the parameters `\picwidth` (which is equivalent to `\picw`) and `\picheight`. By default these parameters are set to zero: the native dimension of the image is used. If only `\picwidth` has a nonzero value, then this is the width of the image (height is calculated automatically in order to respect the aspect of the image). If only `\picheight` has a nonzero value then height is given, width is calculated. If both parameters are non-zero, the height and width are given and the aspect ratio of the image is (probably) broken. We recommend to set these parameters locally in the group where `\inspic` is used in order to not influence the dimensions of another images. But there exist many situations you need to put the same dimensions to more images, so you can set this parameter only once before more `\inspic` macros.

parameters.opm

```
196 \_newdimen\_picwidth \_picwidth=0pt \_let\picw=\_picwidth
197 \_newdimen\_picheight \_picheight=0pt
198 \_public \picwidth \picheight ;
```

The `\everytt` is token list used in `\begtt... \endtt` environment and in the verbatim group opened by `\verbatiminput` macro. You can include a code which is processed inside the group after basic settings were done. On the other hand, it is processed before scanner of verbatim text is started. Your macros should influence scanner (catcode settings) or printing process of the verbatim code or both.

The code from the line immediately after `\begtt` is processed after the `\everytt`. This code should overwrite `\everytt` settings. Use `\everytt` for all verbatim environments in your document and use a code after `\begtt` locally only for this environment.

The `\everyintt` token list does similar work but acts in the in-line verbatim text processed by a pair of `\activettchar` characters or by `\code{<text>}`. You can set `\everyintt={\Red}` for example if you want in-line verbatim in red color.



```

221 \_newtoks\_everytt
222 \_newtoks\_everyintt
223 \_public \everytt \everyintt ;

```

The `\ttline` is used in `\begtt...\endtt` environment or in the code printed by `\verbinp`. If `\ttline` is positive or zero, then the verbatim code have numbered lines from `\ttline+1`. The `\ttline` register is re-set to new value after a code piece is printed, so next code pieces have numbered lines continuously. If `\ttline=-1`, then `\begtt...\endtt` lines are without numbers and `\verbinp` lines shows the line numbers of inputted file. If `\ttline<-1` then no line numbers are printed.

```

237 \_newcount\_ttline \_ttline=-1 % last line number in \begtt...\endtt
238 \_public \ttline ;

```

The `\ttindent` gives default indentation of verbatim lines printed by `\begtt...\endtt` pair or by `\verbinp`.

The `\ttshift` gives the amount of shift of all verbatim lines to right. Despite to the `\ttindent`, it does not shift the line numbers, only the text.

The `\iindent` gives default indentations used in table of contents, captions, lists, bib references, It is strongly recommended to re-set this value if you set `\parindent` to another value than plain TeX default 20pt. A well typeset document should have the same dimension for all indentations, so you should say `\ttindent=\parindent` and `\iindent=\parindent`.

```

258 \_newdimen\_ttindent \_ttindent=\_parindent % indentation in verbatim
259 \_newdimen\_ttshift
260 \_newdimen\_iindent \_iindent=\_parindent
261 \_public \ttindent \ttshift \iindent ;

```

The tabelator `^^I` has its category code like space: it behaves as a space in normal text. This is normal plain TeX setting. But in the multiline verbatim environment it is active and expands to the `\hskip<dimen>` where `<dimen>` is the width of `\tabspaces` spaces. Default `\tabspaces=3` means that tabelator behaves like three spaces in multiline verbatim.

```

273 \_newcount \_tabspaces \_tabspaces=3
274 \_public \tabspaces ;

```

If `\hicolors` is non-empty then its contents is used instead `\_hicolors<name>` declared in the file `hisyntax-<name>.opm`. The user can give his/her preferences about colors for syntax highlighting by this tokens list. Full color set must be declared here.

```

284 \_newtoks\_hicolors
285 \_public \hicolors ;

```

The default item mark used between `\begitems` and `\enditems` is bullet. The `\defaultitem` tokens list declare this default item mark.

The `\everyitem` tokens list is applied in vertical mode at the start of each item.

The `\everylist` tokens list is applied after group is opened by

The `\ilevel` keeps the value of current nesting level of the items list.

The `\listskipamount` gives vertical skip above and below the items list if `\ilevel=1`.

```

302 \_newtoks\_defaultitem \_defaultitem={$_bullet$_enspace}
303 \_newtoks\_everyitem
304 \_newtoks\_everylist
305 \_newskip \_listskipamount \_listskipamount=\_medskipamount
306 \_newcount \_ilevel
307 \_public \defaultitem \everyitem \everylist \listskipamount \ilevel ;

```

The `\tit` macro includes `\vglue\titskip` above the title of the document.

```

313 \_newskip\_titskip \_titskip=40pt \_relax % \vglue above title printed by \tit
314 \_public \titskip ;

```

The `\begmulti \endmulti` pair creates more columns. The parameter `\colsep` declares the space between columns. If  $n$  columns are specified then we have  $n - 1$  `\colseps` and  $n$  columns in total `\hsize`. This gives definite result of columns width.

```

323 \_newdimen\_colsep \_colsep=20pt % space between columns
324 \_public \colsep ;

```

Each line in the Table of contents is printed in a group. The `\everytocline` tokens list is processed here before the internal `\_toc1:<num>` macro which starts printing the line.

```
332 \_newtoks \_everytocline
333 \_public \_everytocline ;
```

parameters.opm

The `\bibtexhook` tokens list is used inside the group when `\usebib` command is processed after style file is loaded and before printing bib-entries. You can re-define a behavior of style file here or you can modify the more declaration for printing (fonts, baselineskip, etc.) or you can define a specific macros used in your .bib file.

```
343 \_newtoks\_\bibtexhook
344 \_public \_\bibtexhook ;
345
346 \_newtoks\_\everycaptiont \_newtoks\_\everycaptionf
347 \_public \_\everycaptiont \_\everycaptionf ;
```

parameters.opm

The `\everyii` tokens list is used before `\noindent` for each Index item when printing the Index.

```
354 \_newtoks\_\everyii
355 \_public \_\everyii ;
```

parameters.opm

The `\everymnote` is used in the `\mnote` group before `\noindent` which immediately precedes marginal note text.

The `\mnotesize` is horizontal size of the marginal notes.

The `\mnoteindent` is horizontal space between body-text and marginal note.

The `\mnoteskip` is a dimen which denotes the vertical shift of marginal note from its normal position. Positive value means shift up, negative down. The `\mnoteskip` register is set to zero after the marginal note is printed. Use it as an exception of marginal note position if the marginal notes overlaps or they are put at bottom of the page.

```
373 \_newtoks\_\everymnote
374 \_newdimen\_\mnotesize \_mnotesize=20mm % the width of the mnote paragraph
375 \_newdimen\_\mnoteindent \_mnoteindent=10pt % distance between mnote and text
376 \_newdimen\_\mnoteskip
377 \_public \_\everymnote \_\mnotesize \_\mnoteindent \_\mnoteskip ;
```

parameters.opm

The `\table` parameters follows. The `\thistable` tokens list register should be used for giving an exception for only one `\table` which follows. It should change locally other parameters of the `\table`. It is reset to empty list after the table is printed.

The `\everytable` tokens list register is applied in every table. There is another difference between these two registers. The `\thistable` is used first, then strut and baselineskip settings are done, then `\everytable` is applied and then the table is printed.

`\tabstrut` configures the height and depth of lines in the table. You can declare `\tabstrut={}`, then normal baselineskip is used in the table. This can be used when you don't use horizontal nor vertical lines in tables.

`\tabiteml` is applied before each item, `\tabitemr` is applied after each item of the table.

`\tablinespace` is additional vertical space between horizontal rules and the lines of the table.

`\hhkern` gives the space between horizontal lines if they are doubled and `\vvkern` gives the space between such vertical lines.

`\tabskipl` is `\tabskip` used before first column, `\tabskipr` is `\tabskip` used after the last column.

`\tsize` is virtual unit of the width of paragraph-like table items when `\table pxto<size>` is used.

```
411 \_newtoks\_\everytable \_newtoks\_\thistable
412 \_newtoks\_\tabiteml \_newtoks\_\tabitemr \_newtoks\_\tabstrut
413 \_newdimen\_\tablinespace \_newdimen\_\vvkern \_newdimen\_\hhkern \_newdimen\_\tsize
414 \_newskip\_\tabskipl \_newskip\_\tabskipr
415 \_everytable={ } % code used after settings in \vbox before table processing
416 \_thistable={ } % code used when \vbox starts, is removed after using it
417 \_tabstrut={\strut}
418 \_tabiteml={\_enspace} % left material in each column
419 \_tabitemr={\_enspace} % right material in each column
420 \_tablinespace=2pt % additional vertical space before/after horizontal rules
421 \_vvkern=1pt % space between double vertical line and used in \frame
422 \_hhkern=1pt % space between double horizontal line and used in \frame
```

parameters.opm

```

423 \_tabskip1=0pt\_relax % \tabskip used before first column
424 \_tabskipr=0pt\_relax % \tabskip used after the last column
425 \_public \everytable \thistable \tabiteml \tabitemr \tabstrut \tablinespace
426 \vvkern \hhkern \tsize \tabskip1r ;

```

The `\eqalign` macro can be configured by `\eqlines` and `\eqstyle` tokens lists. The default values are set in order this macro behaves like in Plain TeX. The `\eqspace` is horizontal space put between equation systems if more columns in `\eqalign` is used.

```

435 \_newtoks \eqlines \eqlines={\_openup\_jot}
436 \_newtoks \eqstyle \eqstyle={\_strut\_displaystyle}
437 \_newdimen \eqspace \eqspace=20pt
438 \_public \eqlines \eqstyle \eqspace ;

```

parameters.opm

The output routine uses token list `\headline` and `\footline` in the same sense as in plain TeX. If they are non-empty then `\hfil` or `\hss` must be here because they are used inside `\hbox to\hsize`.

Assume that page-body text can be typeset in different sizes and different fonts and we don't know in what font context the output routine is invoked. So, it is strongly recommended to declare fixed variants of fonts at the beginning of your document. For example `\fontdef\rmfixed{\rm}`, `\fontdef\itfixed{\it}`. Then use them in headline and footline:

```

\_headline={\itfixed Text of headline, section: \fistmark \hss}
\_footline={\rmfixed \ifodd\pageno \hfill\fi \folio \hfil}

```

```

456 \_newtoks\_headline \_headline={}
457 \_newtoks\_footline \_footline={\_hss\_rmfixed \_folio \_hss}
458 \_public \headline \footline ;

```

parameters.opm

The distance between the `\headline` and the top of the page-text is controlled by the `\headlinedist` register. The distance between bottom of page-text and `\footline` is `\footlinedist`. More precisely: baseline of headline and baseline of first line in page-text have distance `\headlinedist+\topskip`. The baseline of the last line in page-text and the baseline of the footline have distance `\footlinedist`. Default values are inspired from plain TeX.

```

472 \_newdimen \headlinedist \headlinedist=14pt
473 \_newdimen \footlinedist \footlinedist=24pt
474 \_public \headlinedist \footlinedist ;

```

parameters.opm

The `\pgbottomskip` is inserted to the page bottom in the output routine. You can set a less tolerance here than `\raggedbottom` does. By default, no tolerance is given.

```

482 \_newskip \pgbottomskip \pgbottomskip=0pt \_relax
483 \_public \pgbottomskip ;

```

parameters.opm

The `\nextpages` tokens list can include settings which will be used at next pages. It is processed at the end of output routine with `\globaldefs=1` prefix. The `\nextpages` is reset to empty after processing. Example of usage:

```

\_headline={} \nextpages={\_headline={\fixedrm \firstmark \hfil}}

```

This example sets current page with empty headline, but next pages have non-empty headlines.

```

497 \_newtoks \nextpages
498 \_public \nextpages ;

```

parameters.opm

The `\pgbackground` token list can include macros which generate a vertical list. It is used as page background. The top-left corner of such `\vbox` is at the top-left corner of the paper. Example creates the background of all pages yellow:

```

\pgbackground={\Yellow \hrule height 0pt depth\pdfpageheight width\pdfpagewidth}

```

```

510 \_newtoks \pgbackground \pgbackground={} % for page background
511 \_public \pgbackground ;

```

parameters.opm

The parameters used in `\inoval` and `\incircle` macros. The default values (documented in user manual) are set in the macros. The user can re-set these values using tokens `\ovalparams`, `\circleparams`.

```

519 \_newtoks \_ovalparams
520 \_newtoks \_circleparams
521 %\_ovalparams={\_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
522 %          \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt }
523 %\_circleparams={\_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
524 %          \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt}
525
526 \_newdimen \_roundness \_roundness=5mm % used in \clippingoval macro
527
528 \_public \ovalparams \circleparams \roundness ;

```

## 2.8 More OpTeX macros

The second bundle of OpTeX macros is here.

```

3 \_codedecl \eoldef {OpTeX useful macros <2020-05-03>} % preloaded in format

```

We define `\opinput {<file name>}` macro which does `\input {<file name>}` but the catcodes are set to normal catcodes (like OpTeX initializes them) and the catcodes setting are returned back to the current values when the file is read. You can use `\optinput` in any situation inside the document and you will be sure that the file is read correctly with correct catcode settings.

In order to achieve this, we declare `\optexcatcodes` catcode table and `\plaintexcatcodes`. They save the commonly used catcode tables. Note that `\catcodetable` is a part of LuaTeX extension. The catcodetable stack is implemented by OpTeX macros. The `\setctable <catcode table>` pushes current catcode table to the stack and activates catcodes from the `<catcode table>`. The `\restorectable` returns to the saved catcodes from the catcode table stack. So, the `\opinput` macro can be implemented simply:

```

23 \_def\opinput #1{\setctable\optexcatcodes \input {#1}\relax\restorectable}
24
25 \_newcatcodetable \optexcatcodes
26 \_newcatcodetable \plaintexcatcodes
27
28 \_public \optexcatcodes \plaintexcatcodes \opinput ;
29
30 \_savecatcodetable\optexcatcodes
31 {\_catcode\_8 \savecatcodetable\plaintexcatcodes}

```

The implementation of the catcodetable stack follows.

The current catcodes are managed in the `\catcodetable0`. If the `\setctable` is used first (or at the outer level of the stack), then the `\catcodetable0` is pushed to the stack and the current table is re-set to the given `<catcode table>`. The numbers of these tables are stacked to the `\_ctablelist` macro. The `\restorectable` reads the last saved catcode table number from the `\_ctablelist` and uses it.

```

45 \_newcount\_currctable \_currctable=0
46 \_catcodetable0
47
48 \_def\setctable#1{\_edef\_ctablelist{\_the\_currctable}\_ctablelist}%
49 \_catcodetable#1\relax \_currctable=#1\relax
50 }
51 \_def\restorectable{\_ea\_restorectableA\_ctablelist\relax}
52 \_def\_restorectableA#1#2\relax{%
53 \_ifx^#2\_opwarning
54 {You can't use \_noindent\restorectable without previous \_string\setctable}%
55 \_else \_def\_ctablelist{#2}\_catcodetable#1\relax \_currctable=#1\relax \_fi
56 }
57 \_def\_ctablelist{.}
58
59 \_public \setctable \restorectable ;

```

When a special macro is defined with different catcodes then `\normalcatcodes` can be used at the end of such definition. The normal catcodes are restored. The macro reads catcodes from `\optecatodes` table and sets it to the main catcode table 0.

```

69 \_def\_normalcatcodes {\_catcodetable\optexcatcodes \_savecatcodetable0 \_catcodetable0 }
70 \_public \normalcatcodes ;

```

The `\load` [*<filename-list>*] loads files specified in comma separated *<filename-list>*. The first space (after comma) is ignored using the trick `#1#2,:`: first parameter is unseparated. The `\load` macro saves the information about loaded files by setting `\load:<filename>` as a defined macro.

If the `\_afterload` macro is defined then it is run after `\_opinput`. The catcode setting should be here. Note that catcode setting done in the loaded file is forgotten after the `\opinput`.

more-macros.opm

```
84 \_def \_load [#1]{\_loadA #1,,\_end}
85 \_def \_loadA #1#2,{\_ifx,#1 \_ea \_loadE \_else \_loadB{#1#2}\_ea\_loadA\_fi}
86 \_def \_loadB #1{%
87   \_ifcsname \_load:#1\_endcsname \_else
88     \_isfile {#1.opm}\_iftrue \_opinput {#1.opm}\_else \_opinput {#1}\_fi
89     \_sxddef{\_load:#1}{}%
90     \_trycs{\_afterload}{}\_let\_afterload=\_undefined
91   \_fi
92 }
93 \_def \_loadE #1\_end{}
94 \_public \_load ;
```

The declarator `\optdef\macro` [*<opt default>*] *<params>*{*<replacement text>*} defines the `\macro` with the optional parameter followed by normal parameters declared in *<params>*. The optional parameter must be used as the first first parameter in brackets [...]. If it isn't used then *<opt default>* is taken into account. The *<replacement text>* can use `\the\opt` because optional parameter is saved to the `\opt` tokens register. Note the difference from L<sup>A</sup>T<sub>E</sub>X concept where the optional parameter is in #1. OpT<sub>E</sub>X uses #1 as the first normal parameter (if declared).

The `\_nospaceafter` ignores the following optional space at expand processor level using the negative `\romannumeral` trick.

more-macros.opm

```
110 \_def\_optdef#1[#2]{%
111   \_def#1{\_opt={#2}\_isnextchar[{\_cs{oA:\_string#1}}{\_cs{oB:\_string#1}}}%
112   \_sdef{oA:\_string#1}[##1]{\_opt={##1}\_cs{oB:\_string#1\_nospaceafter}}%
113   \_sdef{oB:\_string#1\_nospaceafter}%
114 }
115 \_def\_nospaceafter#1{\_ea#1\_romannumeral-`\.}
116 \_newtoks\_opt
117
118 \_public \_opt \_optdef ;
```

The declarator `\eoldef\macro` #1{*<replacement text>*} defines a `\macro` which scans its parameter to the end of the current line. This is the parameter #1 which can be used in the *<replacement text>*. The catcode of the `\endlinechar` is reset temporarily when the parameter is scanned.

The macro defined by `\eoldef` cannot be used with its parameter inside other macros because the catcode dancing is not possible here. But the `\bracedparam\macro`{*<parameter>*} can be used here. The `\bracedparam` is a prefix which re-sets temporarily the `\macro` to a `\macro` with normal one parameter.

The `\skiptoel` macro reads the text to the end of the current line and ignores it.

more-macros.opm

```
136 \_def\_eoldef #1{\_def #1{\_begingroup \_catcode`^^M=12 \_eoldefA #1}%
137   \_ea\_def\_csname \_csstring #1:M\_endcsname}
138 \_catcode`^^M=12 %
139 \_def\_eoldefA #1#2^^M{\_endgroup\_csname \_csstring #1:M\_endcsname{#2}}%
140 \_normalcatcodes %
141
142 \_eoldef\_skiptoel#1{}
143 \_def\_bracedparam#1{\_ifcsname \_csstring #1:M\_endcsname
144   \_csname \_csstring #1:M\_ea \_endcsname
145   \_else \_csname \_in\_csstring #1:M\_ea \_endcsname \_fi
146 }
147 \_public \_eoldef \skiptoel \bracedparam ;
```

The `\replstring\macro`{*<textA>*}{*<textB>*} replaces all occurrences of *<textA>* by *<textB>* in the `\macro` body. The `\macro` must be defined without parameters. The occurrences of *<textA>* are not replaced if they are “hidden” in braces, for example `...{...<textA>...}....`. The category codes in the *<textA>* must exactly match.

more-macros.opm

```
158 \_catcode`!=3 \_catcode`?=3
159 \_def\_replstring #1#2#3{% \replstring #1{stringA}{stringB}
160   \_long\_def\_replacestringsA##1#2{\_def #1{##1}\_replacestringsB}%
```

```

161 \_long\_def\_replacestringsB##1#2{\_ifx!##1\_relax \_else \_addto #1{#3##1}%
162 \_ea\_replacestringsB\_fi}%
163 \_ea\_replacestringsA #1?#2!#2%
164 \_long\_def\_replacestringsA##1?{\_def #1{##1}}\_ea\_replacestringsA #1}
165 \_normalcatcodes
166
167 \_public \_replstring ;

```

The `\catcode` primitive is redefined here. Why? There is very common cases like `\catcode`<something>` or `\catcode"<number>` but these characters ``` or `"` can be set as active (typically by `\activettchar` macro). Nothing problematic happens if re-defined `\catcode` is used in this case.

If you really need primitive `\catcode` then you can use `\_catcode`.

more-macros.opm

```

179 \_def\_catcode{\_catcode\_string} % more robust in cases \catcode` or \catcode"

```

The `\removespaces` *<text with spaces>*{*<f>*} expands to *<text without spaces>*.

The `\_ea\_ignorept`*<the<dimen>* expands to a decimal number *\the<dimen>* but without pt unit.

The `\ignoreit`*<token>* just ignores the *<token>*.

more-macros.opm

```

190 \_def\_removespaces #1 {\_isempty{#1}\_iffalse #1\_ea\_removespaces\_fi}
191 \_ea\_def \_ea\_ignorept \_ea#\_ea1\_detokenize{pt}{#1}
192 \_def\_ignoreit#1{}
193
194 \_public \removespaces \ignorept \ignoreit ;

```

You can use expandable `\bp{<dimen>}` convertor from T<sub>E</sub>X *<dimen>* (or from an expression accepted by `\dimexpr` primitive) to a decimal value in big points (used as natural unit in the PDF format). So, you can write, for example:

```
\pdfliteral{q \_bp{.3\hsize-2mm} \_bp{2mm} m 0 \_bp{-4mm} 1 S Q}
```

You can use expandable `\expr{<expression>}` for analogical purposes. It expands to the value of the *<expression>* at expand processor level with `\_decdigits` digits after decimal point. The *<expression>* can include `+-*/()` and decimal numbers in common syntax.

The usage of prefixed versions `\_expr` or `\_bp` is more recommended because user can re-define the control sequences `\expr` or `\bp`.

more-macros.opm

```

213 \_def\_decdigits{3} % digits after decimal point in \_bp and \_expr outputs.
214 \_def\_pttopb{%
215 \_directlua{tex.print(string.format('\_pcent.\_decdigits f',
216 token.scan_dimen()/65781.76))}% pt to bp conversion
217 }
218 \def\_bp#1{\_ea\_pttopb\_dimexpr#1\_relax}
219 \def\_expr#1{\_directlua{tex.print(string.format('\_pcent.\_decdigits f',#1))}}
220
221 \_public \expr \bp ;

```

The pair `\_doc ... \_cod` is used for documenting macros and to printing the technical documentation of the OpT<sub>E</sub>X. The syntax is:

```

\_doc <ignored text>
<documentation>
\_cod <ignored text>

```

The *<documentation>* (and *<ignored text>* too) must be *<balanced text>*. It means that you cannot document only the `{` but you must document the `}` too.

more-macros.opm

```

236 \_long\_def\_doc #1\_cod {\_skiptoeol}

```

## 2.9 Plain T<sub>E</sub>X macros

All macros from plain T<sub>E</sub>X are rewritten here. Differences are mentioned in the documentation below.

plain-macros.opm

```

3 \_codedecl \magstep {Macros from plain TeX <2020-02-14>} % preloaded in format

```

The `\dospecials` works like in plain TeX but does nothing with `_`. If you need to do the same with this character, you can re-define:



`\addto \dospecials{\do\_}`

plain-macros.opm

```
13 \def\dospecials {\do\ \do\\\do\{\do\}\do\$\do\&%
14 \do\#\do\^{\do\~{\do\~K\do\~A\do\%\do\~}}
15 \chardef\active = 13
16
17 \public \dospecials \active ;
```

The shortcuts `\chardef\@one` is not defined in OpTeX. Use normal numbers instead of such obscurities. The `\magstep` and `\magstephalf` are defined with `\space`, (no `\relax`), in order to be expandable.

plain-macros.opm

```
27 \def \magstephalf{1095 }
28 \def \magstep#1{\ifcase#1 1000\or 1200\or 1440\or 1728\or 2074\or 2488\fi\space}
29 \public \magstephalf \magstep ;
```

Plain TeX basic macros and control sequences. `\endgraf`, `\endline`. The `^^L` is not defined in OpTeX because it is obsolete.

plain-macros.opm

```
37 \def^^M{\ } % control <return> = control <space>
38 \def^^I{\ } % same for <tab>
39
40 \def\lq{\ } \def\rq{\ }
41 \def\lbrack[\ ] \def\rbrack[\ ] % They are only public versions.
42 % \catcode^^L=\active \outer\def^^L{\par} % ascii form-feed is "\outer\par" % obsolete
43
44 \let\endgraf=\par \let\endline=\_cr
45 \public \endgraf \endline ;
```

Plain TeX classical `\obeylines` and `\obeyspaces`.

plain-macros.opm

```
51 % In \obeylines, we say '\let^^M=\par' instead of '\def^^M{\par}'
52 % since this allows, for example, '\let\par=\cr \obeylines \halign{...}'
53 {\catcode^^M=13 % these lines must end with %
54 \gdef\obeylines{\catcode^^M=13\let^^M=\par}%
55 \global\let^^M=\par} % this is in case ^^M appears in a \write
56 \def\obeyspaces{\catcode\ =13 }
57 {\obeyspaces\global\let \ =\space}
58 \public \obeylines \obeyspaces ;
```

Spaces. `\thinspace`, `\negthinspace`, `\enspace`, `\enskip`, `\quad`, `\qqquad`, `\smallskip`, `\medskip`, `\bigskip`, `\nointerlineskip`, `\offinterlineskip`, `\topglue`, `\vglue`, `\hglue`, `\slash`.

plain-macros.opm

```
68 \protected\def\thinspace {\_kern .16667em }
69 \protected\def\negthinspace {\_kern-.16667em }
70 \protected\def\enspace {\_kern.5em }
71 \protected\def\enskip {\_hskip.5em\relax}
72 \protected\def\quad {\_hskip1em\relax}
73 \protected\def\qqquad {\_hskip2em\relax}
74 \protected\def\smallskip {\_vskip\smallskipamount}
75 \protected\def\medskip {\_vskip\medskipamount}
76 \protected\def\bigskip {\_vskip\bigskipamount}
77 \def\nointerlineskip {\_prevdepth=-1000pt }
78 \def\offinterlineskip {\_baselineskip=-1000pt \_lineskip=0pt \_lineskiplimit=\_maxdimen}
79
80 \public \thinspace \negthinspace \enspace \enskip \quad \qqquad \smallskip
81 \medskip \bigskip \nointerlineskip \offinterlineskip ;
82
83 \def\topglue {\_nointerlineskip\_vglue-\_topskip\_vglue} % for top of page
84 \def\vglue {\_afterassignment\_vgl1a \_skip0=}
85 \def\vgl1a {\_par \_dimen0=\_prevdepth \_hrule height0pt
86 \_nobreak\_vskip\_skip0 \_prevdepth=\_dimen0 }
87 \def\hglue {\_afterassignment\_hgl1a \_skip0=}
88 \def\hgl1a {\_leavevmode \_count255=\_spacefactor \_vrule width0pt
89 \_nobreak\_hskip\_skip0 \_spacefactor=\_count255 }
90 \protected\def~{\_penalty10000 \ } % tie
91 \protected\def\slash {\_penalty\_exhyphenpenalty} % a '/' that acts like a '-'
92
93 \public \topglue \vglue \hglue \slash ;
```



Penalties macros: `\break`, `\nobreak`, `\allowbreak`, `\filbreak`, `\goodbreak`, `\eject`, `\supereject`, `\dosupereject`, `\removelastskip`, `\smallbreak`, `\medbreak`, `\bigbreak`.

plain-macros.opm

```
102 \protected\def \break {\_penalty-10000 }
103 \protected\def \nobreak {\_penalty10000 }
104 \protected\def \allowbreak {\_penalty0 }
105 \protected\def \filbreak {\_par\_vfil\_penalty-200\_vfilneg}
106 \protected\def \goodbreak {\_par\_penalty-500 }
107 \protected\def \eject {\_par\_break}
108 \protected\def \supereject {\_par\_penalty-20000 }
109 \protected\def \dosupereject {\_ifnum \insertpenalties>0 % something is being held over
110 \_line{\_kern-\_topskip \nobreak \vfill \supereject \fi}
111 \def \removelastskip {\_ifdim\_lastskip=0pt \_else \_vskip-\_lastskip \_fi}
112 \def \smallbreak {\_par\_ifdim\_lastskip<\_smallskipamount
113 \_removelastskip \_penalty-50 \_smallskip \_fi}
114 \def \medbreak {\_par\_ifdim\_lastskip<\_medskipamount
115 \_removelastskip \_penalty-100 \_medskip \_fi}
116 \def \bigbreak {\_par\_ifdim\_lastskip<\_bigskipamount
117 \_removelastskip \_penalty-200 \_bigskip \_fi}
118
119 \public \break \nobreak \allowbreak \filbreak \goodbreak \eject \supereject \dosupereject
120 \removelastskip \smallbreak \medbreak \bigbreak ;
```

Boxes. `\line`, `\leftline`, `\rightline`, `\centerline`, `\rlap`, `\llap`, `\underbar`.

plain-macros.opm

```
128 \def \line {\_hbox to\_hsize}
129 \def \leftline #1{\_line{#1\_hss}}
130 \def \rightline #1{\_line{\_hss#1}}
131 \def \centerline #1{\_line{\_hss#1\_hss}}
132 \def \rlap #1{\_hbox to0pt{#1\_hss}}
133 \def \llap #1{\_hbox to0pt{\_hss#1}}
134 \def \underbar #1{\_setbox0=\_hbox{#1}\_dp0=0pt \_math \_underline{\_box0}$}
135
136 \public \line \leftline \rightline \centerline \rlap \llap \underbar ;
```

The `\strutbox` is declared as 10pt size dependent (like in plain T<sub>E</sub>X), but the macro `\setbaselineskip` (from `fonts-opmac.opm`) redefines it.

plain-macros.opm

```
143 \newbox\strutbox
144 \setbox\strutbox=\_hbox{\_vrule height8.5pt depth3.5pt width0pt}
145 \def \strut {\_relax\_ifmmode\_copy\strutbox\_else\_unhcopy\strutbox\_fi}
146
147 \public \strutbox \strut ;
```

Alignment. `\hidewidth` `\ialign` `\multispan`.

plain-macros.opm

```
153 \def \hidewidth {\_hskip\_hideskip} % for alignment entries that can stick out
154 \def \ialign{\_everycr={}\_tabskip=\_zoskip \_halign} % initialized \halign
155 \newcount\_mscount
156 \def \multispan #1{\_omit \_mscount=#1\_relax
157 \_loop \_ifnum\_mscount>1 \_spanA \_repeat}
158 \def \_spanA {\_span\_omit \_advance\_mscount by-1 }
159
160 \public \hidewidth \ialign \multispan ;
```

Tabbing macros are omitted because they are obsolete.

Indentation and others. `\textindent`, `\item`, `\itemitem`, `\narrower`, `\raggedright`, `\ttraggedright`, `\leavevmode`.

plain-macros.opm

```
169 \def \hang {\_hangindent\_parindent}
170 \def \textindent #1{\_indent\_llap{#1\_enspace}\_ignorespaces}
171 \def \item {\_par\_hang\_textindent}
172 \def \itemitem {\_par\_indent \_hangindent2\_parindent \_textindent}
173 \def \narrower {\_advance\_leftskip\_parindent
174 \_advance\_rightskip\_parindent}
175 \def \raggedright {\_rightskip=0pt plus2em
176 \_spaceskip=.3333em \_xspaceskip=.5em\_relax}
177 \def \ttraggedright {\_tt \_rightskip=0pt plus2em\_relax} % for use with \tt only
178 \def \leavevmode {\_unhbox\_voidbox} % begins a paragraph, if necessary
```

```

179
180 \_public \hang \textindent \item \itemitem \narrower \raggedright \ttraggedright \leavevmode ;

```

Few character codes are set for backward compatibility. But old obscurities (from plain TeX) based on `\mathhexbox` are not supported – an error message and recommendation to directly using of the desired character is implemented by the `\_usedirectly` macro). The user can re-define these control sequences of course.

plain-macros.opm

```

191 %\chardef\%=\%
192 \_let\% = \_pcent % more natural, can be used in lua codes.
193 \_chardef\&=\&
194 \_chardef\#=\#
195 \_chardef\$=\$
196 \_chardef\ss="FF
197 \_chardef\ae="E6
198 \_chardef\oe="F7
199 \_chardef\o="F8
200 \_chardef\AE="C6
201 \_chardef\OE="D7
202 \_chardef\O="D8
203 \_chardef\i="11 \chardef\j="12 % dotless letters
204 \_chardef\aa="E5
205 \_chardef\AA="C5
206 \_chardef\S="9F
207 \_def\l{\_errmessage{\_usedirectly l}}
208 \_def\L{\_errmessage{\_usedirectly L}}
209 %\def\_{\_ifmmode \kern.06em \vbox{\hrule width.3em}\else \_fi} % obsolete
210 \_def\_{\_hbox{}}
211 \_def\dag{\_errmessage{\_usedirectly †}}
212 \_def\ddag{\_errmessage{\_usedirectly ‡}}
213 %\def\copyright{\_errmessage{\_usedirectly ©}}
214 \_def\copyright{@} % << example, what to do
215 %\def\Orb{\_mathhexbox20D} % obsolete (part of Copyright)
216 %\def\P{\_mathhexbox27B} % obsolete
217
218 \_def \_usedirectly #1{Load Unicoded font by \string\fontfam\space and use directly #1}
219 \_def \_mathhexbox #1#2#3{\_leavevmode \_hbox{\$ \_math \_mathchar"#1#2#3$}}
220 \_public \mathhexbox ;

```

Accents. The macros `\oalign`, `\d`, `\b`, `\c`, `\dots`, are defined for backward compatibility.

plain-macros.opm

```

228 \_def \_oalign #1{\_leavevmode\_vtop{\_baselineskip=0pt \_lineskip=.25ex
229 \_ialign{##\_crr#1\_crr}}
230 \_def \_oalignA {\_lineskiplimit=0pt \_oalign}
231 \_def \_oalign {\_lineskiplimit=-\_maxdimen \_oalign} % chars over each other
232 \_def \_shiftr #1{\_dimen0=#1\_kern\_ea\_ignorept \_the\_fontdimen1\_font
233 \_dimen0} % kern by #1 times the current slant
234 \_def \_d #1{\_oalignA{\_relax#1\_crr\_hidewidth\_shiftr{-1ex}.\_hidewidth}}
235 \_def \_b #1{\_oalignA{\_relax#1\_crr\_hidewidth\_shiftr{-3ex}%
236 \_vbox to.2ex{\_hbox{\_char\_macron}\_vss}\_hidewidth}}
237 \_def \_c #1{\_setbox0=\_hbox{#1}\_ifdim\_ht0=1ex\_accent\_cedilla #1%
238 \_else\_oalign{\_unhbox0\_crr\_hidewidth\_cedilla\_hidewidth}\_fi}
239 \_def \_dots{\_relax\_ifmmode\_ldots\_else$\_math\_ldots\_thinks$\_fi}
240 \_public \oalign \oalign \d \b \c \dots ;

```

The accents commands like `\v`, `\.`, `\H`, etc. are not defined. Use the accented characters directly – it is best solution. But you can use the macro `\oldaccents` which defines accented macros.

Much more usable is to define these control sequences to other purposes.

plain-macros.opm

```

250 \_def \_oldaccents {%
251 \_def\`##1{\_accent\_tgrave ##1}}%
252 \_def\'##1{\_accent\_tacute ##1}}%
253 \_def\~##1{\_accent\_caron ##1}}%
254 \_def\u##1{\_accent\_tbreve ##1}}%
255 \_def\=##1{\_accent\_macron ##1}}%
256 \_def\^##1{\_accent\_circumflex ##1}}%
257 \_def\.\_##1{\_accent\_dotaccent ##1}}%
258 \_def\H##1{\_accent\_hungarumlaut ##1}}%
259 \_def\~##1{\_accent\_ttilde ##1}}%

```

```

260 \_def\"##1{{\_accent\_dieresis ##1}}%
261 \_def\r##1{{\_accent\_ring ##1}}%
262 }
263 \_public \oldaccents ;
264
265 % ec-lmr encoding (will be changed after \fontfam macro):
266 \_chardef\_tgrave=0
267 \_chardef\_tacute=1
268 \_chardef\_circumflex=2
269 \_chardef\_ttilde=3
270 \_chardef\_dieresis=4
271 \_chardef\_hungarumlaut=5
272 \_chardef\_ring=6
273 \_chardef\_caron=7
274 \_chardef\_tbreve=8
275 \_chardef\_macron=9
276 \_chardef\_dotaccent=10
277 \_chardef\_cedilla=11
278
279 \_def \_uniaccents {% accents with Unicode
280 \_chardef\_tgrave="0060
281 \_chardef\_tacute="00B4
282 \_chardef\_circumflex="005E
283 \_chardef\_ttilde="02DC
284 \_chardef\_dieresis="00A8
285 \_chardef\_hungarumlaut="02DD
286 \_chardef\_ring="02DA
287 \_chardef\_caron="02C7
288 \_chardef\_tbreve="02D8
289 \_chardef\_macron="00AF
290 \_chardef\_dotaccent="02D9
291 \_chardef\_cedilla="00B8
292 \_chardef\_ogonek="02DB
293 \_let \_uniaccents=\_relax
294 }

```

The last part of plain T<sub>E</sub>X macros. `\hrulefill`, `\dotfill`, `\rightarrowfill`, `\leftarrowfill`, `\magnification`, `\bye`. Math macros are defined in the `math-macros.opm` file.

plain-macros.opm

```

303
304 \_def \_hrulefill {\_leaders\_hrule\_hfill}
305 \_def \_dotfill {\_cleaders\_hbox{\_math\_mkern1.5mu\_mkern1.5mu}\_hfill}
306 \_def \_rightarrowfill {\_math\_smash-\_mkern-7mu%
307 \_cleaders\_hbox{\_mkern-2mu\_smash-\_mkern-2mu}\_hfill
308 \_mkern-7mu\_mathord\_rightarrow$}
309 \_def \_leftarrowfill {\_math\_mathord\_leftarrow\_mkern-7mu%
310 \_cleaders\_hbox{\_mkern-2mu\_smash-\_mkern-2mu}\_hfill
311 \_mkern-7mu\_smash-$}
312
313 \_public \hrulefill \dotfill \rightarrowfill \leftarrowfill ;
314
315 % \downbracefill \upbracefill will be re-defined when Unicode-math is used
316 \_mathchardef \_braceld="37A \_mathchardef \_bracerd="37B
317 \_mathchardef \_bracelu="37C \_mathchardef \_braceru="37D
318 \_def \downbracefill {\_math\_setbox0=\_hbox{\_braceld$}%
319 \_braceld\_leaders\_vrule height\_ht0 depth0pt \_hfill \_braceru
320 \_bracelu\_leaders\_vrule height\_ht0 depth0pt \_hfill \_bracerd$}
321 \_def \upbracefill {\_math\_setbox0=\_hbox{\_braceld$}%
322 \_bracelu\_leaders\_vrule height\_ht0 depth0pt \_hfill \_bracerd
323 \_braceld\_leaders\_vrule height\_ht0 depth0pt \_hfill \_braceru$}
324
325 \_def \_magnification {\_afterassignment \_magA \_count255 }
326 \_def \_magA {\_mag=\_count255 \_truedimen\_hsize \_truedimen\_vsize
327 \_dimen\_footins=8truein
328 }
329 % only for backward compatibility, but \margins macro is preferred.
330 \_public \magnification ;
331
332 \_def \_showhyphens #1{\_setbox0=\_vbox{\_parfillskip=0pt \_hsize=\_maxdimen \_tenrm

```

```

333 \_pretolerance=-1 \tolerance=-1 \hbadness=0 \showboxdepth=0 \ #1}}
334
335 \_def \_bye {\_par \_vfill \_supereject \_end}
336 \_public \_bye ;

```

## 2.10 Preloaded fonts for text mode

Format in lua<sub>TEX</sub> can download only non-Unicode fonts. Latin Modern EC is loaded here. These fonts are totally unusable in Lua<sub>TeX</sub> when languages with out of ASCII or ISO-8859-1 alphabets are used (for example Czech). We load only few 8bit fonts here especially for simple testing the format. But, if the user needs to do a more serious work, he/she can use `\fontfam` macro in order to load a selected font family of Unicode fonts.

We have a dilemma: when the Unicode fonts cannot be preloaded in format then basic font set can be loaded by `\everyjob`. But why to load a set of fonts ta the beginning of every job when there is highly likely that the user will load something completely different. Our decision is: there is a basic 8bit font set and user will load the font at beginning of the document.

The fonts selectors `\tenrm`, `\tenbf`, `\tenit`, `\tenbi`, `\tentt` are declared as `\public` here but only for backward compatibility. We don't use them in the Font Selection System. But the protected versions of these control sequences are used in the Font Selection System.

```

3 \_codedecl \tenrm {Latin Modern fonts (EC) preloaded <2020-01-23>} % loaded in format
4
5 % Only few text fonts are preloaded:
6
7 \_font\_tenrm=ec-lmr10 % roman text
8 \_font\_tenbf=ec-lmbx10 % boldface extended
9 \_font\_tenit=ec-lmri10 % text italic
10 \_font\_tenbi=ec-lmbxi10 % bold italic
11 \_font\_tentt=ec-lmtt10 % typewriter
12 \_tenrm
13
14 \_public \tenrm \tenbf \tenit \tenbi \tentt ;

```

fonts-preload.opm

## 2.11 Scaling fonts in text mode (low-level macros)

The `\setfontsize`  $\langle size\ spec \rangle$  saves the information about  $\langle size\ spec \rangle$ . This information is taken into account when a variant selector (for example `\rm`, `\bf`, `\it`, `\bi`) or `\resizethefont` is used. The  $\langle size\ spec \rangle$  can be:

- $\text{at}\langle dimen \rangle$ , for example `\setfontsize{at12pt}`. It gives the desired font size directly.
- $\text{scaled}\langle scale\ factor \rangle$ , for example `\setfontsize{scaled1200}`. The font is scaled in respect to its native size (which is typically 10 pt). It behaves like `\font\... scaled\langle number \rangle`.
- $\text{mag}\langle decimal\ number \rangle$ , for example `\setfontsize{mag1.2}`. The font is scaled in respect to the current size of the fonts given by the previous `\setfontsize` command.

The initialization value in Op<sub>TEX</sub> is given by `\setfontsize{at10pt}`.

The `\resizethefont` resizes the current font to the size given by previous `\setfontsize`. For example

```

Here is 10 pt text,
\setfontsize{at12pt} 10 pt text here unchanged...
\resizethefont      and 12 pt text is here.

```

The `\setfontsize` command acts like *font modifier*. It means that it saves information about fonts but does not change the font actually until variant selector or `\resizethefont` is used.

The following example demonstrates the `mag` format of `\setfontsize` parameter. It is only a curious example probably not used in practical typography.

```

\def\smaller{\setfontsize{mag.9}\resizethefont}
Text \smaller text \smaller text \smaller text.

```

If you load a font directly by `\font` primitive and you want to create a size-dependent selector for such font then you can use `\resizethefont`:

```
\font\tencomfortaa=Comfortaa-Regular-T1 at10pt
\def\comfortaa{\tencomfortaa\resizethefont}
```

```
\comfortaa Here is 10 pt text
\setfontsize{at12pt}
\comfortaa Here is 12 pt text
```

The example above uses the 8bit tfm font. You can use Unicode font too, of course. The `\fontfam` macro initializes the extended `\font` primitive features for LuaTeX. If you didn't use this command, you must to initialize these features by `\initunifonts` command, for example:

```
\initunifonts
\font\tencyklop=[cyklop-regular] at10pt % the font cyklop-regular.otf is loaded
\def\cyklop{\tencyklop\resizethefont}

\cyklop Here is 10 pt text
\setfontsize{at12pt}
\cyklop Here is 12 pt text
```

### 2.11.1 The `\fontdef` declarator

You can declare `\<newfont>` by the `\fontdef` command.

```
\fontdef \<newfont> {\<font modifiers> \<variant-selector>}
example:
\fontdef \bigfont {\setfontsize{at15pt}\bf}
```

This command runs `\<font modifiers> \<variant-selector>` in a group and sets the resulting current font as `\<newfont>`.

The resulting `\<newfont>` declared by `\fontdef` is “fixed font switch” independent of `\setfontsize` and other font modifiers. More exactly, it is fixed font switch when it is used but it can depend on the current font modifiers and font family and given font modifiers when it is declared.

The parameter of the `\fontdef` macro must be exactly finished by the variant selector. More information about font modifiers and variant selectors are in the section 2.12.

### 2.11.2 The `\fontlet` declarator

We have another command for scaling: `\fontlet` which is able to resize arbitrary font given by its font switch. This font switch was declared it by the `\font` primitive or the `\fontdef` macro.

```
\fontlet \<newfont> = \<fontswitch> \<sizespec>
example:
\fontlet \bigfont = \_tenbf at15pt
```

The resulted `\bigfont` is the same as in previous example where `\fontdef` was used. The advantage of `\fontdef` macro will be more clear when you load font families by `\fontfam` and you are using more font modifiers declared in such families.

Summary: you can declare font switches:

- by the `\font` primitive if you know the font file,
- by the `\fontlet` command if you know the font switch and the size, or
- by the `\fontdef` command if you know the variant and modifiers.

### 2.11.3 Optical sizes

There are font families with more font files where almost the same font is implemented in various design sizes: `cmr5`, `cmr6`, `cmr7`, `cmr8`, `cmr9`, `cmr10`, `cmr12`, `cmr17` for example. This feature is called “optical sizes”. OpTeX chooses a font with an optical size closest to desired size specified by the `\setfontsize`, when `at<dimen>` or `mag<coefficient>` is used. When `scaled<scale factor>` is used then optical size is chosen using the value of the `\defaultoptsize` register and such font is scaled by the specified `<scale factor>`. There is `\defaultoptsize=10pt` by default.

Font collections with optical sizes must be registered by the `\_regtfm` for tfm files or `\_regoptsizes` for Unicode fonts. OpTeX registers 8bit Latin Modern fonts in the format (`fonts-resize.opm` file) and OTF Latin Modern fonts in the `f-lmfonts.opm` file.

## 2.11.4 Implementation notes

fonts-resize.opm

```
3 \_codedecl \setfontsize {Font resizing macros <2020-04-17>} % preloaded in format
```

The `\setfontsize`  $\langle sizespec \rangle$  saves the  $\langle sizespec \rangle$  to the `\_sizespec` macro. The `\_optsize` value is calculated from the  $\langle sizespec \rangle$ . If the  $\langle sizespec \rangle$  is in the `mag<number>` format then the contents of the `\_sizespec` macro is re-calculated to the `at<dimen>` format using previous `\_optsize` value.

fonts-resize.opm

```
14 \_newdimen \_optsize \_optsize=10pt
15 \_newdimen \_defaultoptsize \_defaultoptsize=10pt
16 \_newdimen \_lastmagsize
17
18 \_def \_setfontsize #1{%
19   \_edef \_sizespec{#1}%
20   \_ea \_setoptsize \_sizespec \_relax
21   \_reloading
22 }
23 \_def \_setoptsize {\_isnextchar a{\_setoptsizeA}
24   {\_isnextchar m{\_setoptsizeC}{\_setoptsizeB}}}
25 \_def \_setoptsizeA at#1\_relax{\_optsize=#1\_relax \_lastmagsize=\_optsize % at<dimen>
26 \_def \_setoptsizeB scaled#1\_relax{\_optsize=\_defaultoptsize \_relax} % scaled<scalenum>
27 \_def \_setoptsizeC mag#1\_relax{%
28   \_ifdim \_lastmagsize>0pt \_optsize=\_lastmagsize \_else \_optsize=\_pdffontsize \_font \_fi
29   \_optsize=#1\_optsize
30   \_lastmagsize=\_optsize
31   \_edef \_sizespec{at\_the\_optsize}%
32 }
33 \_public \setfontsize \defaultoptsize ;
```

`\_resizefont`  $\langle variant-name \rangle \langle font switch \rangle$ , for example `\resizefont{bf}\_tenbf` resizes the font given by the variant. The variant XX have its font switch `\_tenXX`. The `\_doresizefont\fontswitch` is used. It works in TFM mode (`\_doresizetfmfont`) or OTF mode (`\_doresizeunifont`). In both modes, it does

$$\_font \_tenXX = \langle fontname \rangle \_sizespec$$

The  $\langle fontname \rangle$  is generated by the `\fontname` T<sub>E</sub>X primitive where `\_rfontskipat` removes the `at<dimen>` part of the `\_fontname` output. The  $\langle fontname \rangle$  is generated differently in OTF mode, see `\_doresizeunifont` macro.

The `\_whatresize` is defined as  $\langle variant-name \rangle$ .

fonts-resize.opm

```
52 \_def \_resizefont#1#2{%
53   \_edef \_whatresize{#1}%
54   \_ifx \_fontselector \_undefined \_doresizefont#2%
55   \_else \_ea \_doresizefont \_fontselector \_fi
56   \_lastmagsize=0pt
57   \_slet{\_tryload#1}{\_relax}%
58 }
59 \_def \_doresizetfmfont#1{\_logfont{#1}%
60   \_ea \_font \_ea#1 \_ea \_rfontskipat
61   \_fontname \_cs{\_ten\_whatresize} \_relax \_space \_sizespec \_relax
62 }
63 \_let \_doresizefont=\_doresizetfmfont
64 \_def \_logfont#1{} % default is no logging of used fonts
65
66 \_def \_rfontskipat#1{\_ifx#1\" \_ea \_rfontskipatX \_else \_ea \_rfontskipatN \_ea#1 \_fi}
67 \_def \_rfontskipatX #1" #2\_relax{" \_whichtfm{#1}" }
68 \_def \_rfontskipatN #1 #2\_relax{\_whichtfm{#1}}
```

`\fontdef`  $\langle font switch \rangle \langle modifiers \rangle \langle variant selector \rangle$  opens group, runs  $\langle modifiers \rangle \langle variant selector \rangle$  (i.e. it runs #2 parameter). The font switch #1 saved in the `\_fontselector` macro is re-declared because the variant selector runs the `\_resizefont`. Now, we need to keep the current meaning of the font switch #1 but we must leave the opened group. This is done by the `\_keepmeaning` macro.

`\fontlet`  $\langle font switch A \rangle \langle font switch B \rangle \langle size spec \rangle$  does

$$\_font \langle font switch A \rangle = \langle fontname \rangle \langle sizespec \rangle$$

The  $\langle fontname \rangle$  is extracted using the primitive command `\_fontname`  $\langle font switch B \rangle$ .



```

85 \def \fontdef #1#2{\begingroup
86   \ifx\fontselector\undefined \def\fontselector{#1}\fi
87   \reloading #2%
88   \ea \keepmeaning \fontselector \endgroup
89 }
90 \def\fontlet#1#2{\ifx #2=\ea\fontlet \ea#1\else
91   \ea\font\ea#1\ea\rfontskipat\fontname#2 \relax\space \fi
92 }
93 \def \keepmeaning #1#2{\global\let\keepmeaningdata=#1%
94   #2\let#1=\keepmeaningdata \global\let\keepmeaningdata=\undefined
95 }
96 \public \fontdef \fontlet ;

```

`\newcurrfontsize`  $\langle size spec \rangle$  sets current font size to the  $\langle size spec \rangle$ . It is implemented by `\fontlet`. The font switch of the current font is extracted by `\the\font`. We must re-create the control sequence `\the\font` because its original meaning is set to “inaccessible” by T<sub>E</sub>X when `\font` primitive is started. `\resizethefont` is implemented by `\newcurrfontsize` using data from the `\sizespec` macro.

```

110 \def \newcurrfontsize #1{% \newcurrfontsize{at25pt}
111   \edef\tmp{\ea\csstring \the\font}%
112   \ea \fontlet \csname \tmp\ea\endcsname \the\font \space #1\relax
113   \csname \tmp\endcsname
114 }
115 \protected\def \resizethefont{\newcurrfontsize\sizespec}
116
117 \public \newcurrfontsize \resizethefont ;

```

The variant selector is defined by `\protected\def\XX{\_tryloadXX \_tenXX}`. The `\_tryloadXX` can be in `\_relax` state if no font modifiers were declared. But normally it does `\_resizefont{XX}\tenXX`. This meaning is activated by the `\_reloading` macro.

```

126 \def\_reloading{\_loadf{rm}\_tenrm \loadf{bf}\_tenbf
127   \loadf{it}\_tenit \loadf{bi}\_tenbi
128 }
129 \def\_loadf#1#2{\_sdef\_tryload#1}{\_ifmode \else \_resizefont{#1}#2\fi}}
130 \def\_tryloadtt{\_resizefont{tt}\_tentt}
131
132 \let\_tryloadrm=\_relax
133 \let\_tryloadbf=\_relax
134 \let\_tryloadit=\_relax
135 \let\_tryloadbi=\_relax

```

The font selection system allows to use `\currvar` instead explicitly specified variant selector. The current variant is extracted from `\the\font` output which could be `\_tenXX` control sequence. Then `\currvar` expands to `\_rm` or `\_it` etc.

```

144 \protected \def \currvar{\_cs\_currvar:\ea \csstring \the\font}}
145 \_sdef\_currvar:\tenrm{\_rm}
146 \_sdef\_currvar:\tenbf{\_bf}
147 \_sdef\_currvar:\tenit{\_it}
148 \_sdef\_currvar:\tenbi{\_bi}
149 \_sdef\_currvar:\tentt{\_tt}
150 \public \currvar ;

```

The `\_regtfm`  $\langle font id \rangle$   $\langle optical size data \rangle$  saves the  $\langle optical size data \rangle$  concerned to  $\langle font id \rangle$ . The  $\langle optical size data \rangle$  is in the form as show below in the code where `\_regtfm` is used.

The `\_wichtfm`  $\langle fontname \rangle$  expands to the  $\langle fontname \rangle$  or to the corrected  $\langle fontname \rangle$  read from the  $\langle optical size data \rangle$ . It is used in the `\_rfontskipat` macro and it is used in `\fontlet` macro. It means that each  $\langle fontname \rangle$  generated by the `\fontname` primitive in the `\fontlet` macro is processed by the `\_wichtfm`. The real  $\langle fontname \rangle$  or corrected  $\langle fontname \rangle$  (depending on the optical data does not exist or exist) is the output of the expansion before `\font` primitive takes this output as its parameter.

The implementation detail: The `\_font id:reg` is defined as the  $\langle optical size data \rangle$  and all control sequences `\_fontname:reg` from this data line has the same meaning because of the `\_reversetfm` macro. The `\_wichtfm` expands this data line and apply `\_dowhichtfm`. This macro select the right result from the data line by testing with the current `\_optsize` value.



```

175 \def\regtfm #1 0 #2 *{\ea\def \_csname _#1:reg\_endcsname{#2 16380 \_relax}%
176 \_def\_tmpa{#1}\_reversetfm #2 * %
177 }
178 \def\_reversetfm #1 #2 {% we need this data for \_setmathfamily
179 \ea\_let\_csname _#1:reg\_ea\_endcsname
180 \_csname _\_tmpa:reg\_endcsname
181 \_if*#2\_else \ea\_reversetfm \_fi
182 }
183 \def\_whichtfm #1{%
184 \_ifcsname _#1:reg\_endcsname
185 \ea\_ea\_ea \_dowhichtfm
186 \_csname _#1:reg\_ea\_endcsname
187 \_else
188 #1%
189 \_fi
190 }
191 \def\_dowhichtfm #1 #2 {%
192 \_ifdim\_optsize<#2pt #1\_ea\_ignoretfm\_else \ea\_dowhichtfm
193 \_fi
194 }
195 \def\_ignoretfm #1\_relax{}

```

Optical sizes data for preloaded 8bit Latin Modern fonts:

```

201 \regtfm lmr 0 ec-lmr5 5.5 ec-lmr6 6.5 ec-lmr7 7.5 ec-lmr8 8.5 ec-lmr9 9.5
202 ec-lmr10 11.1 ec-lmr12 15 ec-lmr17 *
203 \regtfm lmbx 0 ec-lmbx5 5.5 ec-lmbx6 6.5 ec-lmbx7 7.5 ec-lmbx8 8.5 ec-lmbx9 9.5
204 ec-lmbx10 11.1 ec-lmbx12 *
205 \regtfm lmri 0 ec-lmri7 7.5 ec-lmri8 8.5 ec-lmri9 9.5 ec-lmri10 11.1 ec-lmri12 *
206 \regtfm lmtt 0 ec-lmtt10 11.1 ec-lmtt12 *
207
208 \setfontsize {at10pt} % default font size

```

## 2.12 The Font Selection System

The basic principles of the Font Selection System used in OpTeX was documented in the section 1.3.1.

### 2.12.1 Terminology

We distinguish between

- *font switchers*, they are declared by the `\font` primitive or by `\fontlet` or `\fontdef` macros,
- *variant selectors*, there are four basic variant selectors `\rm`, `\bf`, `\it`, `\bi`, there is a special selector `\currvar` and more variant selectors can be declared by the `\famvardef` macro.
- *font modifiers* (for example `\cond`, `\caps`, `\setfontsize{<size spec>}`), they are in two types: build in (like `\setfontsize`) or declared modifiers (by the `\moddef` macro).
- *family selectors* (for example `\Termes`, `\LMfonts`), they are declared typically in the *font family files*.

These selectors / switchers sets its values locally. When the TeX group is leaved then selected font and the *font context* are returned back to the values used when the group was opened. They have the following features:

- The *font switchers* select fonts independent on the font context.
- The *variant selectors* select the font depending on the font context and on the specified variant.
- The *font modifiers* create a change in the font context but they don't select the font itself.
- The *family selectors* set a family in the font context and resets all font modifiers. They don't select the font itself.

The variant selectors and declared font modifiers are defined in the family context. They can behave differently in different families.

The fonts registered in OpTeX have their macros in the *font family files*, each family is declared in one font family file with the name `f-famname.opm`. All families are collected in `fams-ini.opm` and user can give more declarations in the file `fams-local.opm`.

### 2.12.2 Font families, selecting fonts

The `\fontfam` [*Font Family*] opens the relevant font family file where the *Font Family* is declared. The family selector is defined here by rules described in the section 2.12.7. Font modifiers and variant selectors may be declared here. Their definitions depends on given family. The family is set as active in the font context and `\rm` variant selector is run.

The available declared font modifiers and declared variant selectors are listed in the log file when font family is load. Or you can print `\fontfam[catalog]` to show available font modifiers and variant selectors.

The font modifiers can be independent, like `\cond` and `\light`. They can be arbitrary combined (in arbitrary order) and if the font family disposes with all such sub-variants then the desired font is selected (after variant selector is used). On the other hand there are font modifiers which negates the previous font modifier, for example `\cond`, `\extend`. You can reset all modifiers to their initial value by the `\resetmod` command.

You can open more font families by more `\fontfam` commands. Then the general method to selecting the individual font is:

*family selector* *font modifiers* *variant selector*

For example:

```
\fontfam [Heros] % Heros family is active here, default \rm variant.
\fontfam [Termes] % Termes family is active here, default \rm variant.
{\Heros \caps \cond \it The caps+condensed italics in Heros family is here.}
The Termes roman is here.
```

There is one special command `\currvar` which acts as variant selector. It keeps the current variant and the font of such variant is reloaded with respect to the current font context by previously given family selector and font modifiers.

You can use the `\setfontsize` {*sizespec*} command in the same sense as other font modifiers. It saves only information about font size to the font context. See section 2.11. Example:

```
\rm default size \setfontsize{at14pt}\rm here is 14pt size \it italic is
in 14pt size too \bf bold too.
```

Much more comfortable way to resize fonts is using OPmac-like command `\typoysize`, `\typoscale`. These commands prepare the right sizes for math fonts too and re-calculates many internal parameters like `\baselineskip`. See section 2.16 for more information.

### 2.12.3 Math Fonts

Most font families are connected with a preferred Unicode-math font. This Unicode-math is activated when the font family is loaded. If you don't prefer this and you are satisfied with 8bit math CM+AMS fonts preloaded in the OpTeX format then you can use command `\noloadmath` before you load a first font family.

If you want to use your specially selected Unicode-math font then use `\loadmath` {*font\_file*} or `\loadmath` {*font\_name*} before first `\fontfam` is used.

### 2.12.4 Declaring font commands

The font switches can be declared by `\font` primitive or by `\fontdef` or `\fontlet` macros. See the sections 2.11.1 and 2.11.2 for more details. The general format for `\fontdef` is

`\fontdef`{*font switch*} {*family selector*} *font modifiers* *variant selector*}

Such font switches should be used in `\output` routine (headers, footers) for example. We need fixed sizes here. But they are less usable in common text. For example the document includes notices in smaller font. When the notice is started then we want to do all variants smaller: `\rm`, `\it`, `\bf`, etc. It means that the smaller font for notices should be initialized by `\setfontsize{at9pt}\rm` for example. If you want a “notices font selector” then you can do `\def\noticefont{\setfontsize{at9pt}\rm}`. This font selector does not change the `\baselineskip`. If you want to do this then put different `\baselineskip` setting to your definition. But you must not forget that the end of group before `\par` is a typical mistake of TeX users: the last paragraph is in smaller font but in normal `\baselineskip`, because `\baselineskip` setting is taken into account when `\par` command is processed.

Somewhat more complicated task is the “title font selector”, because titles are not only bigger but they are typically in bold variant. When the user puts `{\it...}` into the title then he/she expects bold italic here, no normal italic. You can remember the great song by John Lennon “Let It Be” and define:

```
\def\titelfont{\setfontsize{at14pt}\bf \let\it\bi}
...
{\titelfont here we have bold 14pt font and {\it here} was bold 14pt italics}
```

You can declare a new variant selector by the `\famvardef` macro. This macro has similar syntax as `\fontdef`:

```
\famvardef\⟨new variant selector⟩ {\⟨font modifiers⟩ \⟨variant selector⟩}
```

The `\⟨new variant selector⟩` should be used in the same sense as `\rm`, `\bf` etc. It can be used as the final command in the `\fontdef` or `\famvardef` declarators too. When the `\⟨new variant selector⟩` is used in normal text then it does following steps: pushes current font context to a stack, modifies font context by declared `\⟨font modifiers⟩`, runs following `\⟨variant selector⟩`. It selects a font. Then pops the stack. The font context have its original values but new font is selected.

The `\famvardef` creates the `\⟨new variant selector⟩` family dependent. When the selector is used in another family than it is defined then warning is printed on the terminal “`\⟨var selector⟩` is undeclared in current family” and nothing happens. But you can declare the same variant selector by `\famvardef` macro in the context of new family. Then the same command will be do different work depending on the current font family.

Suppose that the selected font family provides the font modifier `\medium` for mediate weight of fonts but supports only basic variant selectors `\rm`, `\bf`, `\it`, and `\bi`. Then you can declare:

```
\famvardef \mr {\medium\rm}
\famvardef \mi {\medium\it}
```

Now, you can use six independent variant selectors `\rm`, `\bf`, `\it`, `\bi`, `\mr` and `\mi` in the selected font family.

A `\⟨family selector⟩` can be written before `\⟨font modifiers⟩` in the `\famvardef` parameter. Then the `\⟨new variant selector⟩` is declared in the current family but it can use fonts from another family represented by the `\⟨family selector⟩`.

When you are mixing fonts from more families then you probably run into problem with incompatible ex-heights. This problem can be solved using `\setfontsize` and `\famvardef` macros:

```
\fontfam[Heros] \fontfam[Termes]

\def\exhcorr{\setfontsize{mag.88}}
\famvardef\rmsans{\Heros\exhcorr\rm}
\famvardef\itsans{\Heros\exhcorr\it}
```

Compare ex-height of Termes `\rmsans` with Heros `\rm` and Termes.

There exists analogical declarator `\moddef` for declaration family dependent font modifiers. It is described in detail the section [2.12.7](#).

## 2.12.5 Modifying font features

Each OTF font provides “font features”. You can list these font features by `otfinfo -f font.otf`. For example LinLibertine fonts provide `frac` font feature. If it is active then fractions like  $1/2$  are printed in a special form.

The font features are part of the font context data. The macro `\setff {\⟨feature⟩}` acts like family independent font modifier and prepares a new `\⟨feature⟩`. You must use a variant selector in order to reinitialize the font with the new font feature. For example `\setff{+frac}\rm` or `\setff{+frac}\currvar`. You can declare a new variant selector too:

```
\fontfam[LinLibertine]
\famvardef \fraclig {\setff{+frac}\currvar}
Compare 1/2 or 1/10 \fraclig to 1/2 or 1/10.
```

If the used font does not supports given font feature then font is reloaded without warning nor error, silently. The font feature is not activated.

The `onum` font feature (old style digits) is connected to `\caps` macro for Caps+SmallCaps variant in OpTeX font family files. So you need not to create a new modifier, just use `{\caps\currvar 012345}`.

### 2.12.6 Special font modifiers

Despite the font modifiers declared in the font family file (and dependent on the font family), we have following font modifiers (independent of font family):

```
\setfontsize{<size spec>} % sets the font size
\setff{<font feature>} % adds the font feature
\setfontcolor{<color>} % sets font color
\setletterspace{<number>} % sets letter spacing
\setwordspace{<scaling>} % modifies word spacing
```

The `\setfontsize` command is described in the section 2.11. The `\setff` command was described in previous subsection.

`\setfontcolor {<color>}` specifies the color and the opacity of the text. The `<color>` parameter should be in hexadecimal format of four bytes `<red><green><blue><opacity>`, for example `FF0080FF` means full red, zero green, half blue and full opacity. You can use names `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`, `white`, `grey`, `lgrey` (without backslash) instead of the hexadecimal specification. The empty parameter `<color>` means default black color.

That colors of fonts are implemented using LuaTeX internal font feature. This is different approach than using colors in section 2.19.

`\setletterspace {<number>}` specifies letter spacing of the font. The `<number>` is decimal number without unit. The unit is supposed as 1/100 of the font size. I.e. 2.5 means 0.25 pt when the font is at 10 pt size. The empty parameter `<number>` means no letter spacing which is default.

`\setwordspace {<scaling>}` scales the default inter word space (defined in the font) and its stretching and shrinking parameters by given `<scaling>` factor. For example `\setwordspace{2.5}` multiplies inter word space by 2.5.

If you need another font transformations, you can use `\setff` with following font features provided by LuaTeX:

```
\setff{embolden=1.5}\rm % font is bolder because outline has nonzero width
\setff{slant=0.2}\rm % font is slanted by a linear transformation
\setff{extend=1.2}\rm % font is extended by a linear transformation.
\setff{colr=yes}\rm % if the font includes colored characters, use colors
```

Use font transformations mentioned above and `\setletterspace`, `\setwordspace` with care. The best setting of these values is default setting in every font, of course. If you really needs to set a different letter spacing then it is strongly recommended to add `\setff{-liga}` in order to disable ligatures. And setting a positive letter spacing probably needs to scale inter word spacing too.

All mentioned font modifiers (with the exception of `\setfontsize`) work only with Unicode fonts loaded by `\fontfam`.

### 2.12.7 How to create the font family file

The font family file declares the font family for selecting fonts from such family at arbitrary size and with various shapes. Unicode fonts (OTF) are preferred. The following example declares the Heros family:

```
f-heros.opm
3 \famdecl [Heros] \Heros {TeX Gyre Heros fonts based on Helvetica}
4   {\caps \cond} {\rm \bf \it \bi} {\FiraMath}
5   {[texgyreheros-regular]}
6   {\_def\_fontnamegen{[texgyreheros\_condV-\_currV]:\_capsV\_fontfeatures}}
7
8 \wlog{\_detokenize{
9 Modifiers:^^J
10 \caps ..... caps & small caps^^J
11 \cond ..... condensed variants^^J
12 }}
13
14 \moddef \resetmod {\_fsetV caps={},cond={} \_fvars regular bold italic bolditalic }
15 \moddef \caps {\_fsetV caps+=smcp;+onum; }
16 \moddef \nocaps {\_fsetV caps={} }
17 \moddef \cond {\_fsetV cond=cn }
```

```

18 \_moddef \nocond {\_fsetV cond={}}
19
20 \_initfontfamily % new font family must be initialized
21
22 \_loadmath {[FiraMath-Regular]}

```

If you want to write such font file, you need to keep following rules.

- Use the `\_famdecl` command first. It has the following syntax:

```

\_famdecl [<Name of family>] \<Familyselector> {\<comments>}
          {\<modifiers>} {\<variant selectors>} {\<comments about math fonts>}
          {\<font-for-testing>}
          {\_def\_fontnamegen{\<font name or font file name generated>}}

```

This writes information about font family at the terminal and prevents loading such file twice. Moreover, it probes existence of `<font-for-testing>` in your system. If it doesn't exist, the file loading is skipped with a warning on the terminal. The `\_ifexistfam` macro returns false in such case. The `\_fontnamegen` macro must be defined in the last parameter of the `\_famdecl`. More about it is documented below.

- You can use `\_wlog{\_detokenize{...}}` to write additional information into log file.
- You can declare optical sizes using `\_regoptsizes` if there are more font files with different optical sizes (like in Latin Modern). See `f-lmfonts.ofm` file for more information about this special feature.
- Declare font modifiers using `\_moddef` if they are present. The `\resetmod` must be declared in each font family.
- Check if all your declared modifiers does not produce any space in horizontal mode. For example check: `X\caps Y`, the letters `XY` must be printed without any space.
- Optionally, declare new variants by the `\famvardef` macro.
- Run `\_initfontfamily` in order to start the family.
- If math font should be loaded, use `\_loadmath{\<math font>}`.

The `\_fontnamegen` macro (declared in the last parameter of the `\_famdecl`) must expand (at expand processor level only) to a file name of loaded font (or to its font name) and to optional font features appended. The Font Selection System uses this macro at primitive level in the following sense:

```
\font \<selector> {\_fontnamegen} \_sizespec
```

Note that the extended `\font` syntax `\font\<selector> {\<font name>:\<font features>} \<size spec.>` or `\font\<selector> {[{\<font file name>}:\<font features>} \<size spec.>` is expected here.

**Example.** Assume an abstract font family with fonts `xx-Regular.otf`, `xx-Bold.otf`, `xx-Italic.otf` and `xx-BoldItalic.otf`. Then you can declare the `\resetmod` (for initializing the family) by:

```
\_moddef\resetmod{\_fvars Regular Bold Italic BoldItalic }
```

and define the `\_fontnamegen` in the last parameter of the `\_famdecl` by:

```

\_famdecl ...
  {\def\_fontnamegen{[xx-\_currV]}}

```

The following auxiliary macros are used here:

- `\_moddef` declares the family dependent modifier. The `\resetmod` saves initial values for the family.
- `\_fvars` saves four names to the memory, they are used by the `\_currV` macro.
- `\_currV` expands to one of the four names dependent on `\rm` or `\bf` or `\it` or `\bi` variant is required.

Assume that the user needs `\it` variant in this family. Then the `\_fontnamegen` macro expands to `[xx-\_currV]` and it expands to `[xx-Italic]`. The Font Selection System uses `\font {[xx-Italic]}`. This command loads the `xx-Italic.otf` font file.

See more advanced examples in `f-<family>.opm` files. The `f-heros.opm` is listed here. When Heros family is selected and `\bf` is asked then

```
\font {[texgyreheros-bold]:+tlig;} at10pt
```

is processed.

You can use any expandable macros or expandable primitives in the `\_fontnamegen` macro. The simple macros in our example with names `\_⟨word⟩V` are preferred. They expand typically to their content. The macro `\_fsetV ⟨word⟩=⟨content⟩` (terminated by a space) is equivalent to `\def\_⟨word⟩V{⟨content⟩}` and you can use it in font modifiers. You can use the `\_fsetV` macro in more general form:

```
\_fsetV ⟨word-a⟩=⟨value-a⟩,⟨word-b⟩=⟨value-b⟩ ...etc. terminated by a space
```

with obvious result `\def\_⟨word-a⟩V {⟨value-a⟩}\def\_⟨word-b⟩V {⟨value-b⟩}` etc.

Example: if both font modifiers `\caps` and `\cond` were applied from the Heros family, then `\def\_capsV{+smcp;+onum}` and `\def\_condV{cn}` were processed by these font modifiers. If user needs the `\bf` variant at 11 pt now then the

```
\font [{texgyreheroscn-bold]:+smcp;+onum;+tlig;} at11pt
```

is processed. We assume that a font file `texgyreheroscn-bold.otf` is present in your TeX system.

Recommendation: the `\_fontfeatures` macro at the end of the `\_fontnamegen` macro in order to the `\setff`, `\setfontcolor`, `\setletterspace` macros can work.

The `\moddef` macro does more things than simple `\_def`:

- The modifier macros are defined as `\_protected`.
- The modifier macros are defined as family-dependent.

The `\famvardef` macro has the same features.

The `\⟨Familyselector⟩` is defined by the `\_famdecl` macro as:

```
\protected\def\⟨Familyselector⟩ {%
  \_def\_currfamily {⟨Familyselector⟩}%
  \_def\_fontnamegen {⟨font name or font file name generated⟩}%
  \resetmod
```

The font context consists from

- Family context, i.e. `\_currfamily` and `\_fontnamegen` values saved by the `\⟨Familyselector⟩`,
- `\_sizespec` value saved by the `\setfontsize` macro,
- whatever what influences the expansion of the `\_fontnamegen` macro, they are typically macros `\_⟨key⟩V` saved by the font modifiers.

The `\_initfontfamily` must be run after modifiers declaration. It sets `\_let\_resetmod=\resetmod` and runs the `\⟨Familyselector⟩`. Finally, it runs `\_rm`, so first font from new family is loaded and it is ready to use it.

**Name conventions.** Create font modifiers, new variants and the `\⟨Familyselector⟩` only as public, i.e. without `_` prefix. We assume that if user re-defines them then he/she needs not them, so we have no problems.

The name of `\⟨Familyselector⟩` should begin with uppercase letter.

If you need to declare your private modifier (because it is used in another modifiers or macros, for example), use the name `\_wordM`. You can be sure that such name does not influence the private name space used by OpTeX.

**Additional notes.** See the font family file `f-libertine-s.opm` which is another example where no font files but font names are used.

See the font family file `f-lmfonts.opm` where you can find the the example of the optical sizes declaration including a documentation about it.

If you need to create font family file with non-Unicode font, you can do it. The `\_fontnamegen` must expand to the name of TFM file in such case. But we don't prefer such font family files, because they are usable only with languages with alphabet subset to ISO-8859-1 (Unicode are equal to letter codes of such alphabets), but middle or east Europe use languages where such condition is not true.



## 2.12.8 How to write the font family file with optical sizes

You can use `\_optname` macro when `\_fontnamegen` is expanded. This macro is fully expandable and its input is  $\langle internal-template \rangle$  and its output is a part of the font file name  $\langle size-dependent-template \rangle$  with respect to given optical size.

You can declare a collection of  $\langle size-dependent-template \rangle$ s for one given  $\langle internal-template \rangle$  by the `\_regoptsizes` macro. The syntax is shown for one real case:

```
\_regoptsizes lmr.r lmroman?-regular
      5 <5.5 6 <6.5 7 <7.5 8 <8.5 9 <9.5 10 <11.1 12 <15 17 <*
```

In general:

```
\_regoptsizes  $\langle internal-template \rangle$   $\langle general-output-template \rangle$   $\langle resizing-data \rangle$ 
```

Suppose our example above. Then `\_optname{lmr.r}` expands to `lmroman?-regular` where the question mark is substituted by a number depending on current `\_optsize`. If the `\_optsize` lies between two boundary values (they are prefixed by `<` character) then the number written between them is used. For example if  $11.1 < \_optsize \leq 15$  then 12 is substituted instead question mark. The  $\langle resizing-data \rangle$  virtually begins with zero `<0`, but it is not explicitly written. The right part of  $\langle resizing-data \rangle$  must be terminated by `<*` which means "less than infinity".

If `\_optname` gets an argument which is not registered  $\langle internal-template \rangle$  then it expands to `\_failedoptname` which typically ends to error message about missing font. You can redefine `\_failedoptname` macro to some existing font if you find it useful.

We are using a special macro `\_LMregfont` in `f-lmfonts.opm`. It sets the file names to lowercase and enables to use a shortcuts instead real  $\langle resizing-data \rangle$ . There are shortcuts `\_regoptFS`, `\_regoptT`, etc. here. The collection of  $\langle internal-templates \rangle$  are declared, each of them covers a collection of real file names.

The `\_optfontalias`  $\{\langle new-template \rangle\} \{\langle internal-template \rangle\}$  declares  $\langle new-template \rangle$  with the same meaning as previously declared  $\langle internal-template \rangle$ .

The `\_optname` macro can be used even if no optical sizes are provided by a font family. Suppose that font file names are much more chaotic (because artists are very creative people), so you need to declare more systematic  $\langle internal-templates \rangle$  and do an alias from each  $\langle internal-template \rangle$  to  $\langle real-font-name \rangle$ . For example, you can do it as follows:

```
\def\fontalias #1 #2 {\_regoptsizes #1 ?#2 {} <*}
%      alias name      real font name
\fontalias crea-a-regular      {Creative Font}
\fontalias crea-a-bold        {Creative FontBold}
\fontalias crea-a-italic       {Creative oblique}
\fontalias crea-a-bolditalic   {Creative Bold plus italic}
\fontalias crea-b-regular      {Creative Regular subfam}
\fontalias crea-b-bold         {Creative subfam bold}
\fontalias crea-b-italic       {Creative-subfam Oblique}
\fontalias crea-b-bolditalic   {Creative Bold subfam Oblique}
```

## 2.12.9 How to register the font family in the Font Selection System

Once you have prepared a font family file with the name `f- $\langle famname \rangle$ .opm` and T<sub>E</sub>X is able to see it in your filesystem then you can type `\fontfam[ $\langle famname \rangle$ ]` and the file is read, so the information about the font family is loaded. The name  $\langle famname \rangle$  must be lowercase and without spaces in the file name `f- $\langle famname \rangle$ .opm`. On the other hand the `\fontfam` command is more tolerant: you can write uppercase letters and spaces here. The spaces are ignored and uppercase letters are converted to lowercase. For example `\fontfam [LM Fonts]` is equivalent to

and both commands load the file `f-lmfonts.opm`.

You can use your font file in sense of previous paragraph without registering it. But problem is that such families are not listed when `\fontfam[?]` is used and it is not included in font catalogue when `\fontfam[catalog]` is printed. The list of families taken in the catalogue and listed on the terminal is declared in two files: `fams-ini.opm` and `fams-local.opm`. The second file is optional. User can create it and write to it the information about user-defined families using the same syntax as in existed file `fams-ini.opm`.



The information from the user's `fams-local.opm` file has precedence. For example `fams-ini.opm` declares aliases Times→Termes etc. If you have the original Times purchased from Adobe then you can register your declaration of Adobe's Times family in `fams-local.opm`. When a user writes `\fontfam[Times]` then the original Times (not Termes) is used.

The `fams-ini.opm` and `fams-local.opm` files use the macros `\_famifo`, `\_famalias` and `\_famtext`. See the example from `fams-ini.tex`:

```
fams-ini.opm
```

```

3 % Version <2020-02-28>. Loaded in format and secondly on demand by \fontfam[catalog]
4
5 \_famtext {Special name for printing a catalogue:}
6
7 \_faminfo [Catalogue] {Catalogue of all registered font families} {fonts-catalog} {}
8 \_famalias [Catalog]
9
10 \_famtext {Computer Modern like family:}
11
12 \_faminfo [Latin Modern] {TeX Gyre fonts based on Coputer Modern} {f-lmfonts}
13   { -, \nbold, \sans, \sans\nbold, \slant, \ttset, \ttset\slant, \ttset\caps, %
14     \ttprop, \ttprop\bolder, \quotset: {\rm\bf\it\bi}
15     \caps: {\rm\it}
16     \ttlight, \ttcond, \dunhill: {\rm\it} \upital: {\rm} }
17 \_famalias [LMfonts] \_famalias [Latin Modern Fonts]
18
19 \_famtext {TeX Gyre fonts based on Adobe 35:}
20
21 \_faminfo [Termes] {TeX Gyre Termes fonts based on Times} {f-termes}
22   { -, \caps: {\rm\bf\it\bi} }
23 \_famalias [Times]
24
25 \_faminfo [Heros] {TeX Gyre Heros fonts based on Helvetica} {f-heros}
26   { -, \caps, \cond, \caps\cond: {\rm\bf\it\bi} }
27 \_famalias [Helvetica]
```

... etc.

The `\_faminfo` command has the syntax:

```
\_faminfo [<Family Name>] {<comments>} {<file-name>}
           { <mod-plus-vars> }
```

The `<mod-plus-vars>` data is used only when printing catalogue. It consists with one or more pairs `<mods>: {<vars>}` `<mods>: {<vars>}` etc. For each pair: each modifiers (separated by comma) are applied to each `<vars>` and prepared sample is printed. The `-` character means no modifiers should be applied.

The `\_famalias` declares an alias to the last declared family.

The `\_famtext` writes a line to the terminal and to the log file when all families are listed.

## 2.12.10 Implementation of the Font Selection System

```
fonts-select.opm
```

```

3 \_codedecl \fontfam {Fonts selection system <2020-03-18>} % preloaded in format
```

The `\_initunifonts` initializes extended `\font` primitive (to be able to load Unicode fonts). Unfortunately, this part of OpTeX depends on L<sup>A</sup>T<sub>E</sub>X lua codes `lATEX lua codes ltluatex.lua` and `luaotfload-main.lua`. And this code need to be declared a control sequence `\e@alloc@attribute@count` by `\countdef` primitive. Moreover, the `\_initunifont` switches with the `\_doresizefont` macro to OTF mode which is represented by the macro `\_doresizeunifont`. This mode includes a fallback to TFM mode if `\_fontnamegen` is not defined. Finally, the `\_initunifnt` sets itself to relax because we need not to do this work twice.

```
fonts-select.opm
```

```

19 \_def\_initunifonts {%
20   \_ea\_newcount \_csname e@alloc@attribute@count\_endcsname
21   \_global \_csname e@alloc@attribute@count\_endcsname=-1
22   \_directlua{%
23     require("ltluatex")
24     require('luaotfload-main') local _void = luaotfload.main ()
25   }%
26   \_gdef\_rfskipatX ##1" ##2\_relax{"##1"}%
27   \_global\_let \_doresizefont=\_doresizeunifont
28   \_gdef\_tryloadtt {\_fontdef\_tentt{\_def\_fontnamegen{lmmono10-regular}}\_rm}}%
```

```

29 \global\let \initunifonts=\relax % we need not to do this work twice
30 \global\let \initunifonts=\relax
31 }
32 \gdef\doresizeunifont #1{\logfont{#1}%
33 \ifx\fontnamegen\undefined \doresizetfmfont#1\else
34 \font#1=\fontnamegen \sizespec \relax \setwsp#1\relax
35 \fi
36 }
37 \public \initunifonts ;

```

The `\famdecl` [*<Family Name>*] *<Famselector>* {*<comment>*} {*<modifiers>*} {*<variants>*} {*<math>*} {*<font for testing>*} {*<def\fontnamegen{<data>}>*} runs `\initunifonts`, then checks if *<Famselector>* is defined. If it is true, then closes the file by `\endinput`. Else it defines *<Famselector>* and saves it to the `\mainfamcommand` macro because the `\initfontfamily` needs it. The `\currfamily` is set to the *<Famselector>* because the following `\moddef` commands need to be in the right font family context. The `\currfamily` is set to the *<Famselector>* by the *<Famselector>* too, because *<Famselector>* must set the right family context. The font family context is given by the current `\currfamily` value and by the actual meaning of the `\fontnamegen` macro.

fonts-select.opm

```

52 \def\famdecl [#1]#2#3#4#5#6#7#8{%
53 \initunifonts \uniaccents
54 \ifx #2\undefined
55 \isfont{#7}\iffalse
56 \opwarning[Family [1] skipped, font "#7" not found]\ea\ea\ea\endinput \else
57 \edef\currfamily {\csstring #2}%
58 \def\mainfamcommand{#2}\def\mathfaminfo{#6}%
59 \protected\edef#2{\def\noexpand\currfamily{\csstring #2}\unexpanded{#8\resetmod}}%
60 \wterm {FONT: [1] -- \string#2 \detokenize{(#3)^J mods:{#4} vars:{#5} math:{#6}}}%
61 \fi
62 \else \ea #2\ea\endinput \fi
63 }
64 \def\initfontfamily{%
65 \mainfamcommand \reloading \rm
66 }

```

`\regoptsizes` *<internal-template>* *<left-output>*?*<right-output>* *<resizing-data>* prepares data for using by the `\optname` *<internal-template>* macro. The data are saved to the `\_oz:`*<internal-template>* macro. When the `\optname` is expanded then the data are scanned by the macro `\optnameA` *<left-output>*?*<right-output>* *<mid-output>* *<size>* in the loop.

`\optfontalias` {*<template A>*} {*<template B>*} is defined as `\let\_oz:<templateA>=\_oz:<templateB>`.

fonts-select.opm

```

79 \def\regoptsizes #1 #2?#3 #4*{\sdef{\_oz:#1}{#2?#3 #4* }}
80 \def\optname #1{\ifcsname \_oz:#1\endcsname
81 \ea\ea\ea \optnameA \csname \_oz:#1\ea\endcsname
82 \else \failedoptname{#1}\fi
83 }
84 \def\failedoptname #1{optname-fails:(#1)}
85 \def\optnameA #1?#2 #3 <#4 {\ifx##4#1#3#2\else
86 \ifdim\optsize<#4pt #1#3#2\optnameC
87 \else \afterfifi \optnameA #1?#2 \fi\fi
88 }
89 \def\optnameC #1* {\fi\fi}
90 \def\afterfifi #1\fi\fi{\fi\fi #1}
91 \def\optfontalias #1#2{\slet{\_oz:#1}{\_oz:#2}}

```

`\fvars` *<rm-template>* *<bf-template>* *<it-template>* *<bi-template>* saves data for usage by the `\currV` macro. If a template is only dot then previous template is used (it can be used if the font family doesn't dispose with all standard variants).

`\currV` expands to a template declared by `\fvars` depending on the *<variant name>*. Usable only of standard four variants. Next variants can be declared by the `\famvardef` macro.

`\fset` *<key>*=*<value>*, ..., *<key>*=*<value>* expands to `\def\_<key>V{\_<value>}` in the loop.

`\onlyif` *<key>*=*<value-a>*, *<value-b>*, ..., *<value-z>*: {*<what>*} runs *<what>* only if the `\_<key>V` is defined as *<value-a>* or *<value-b>* or ... or *<value-z>*.

fonts-select.opm

```

111 \def\fvars #1 #2 #3 #4 {%
112 \sdef{\fvar:rm}{#1}%

```

```

113 \_sdef{fvar:bf}{#2}%
114 \_ifx.#2\_slet{fvar:bf}{fvar:rm}\_fi
115 \_sdef{fvar:it}{#3}%
116 \_ifx.#3\_slet{fvar:it}{fvar:rm}\_fi
117 \_sdef{fvar:bi}{#4}%
118 \_ifx.#4\_slet{fvar:bi}{fvar:it}\_fi
119 }
120 \_def\_currV{\_cs{fvar:\_whatresize}}
121 \_def\_V{ }
122 \_def \_fsetV #1 {\_fsetVa #1,=}
123 \_def \_fsetVa #1=#2,{\_isempty{#1}\_iffalse
124 \_ifx.#1\_else\_sdef{#1V}{#2}\_ea\_ea\_ea\_fsetVa\_fi\_fi
125 }
126 \_def \_onlyif #1=#2:#3{%
127 \_edef\_act{\_noexpand\_isinlist{, #2,}{, \_cs{#1V},}}\_act
128 \_iftrue #3\_fi
129 }

```

The `\moddef`  $\langle modifier \rangle \{ \langle data \rangle \}$  simply speaking does `\def`  $\langle modifier \rangle \{ \langle data \rangle \}$ , but we need to respect the family context. In fact, `\protected\def\_f:  $\langle current family \rangle : \langle modifier \rangle \{ \langle data \rangle \}$`  is performed and the  $\langle modifier \rangle$  is defined as `\famdepend`  $\langle modifier \rangle \{ \_f : \_currfamily : \langle modifier \rangle \}$ . It expands to `\_f : \_currfamily : \langle modifier \rangle` value if it is defined or it prints warning. When the `\_currfamily` value is changed then we can declare the same  $\langle modifier \rangle$  with different meaning.

When user declare a prefixed variant of the  $\langle modifier \rangle$  then unprefixed modifier name is used in internal macros, this is reason why we are using the `\remifirstunderscore\_tmp` (where `\_tmp` expands to  $\langle something \rangle$  or to  $\langle something \rangle$ ). The `\remifirstunderscore` redefines `\_tmp` in the way that it expands only to  $\langle something \rangle$  without the first `_`.

fonts-select.opm

```

149 \_def \_moddef #1#2{\_edef\_tmp{\_csstring#1}\_remifirstunderscore\_tmp
150 \_sdef\_f:\_currfamily:\_tmp}{#2\_reloading}%
151 \_protected \_edef #1{\_noexpand\_famdepend\_noexpand#1\_f:\_noexpand\_currfamily:\_tmp}}%
152 \_ea \_ifx \_csname\_tmp\_endcsname #1\_else
153 \_ea \_public \_csname\_tmp\_endcsname ;\_fi
154 }
155 \_def\_remifirstunderscore#1{\_ea\_remifirstunderscoreA#1\_relax#1}
156 \_def\_remifirstunderscoreA#1#2\_relax#3{\_if \_#1\_def#3{#2}\_fi}
157
158 \_protected \_def\_resetmod {\_cs{f:\_currfamily:resetmod}} % private variant of \resetmod
159 \_def\_currfamily{} % default current family is empty
160
161 \_def\_famdepend#1#2{\_ifcsname#2\_endcsname \_csname#2\_ea\_endcsname \_else
162 \_opwarning{\_string#1 is undeclared in current family "\_currfamily", ignored}\_fi
163 }
164 \_public \_moddef ;

```

The `\famvardef`  $\langle XX \rangle \{ \langle data \rangle \}$  uses analogical trick like `\moddef` with the `\famdepend` macro. The auxiliary `\famvardefA`  $\langle XX \rangle \_ten \langle XX \rangle \_tryload \langle XX \rangle \{ \langle data \rangle \}$  is used. It does:

- `\protected\def`  $\langle XX \rangle \{ \_famdepend \langle XX \rangle \{ \_f : \_currfamily : \langle XX \rangle \} \}$ ,
- `\def`  $\_f : \langle current family \rangle : \langle XX \rangle \{ \_tryload \langle XX \rangle \_ten \langle XX \rangle \}$  keeps family dependent definition,
- `\def`  $\_tryload \langle XX \rangle \{ \_fontdef \_ten \langle XX \rangle \{ \langle data \rangle \} \}$  loads actually the font  $\_ten \langle XX \rangle$ ,
- `\def`  $\_currvar : \_ten \langle XX \rangle \{ \langle XX \rangle \}$  in order to the `\currvar` macro work correctly.

fonts-select.opm

```

181 \_def\_famvardef#1{\_edef\_tmp{\_csstring#1}\_remifirstunderscore\_tmp
182 \_ea\_famvardefA \_ea#1\_csname\_ten\_tmp\_ea\_endcsname
183 \_csname\_tryload:\_tmp\_endcsname
184 }
185 \_def\_famvardefA #1#2#3#4{% #1=\_XX #2=\_tenXX #3=\_tryloadXX #4=data
186 \_isinlist{.\_rm\_bf\_it\_bi\_currvar\_currvar}#1\_iftrue
187 \_opwarning{\_stringfamvardef:
188 You cannot re-declare private standard variant selector \_string#1}%
189 \_else
190 \_protected\_edef #1{\_noexpand\_famdepend\_noexpand#1\_f:\_noexpand\_currfamily:\_tmp}}%
191 \_sdef\_f:\_currfamily:\_tmp}{#3#2}%
192 \_def#3{\_fontdef#2{#4}}%
193 \_ifx#1\_tt \_addto#1{\_fam\_ttfam}\_fi
194 \_sdef\_currvar:\_csstring#2{#1}%

```

```

195 \_fi
196 }
197 \_public \famvardef ;

```

The `\fontfam` [*Font Family*] does:

- Convert its parameter to lower case and without spaces, e.g. *fontfamily*.
- If the file `f-fontfamily.opm` exists read it and finish.
- Try to load user defined `fams-local.opm`.
- If the *fontfamily* is declared in `fams-local.opm` or `fams-ini.opm` read relevant file and finish.
- Print the list of declared families.

The `fams-local.opm` is read by the `\_tryloadfamslocal` macro. It sets itself to `\_relax` because we need not to load this file twice. The `\_listfamnames` macro prints registered font families to the terminal and to the log file.

```

                                                                    fonts-select.opm
215 \_def\_fontfam[#1]{%
216 \_lowercase{\_edef\_famname{\_ea\_removespaces #1 {} }}%
217 \_isfile {f-\_famname.opm}\_iftrue \_opinput {f-\_famname.opm}%
218 \_else
219 \_tryloadfamslocal
220 \_edef\_famfile{\_trycs{\_famf:\_famname}{}}%
221 \_ifx\_famfile\_empty \_listfamnames
222 \_else \_opinput {\_famfile.opm}%
223 \_fi\_fi
224 }
225 \_def\_tryloadfamslocal{%
226 \_isfile {fams-local.opm}\_iftrue
227 \_opinput {fams-local.opm}
228 \_fi
229 \_let \_tryloadfamslocal=\_relax % need not to load fams-local.opm twice
230 }
231 \_def\_listfamnames {%
232 \_wterm{==== List of font families =====}
233 \_begingroup
234 \_let\_famtext=\_wterm
235 \_def\_faminfo [#1]#2#3#4{%
236 \_wterm{ \_space\_noexpand\fontfam [#1] -- #2}%
237 \_let\_famalias=\_famaliasA}%
238 \_opinput {fams-ini.opm}
239 \_isfile {fams-local.opm}\_iftrue \_opinput {fams-local.opm}\_fi
240 \_message{^~J}%
241 \_endgroup
242 }
243 \_def\_famaliasA{\_message{ \_space\_space\_space\_space -- alias:}
244 \_def\_famalias[#1]{\_message{[#1]}}\_famalias
245 }
246 \_public \fontfam ;

```

When the `fams-ini.opm` or `fams-loca.opm` files are read then we need to save only a mapping from family names or alias names to the font family file names. All other information is ignored in this case. But if these files are read by the `\_listfamnames` macro or when printing a catalog then more information is used and printed.

`\_famtext` does nothing or prints the text on the terminal.

`\_faminfo` [*Family Name*] {*comments*} {*file-name*} {*mod-plus-vars*} does

`\_def \_famf:`*familyname* {*file-name*} or prints information on the terminal.

`\_famalias` [*Family Alias*] does `\_def \_famf:`*familyalias* {*file-name*} where *file-name* is stored from the previous `\_faminfo` command. Or prints information on the terminal.

```

                                                                    fonts-select.opm
265 \_def\_famtext #1{}
266 \_def\_faminfo [#1]#2#3#4{%
267 \_lowercase{\_edef\_tmp{\_ea\_removespaces #1 {} }}%
268 \_sdef{\_famf:\_tmp}{#3}%
269 \_def\_famfile{#3}%
270 }
271 \_def\_famalias [#1]{%
272 \_lowercase{\_edef\_famname{\_ea\_removespaces #1 {} }}%

```

```

273 \_sdef{_famf:\_famname\_ea}\_ea{\_famfile}%
274 }
275 \_input fams-ini.opm
276 \_let\_famfile=\_undefined

```

When the `\fontfam[catalog]` is used then the file `fonts-tatalog.opm` is read. The macro `\_faminfo` is redefined here in order to print catalog samples of all declared modifiers/variant pairs. The user can declare different samples and different behavior of the catalog, see the end of catalog listing for more information. The default parameters `\catalogsample`, `\catalogmathsample`, `\catalogonly` and `\catalogexclude` of the catalog are declared here.

fonts-select.opm

```

289 \_newtoks \_catalogsample
290 \_newtoks \_catalogmathsample
291 \_newtoks \_catalogonly
292 \_newtoks \_catalogexclude
293 \_catalogsample={ABCDabcd Qsty fi fl áéíóúüű řžč ĀÉĪŌŪ ŘŽČ 0123456789}
294
295 \_public \_catalogonly \_catalogexclude \_catalogsample \_catalogmathsample ;

```

The font features are managed in the `\_fontfeatures` macro. They have their implicit values saved in the `\_defaultfontfeatures` and the `\setff {<features>}` can add next font features. If there is the same font feature as the newly added one then the old value is removed from the `\_fontfeatures` list.

fonts-select.opm

```

305 \_def \_defaultfontfeatures {+tlig;}
306 \_def \_setff #1{%
307   \_ifx~#1~\_let \_fontfeatures=\_defaultfontfeatures
308   \_else \_edef\_fontfeatures{\_fontfeatures #1;}\_fi
309   \_reloading
310 }
311 \_setff {} % default font features: +tlig;
312 \_def \_removefeature #1{%
313   \_isinlist\_fontfeatures{#1}\_iftrue
314     \_def\_tmp ##1#1##2;##3\_relax{\_def\_fontfeatures{##1##3}}%
315     \_ea \_tmp \_fontfeatures \_relax
316   \_fi
317 }
318 \_public \_setff ;

```

The `\setfontcolor` and `\setletterspace` are macros based on the special font features provided by LuaTeX (and by XeTeX too but it is not our business). The `\setwordspace` recalculates the `\fontdimen2,3,4` of the font using the `\setwsp` macro which is used by the `\doresizeunifont` macro. It activates a dummy font feature `+Ws` too in order the font is reloaded by the `\font` primitive (with independent `\fontdimen` registers).

fonts-select.opm

```

330 \_def\_savedfontcolor{}
331 \_def\_savedletterspace{}
332 \_def\_savedwsp{}
333
334 \_def \_setfontcolor #1{\_removefeature{color=}%
335   \_edef\_tmp{\_calculatefontcolor{#1}}%
336   \_ifx\_tmp\_empty \_else \_edef\_fontfeatures{\_fontfeatures color=\_tmp;}\_fi
337   \_reloading
338 }
339 \_def \_setletterspace #1{\_removefeature{letterspace=}%
340   \_if~#1~\_else \_edef\_fontfeatures{\_fontfeatures letterspace=#1;}\_fi
341   \_reloading
342 }
343 \_def \_setwordspace #1{%
344   \_if~#1~\_def\_setwsp##1{\_removefeature{+Ws}%
345     \_else \_def\_setwsp{\_setwspA{#1}}\_setff{+Ws}\_fi
346     \_reloading
347 }
348 \_def\_setwsp #1{}
349 \_def\_setwspA #1#2{\_fontdimen2#2=#1\_fontdimen2#2%
350   \_fontdimen3#2=#1\_fontdimen3#2\_fontdimen4#2=#1\_fontdimen4#2}
351
352 \_def\_calculatefontcolor#1{\_trycs{fc:#1}{#1}} % you can define more smart macro ...
353 \_sdef{fc:red}{FF0000FF} \_sdef{fc:green}{00FF00FF} \_sdef{fc:blue}{0000FFFF}

```

```

354 \sdef{fc:yellow}{FFFF00FF} \sdef{fc:cyan}{00FFFFFF} \sdef{fc:magenta}{FF00FFFF}
355 \sdef{fc:white}{FFFFFF} \sdef{fc:grey}{00000080} \sdef{fc:lgrey}{00000025}
356 \sdef{fc:black}{} % ... you can declare more colors...
357
358 \_public \setfontcolor \setletterspace \setwordsspace ;

```

## 2.13 Preloaded fonts for math mode

The Computer Modern and AMS fonts are preloaded here in classical math-fam concept, where each math family includes three fonts with max 256 characters (typically 128 characters).

On the other hand, when `\fontfam` macro is used in the document then text font family and appropriate math family is loaded with Unicoed fonts, i.e. Unicoed-math is used. It re-defines all settings given here.

The general rule of usage the math fonts in different sizes in OpTeX says: set three sizes by the macro `\setmathsizes` [*text-size*]/[*script-size*]/[*scriptscript-size*] and then load all math fonts in given sizes by `\normalmath` or `\boldmath` macros. For example

```
\setmathsizes[12/8.4/6]\normalmath ... math typesetting at 12 pt is ready.
```

```

3 \_codedecl \normalmath {Math fonts CM + AMS preloaded <2020-05-06>} % preloaded in format
math-preload.opm

```

We have two math macros `\normalmath` for normal shape of all math symbols and `\boldmath` for bold shape of all math symbols. The second one can be used in bold titles, for example. These macros load all fonts from all given math font families.

```

12 \_def\normalmath{%
13   \_loadmathfamily 0 cmr % CM Roman
14   \_loadmathfamily 1 cmmi % CM Math Italic
15   \_loadmathfamily 2 cmsy % CM Standard symbols
16   \_loadmathfamily 3 cmex % CM extra symbols
17   \_loadmathfamily 4 msam % AMS symbols A
18   \_loadmathfamily 5 msbm % AMS symbols B
19   \_loadmathfamily 6 rsfs % script
20   \_loadmathfamily 7 eufm % fractur
21   \_loadmathfamily 8 bfsans % sans serif bold
22   \_loadmathfamily 9 bisans % sans serif bold slanted (for vectors)
23 % \_setmathfamily 10 \_tentt
24 % \_setmathfamily 11 \_tenit
25   \_setmathdimens
26 }
27 \_def\boldmath{%
28   \_loadmathfamily 0 cmbx % CM Roman Bold Extended
29   \_loadmathfamily 1 cmmib % CM Math Italic Bold
30   \_loadmathfamily 2 cmbxy % CM Standard symbols Bold
31   \_loadmathfamily 3 cmexb % CM extra symbols Bold
32   \_loadmathfamily 4 msam % AMS symbols A (bold not available?)
33   \_loadmathfamily 5 msbm % AMS symbols B (bold not available?)
34   \_loadmathfamily 6 rsfs % script (bold not available?)
35   \_loadmathfamily 7 eufb % fractur bold
36   \_loadmathfamily 8 bbfsans % sans serif extra bold
37   \_loadmathfamily 9 bbisans % sans serif extra bold slanted (for vectors)
38 % \_setmathfamily 10 \_tentt
39 % \_setmathfamily 11 \_tenbi
40   \_setmathdimens
41 }
42 \_count18=9 % families declared by \newfam are 12, 13, ...
43
44 \_def \normalmath {\_normalmath} \_def\boldmath {\_boldmath}
math-preload.opm

```

The classical math family selectors `\mit`, `\cal`, `\bbchar`, `\frak` and `\script` are defined here. The `\rm`, `\bf`, `\it`, `\bi` and `\tt` does two things: they are variant selectors for text fonts and math family selectors for math fonts. The idea was adapted from plain TeX.

These macros are redefined when `unimat-codes.opm` is loaded, see the section 2.15.2.

```

57 \_chardef\bffam = 8
58 \_chardef\bifam = 9
math-preload.opm

```



```

59 %\chardef\ttfam = 10
60 %\chardef\itfam = 11
61
62 \protected\def \rm {\tryloadrm \tenrm \fam0 }
63 \protected\def \bf {\tryloadbf \tenbf \fam\bfam}
64 \protected\def \it {\tryloadit \tenit \fam1 }
65 \protected\def \bi {\tryloadbi \tenbi \fam\bifam}
66 \protected\def \tt {\tryloadtt \tentt}
67
68 \protected\def \mit {\fam1 }
69 \protected\def \cal {\fam2 }
70 \protected\def \bbchar {\fam5 } % double stroked letters
71 \protected\def \frak {\fam7 } % fraktur
72 \protected\def \script {\fam6 } % more extensive script than \cal
73
74 \public \rm \bf \it \bi \tt \mit \cal \bbchar \frak \script ;

```

The optical sizes of Computer Modern fonts, AMS and other fonts are declared here.

math-preload.opm

```

81 %% CM math fonts, optical sizes:
82
83 \regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
84             cmmi10 11.1 cmmi12 *
85 \regtfm cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
86 \regtfm cmtex 0 cstex8 8.5 cstex9 9.5 cstex10 *
87 \regtfm cmsy 0 cmsy5 5.5 cmsy6 6.5 cmsy7 7.5 cmsy8 8.5 cmsy9 9.5 cmsy10 *
88 \regtfm cmb5 0 cmb55 5.5 cmb56 6.5 cmb57 7.5 cmb58 8.5 cmb59 9.5 cmb510 *
89 \regtfm cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
90 \regtfm cmexb 0 cmexb10 *
91
92 \regtfm cmr 0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
93             cmr10 11.1 cmr12 15 cmr17 *
94 \regtfm cmbx 0 cmbx5 5.5 cmbx6 6.5 cmbx7 7.5 cmbx8 8.5 cmbx9 9.5
95             cmbx10 11.1 cmbx12 *
96 \regtfm cmti 0 cmti7 7.5 cmti8 8.5 cmti9 9.5 cmti10 11.1 cmti12 *
97 \regtfm cmtt 0 cmtt10 11.1 cmtt12 *
98
99 %% AMS math fonts, optical sizes:
100
101 \regtfm msam 0 msam5 5.5 msam6 6.5 msam7 7.5 msam8 8.5 msam9 9.5 msam10 *
102 \regtfm msbm 0 msbm5 5.5 msbm6 6.5 msbm7 7.5 msbm8 8.5 msbm9 9.5 msbm10 *
103
104 %% fraktur, rsfs, optical sizes:
105
106 \regtfm eufm 0 eufm5 5.5 eufm6 6.5 eufm7 7.5 eufm8 8.5 eufm9 9.5 eufm10 *
107 \regtfm eufb 0 eufb5 5.5 eufb6 6.5 eufb7 7.5 eufb8 8.5 eufb9 9.5 eufb10 *
108 \regtfm rsfs 0 rsfs5 6 rsfs7 8.5 rsfs10 *
109
110 %% bf and bi sansserif math alternatives:
111
112 \regtfm bfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
113             8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
114 \regtfm bisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
115             8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *
116 \regtfm bbfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
117             8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
118 \regtfm bbisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
119             8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *

```

`\loadmathfamily`  $\langle number \rangle$   $\langle font \rangle$  loads one math family, i. e. the triple of fonts in the text size, script size and script-script size. The  $\langle font \rangle$  is  $\langle font-id \rangle$  used in the `\regtfm` parameter or the real TFM name. The family is saved as `\fam $\langle number \rangle$` .

`\setmathfamily`  $\langle number \rangle$   $\langle font-switch \rangle$  loads one math family like `\loadmathfamily` does it. But the second parameter is a  $\langle font-switch \rangle$  declared previously by the `\font` primitive.

The font family is loaded at `\sizemtext`, `\sizemscript` and `\sizemsscript` sizes. These sizes are set by the `\setmathsizes` [ $\langle text-size \rangle$ / $\langle script-size \rangle$ / $\langle scriptscript-size \rangle$ ] macro. These parameters are given in the `\ptmunit` unit, it is set to `1\ptunit` and it is set to 1 pt by default.



`\_corrmsizes` should be used in the `\normalmath` and `\boldmath` macros if you need a size correction when a selected math family is loaded. It is similar as ex-height correction but for math fonts.

`math-preload.opm`

```

142 \def\_corrmsizes{\_ptmunit=1\_ptunit\_relax} % for corrections of sizes in diferent fomts
143
144 \def\_loadmathfamily #1 #2 {\_chardef\_tmp#1\_corrmsizes
145 \_edef\_optsize\save{\_the\_optsize}%
146 \_optsize=\_sizemtext \_font\_mF=\_whichfm{#2} at\_optsize \_textfont#1=\_mF
147 \_optsize=\_sizemscript \_font\_mF=\_whichfm{#2} at\_optsize \_scriptfont#1=\_mF
148 \_optsize=\_sizemsscript \_font\_mF=\_whichfm{#2} at\_optsize \_scriptscriptfont#1=\_mF
149 \_optsize=\_optsize\save \_relax
150 }
151 \def\_setmathfamily #1 #2{\_let\_mF=#2\_chardef\_tmp#1\_corrmsizes
152 \_edef\_optsize\save{\_the\_optsize}%
153 \_optsize=\_sizemtext \_fontlet#2=#2 at\_optsize \_textfont#1=#2%
154 \_optsize=\_sizemscript \_fontlet#2=#2 at\_optsize \_scriptfont#1=#2%
155 \_optsize=\_sizemsscript \_fontlet#2=#2 at\_optsize \_scriptscriptfont#1=#2%
156 \_optsize=\_optsize\save \_let#2=\_mF
157 }
158 \def\_setmathsizes[#1/#2/#3]{%
159 \_def\_sizemtext{#1\_ptmunit}\_def\_sizemscript{#2\_ptmunit}%
160 \_def\_sizemsscript{#3\_ptmunit}%
161 }
162 \_newdimen\_ptunit \_ptunit=1pt
163 \_newdimen\_ptmunit \_ptmunit=1\_ptunit
164
165 \_public \setmathsizes \_ptunit \_ptmunit ;

```

The `\_setmathdimens` macro is used in `\normalmath` or `\boldmath` macros. It makes math dimensions dependent on the font size (plain T<sub>E</sub>X sets them only for 10pt typesetting). The `\skewchar` of some math families are set here too.

`math-preload.opm`

```

174 \def\_setmathdimens{% PlainTeX sets these dimens for 10pt size only:
175 \_delimitershortfall=0.5\_fontdimen6\_textfont3
176 \_nulldelimiterspace=0.12\_fontdimen6\_textfont3
177 \_scriptspace=0.05\_fontdimen6\_textfont3
178 \_skewchar\_textfont1=127 \_skewchar\_scriptfont1=127
179 \_skewchar\_scriptscriptfont1=127
180 \_skewchar\_textfont2=48 \_skewchar\_scriptfont2=48
181 \_skewchar\_scriptscriptfont2=48
182 \_skewchar\_textfont6=127 \_skewchar\_scriptfont6=127
183 \_skewchar\_scriptscriptfont6=127
184 }

```

Finally, we preload a math fonts colleciton in [10/7/5] sizes when the format is generated. This is done when `\_suppressfontnotfounderror=1` because we need not errors when format is generated. Maybe there are not all fonts in the T<sub>E</sub>X distribution installed.

`math-preload.opm`

```

194 \_suppressfontnotfounderror=1
195 \_setmathsizes[10/7/5]\_normalmath
196 \_suppressfontnotfounderror=0

```

## 2.14 Math macros

`math-macros.opm`

```

3 \_codedecl \sin {Math macros plus mathchardefs <2020-05-11>} % preloaded in format

```

The category code of the character `_` remains as letter (11) and the mathocode of it is "8000. It means that it is active character in math mode. It is defined as subscript prefix.

There is a problem: The `xn` is tokenized as `x`, `_`, `n` and it works without problem. But `\intab` is tokenized as `\inta`, `^`, `b`. The control sequence `\inta` isn't defined. We must write `\intab`.

The Lua code presented here solves this problem. But you cannot set our own control sequence in the form `\⟨word⟩` or `\⟨word⟩one-letter` (where `⟨word⟩` is sequence of letters) because such control sequences are inaccessible: preprocessor rewrites it.

The `\mathsbon` macro activates the rewriting rule `\⟨word⟩nonleter` to `\⟨word⟩nonletter` and `\⟨word⟩letternonletter` to `\⟨word⟩letternonletter` at input processor level. The `\mathsboff`

deactivates it. You can ask by `\_ifmathsb` if this feature is activated or deactivated. By default, is is activated in the `\everyjob`.

math-macros.opm

```

27 \catcode\_ = 8 \let\sb = _
28 \catcode\_ = 13 \let _ = \sb
29 \catcode\_ = 11
30 \_private \sb ;
31
32 \_newif\_ifmathsb \_mathsbfalse
33 \_def \_mathsbon {%
34 \_directlua{
35 callback.register("process_input_buffer",
36 function (str)
37 return string.gsub(str.." ", "(\_nbb[a-zA-Z]+)([a-zA-Z]?[\_a-zA-Z])", "\_pcent 1 \_pcent 2")
38 end) }%
39 \_global\_mathsbtrue
40 }
41 \_def \_mathsboff {%
42 \_directlua{ callback.register("process_input_buffer", nil) }%
43 \_global \_mathsbfalse
44 }
45 \_public \_mathsboff \_mathsbon ;

```

All mathcodes are set to equal values as in plain $\TeX$ . But all encoding-dependend declarations (like these) will be set to different values when Unicode-math font is used.

math-macros.opm

```

53 \_mathcode\^@="2201 % \cdot
54 \_mathcode\^A="3223 % \downarrow
55 \_mathcode\^B="010B % \alpha
56 \_mathcode\^C="010C % \beta
57 \_mathcode\^D="225E % \land
58 \_mathcode\^E="023A % \lnot
59 \_mathcode\^F="3232 % \in
60 \_mathcode\^G="0119 % \pi
61 \_mathcode\^H="0115 % \lambda
62 \_mathcode\^I="010D % \gamma
63 \_mathcode\^J="010E % \delta
64 \_mathcode\^K="3222 % \uparrow
65 \_mathcode\^L="2206 % \pm
66 \_mathcode\^M="2208 % \oplus
67 \_mathcode\^N="0231 % \infty
68 \_mathcode\^O="0140 % \partial
69 \_mathcode\^P="321A % \subset
70 \_mathcode\^Q="321B % \supset
71 \_mathcode\^R="225C % \cap
72 \_mathcode\^S="225B % \cup
73 \_mathcode\^T="0238 % \forall
74 \_mathcode\^U="0239 % \exists
75 \_mathcode\^V="220A % \otimes
76 \_mathcode\^W="3224 % \leftrightarrow
77 \_mathcode\^X="3220 % \leftarrow
78 \_mathcode\^Y="3221 % \rightarrow
79 \_mathcode\^Z="8000 % \ne
80 \_mathcode\^["="2205 % \diamond
81 \_mathcode\^\"="3214 % \le
82 \_mathcode\^]="3215 % \ge
83 \_mathcode\^\"="3211 % \equiv
84 \_mathcode\^\"="225F % \lor
85 \_mathcode\_ = "8000 % \space
86 \_mathcode\_!="5021
87 \_mathcode\_\'="8000 % \prime
88 \_mathcode\_("="4028
89 \_mathcode\_)"="5029
90 \_mathcode\_*="2203 % \ast
91 \_mathcode\_+="202B
92 \_mathcode\_\"="613B
93 \_mathcode\_\"="2200
94 \_mathcode\_\"="013A
95 \_mathcode\_\"="013D

```

```

96 \_mathcode`\:="303A
97 \_mathcode`\;="603B
98 \_mathcode`\<="313C
99 \_mathcode`\=="303D
100 \_mathcode`\>="313E
101 \_mathcode`\?="503F
102 \_mathcode`\[="405B
103 \_mathcode`\\"="026E % \backslash
104 \_mathcode`\]= "505D
105 \_mathcode`\_="8000 % math-active subscript
106 \_mathcode`\{"="4266
107 \_mathcode`\|="026A
108 \_mathcode`\}="5267
109 \_mathcode`\^?="1273 % \smallint
110
111 \_delcode`\(="028300
112 \_delcode`\)="029301
113 \_delcode`\[="05B302
114 \_delcode`\]="05D303
115 \_delcode`\<="26830A
116 \_delcode`\>="26930B
117 \_delcode`\_="02F30E
118 \_delcode`\|="26A30C
119 \_delcode`\\"="26E30F

```

All control sequences declared by `\mathchardef` are supposed (by default) only for public usage. It means that they are declared without `_` prefix. If such sequences are used in internal OpTeX macro then their internal prefixed form is declared using `\_private` macro.

These encoding dependent declarations will be set to different values when Unicode-math font is loaded. The declared sequences for math symbols are not hyperlinked in this documentation.

`math-macros.opm`

```

132 \_mathchardef\alpha="010B
133 \_mathchardef\beta="010C
134 \_mathchardef\gamma="010D
135 \_mathchardef\delta="010E
136 \_mathchardef\epsilon="010F
137 \_mathchardef\zeta="0110
138 \_mathchardef\eta="0111
139 \_mathchardef\theta="0112
140 \_mathchardef\iota="0113
141 \_mathchardef\kappa="0114
142 \_mathchardef\lambda="0115
143 \_mathchardef\mu="0116
144 \_mathchardef\nu="0117
145 \_mathchardef\xi="0118
146 \_mathchardef\pi="0119

```

...etc. (see `math-macros.opm`)

The math functions like `log`, `sin`, `cos` are declared in the same way as in plainTeX, but they are `\protected` in OpTeX.

`math-macros.opm`

```

304 \_protected\_def\log {\_mathop{\_rm log}\_nolimits}
305 \_protected\_def\lg {\_mathop{\_rm lg}\_nolimits}
306 \_protected\_def\ln {\_mathop{\_rm ln}\_nolimits}
307 \_protected\_def\lim {\_mathop{\_rm lim}}
308 \_protected\_def\limsup {\_mathop{\_rm lim\_thinsk sup}}
309 \_protected\_def\liminf {\_mathop{\_rm lim\_thinsk inf}}
310 \_protected\_def\sin {\_mathop{\_rm sin}\_nolimits}
311 \_protected\_def\arcsin {\_mathop{\_rm arcsin}\_nolimits}
312 \_protected\_def\sinh {\_mathop{\_rm sinh}\_nolimits}
313 \_protected\_def\cos {\_mathop{\_rm cos}\_nolimits}
314 \_protected\_def\arccos {\_mathop{\_rm arccos}\_nolimits}
315 \_protected\_def\cosh {\_mathop{\_rm cosh}\_nolimits}
316 \_protected\_def\tan {\_mathop{\_rm tan}\_nolimits}
317 \_protected\_def\arctan {\_mathop{\_rm arctan}\_nolimits}
318 \_protected\_def\tanh {\_mathop{\_rm tanh}\_nolimits}
319 \_protected\_def\cot {\_mathop{\_rm cot}\_nolimits}
320 \_protected\_def\coth {\_mathop{\_rm coth}\_nolimits}

```

```

321 %\protected\def\sec {\mathop{\rm sec}\nolimits} % \sec is section
322 \protected\def\csc {\mathop{\rm csc}\nolimits}
323 \protected\def\max {\mathop{\rm max}\nolimits}
324 \protected\def\min {\mathop{\rm min}\nolimits}
325 \protected\def\sup {\mathop{\rm sup}\nolimits}
326 \protected\def\inf {\mathop{\rm inf}\nolimits}
327 \protected\def\arg {\mathop{\rm arg}\nolimits}
328 \protected\def\ker {\mathop{\rm ker}\nolimits}
329 \protected\def\dim {\mathop{\rm dim}\nolimits}
330 \protected\def\hom {\mathop{\rm hom}\nolimits}
331 \protected\def\det {\mathop{\rm det}\nolimits}
332 \protected\def\exp {\mathop{\rm exp}\nolimits}
333 \protected\def\Pr {\mathop{\rm Pr}\nolimits}
334 \protected\def\gcd {\mathop{\rm gcd}\nolimits}
335 \protected\def\deg {\mathop{\rm deg}\nolimits}

```

These macros are defined similarly as in plain $\TeX$ . Only internal macro names from plain $\TeX$  with @ character are re-written in more readable form.

$\backslash\mathrm{sp}$  is alternative for  $\wedge$ . The  $\backslash\mathrm{sb}$  alternative for  $_$  was defined at the line 27 of the file `math-macros.opm`.

```

345 \let\sp=\public \sp ;
346 % \sb=_ , defined at beginning of this file
347
348 \def\thinskip {\mskip\thinmuskip}
349 \protected\def\relax{\ifmmode \thinskip \else \thinspace \fi}
350 \protected\def\medskip{\mskip\medmuskip} \let\medskip = \>
351 \protected\def\thickskip{\mskip\thickmuskip} \let\thickskip = \>
352 \protected\def\thinneg{\mskip-\thinmuskip} \let\thinneg = \>
353 %\def\discretionary{\thinspace\the\textfont2\char2\{ }\{ }\} % obsolete

```

Active  $\backslash\mathrm{prime}$  character is defined here.

```

359 {\catcode\active \gdef'\{\bgroup\primes}} % primes dance
360 \def\primes{\prime\isnextchar'\{\primesA}%
361 {\isnextchar^{\primesB}\{egroup}}}
362 \def\primesA #1{\primes}
363 \def\primesB #1#2{\egroup}
364 \private \prime ;

```

$\backslash\mathrm{big}$ ,  $\backslash\mathrm{Big}$ ,  $\backslash\mathrm{bigg}$ ,  $\backslash\mathrm{Bigg}$ ,  $\backslash\mathrm{bigl}$ ,  $\backslash\mathrm{bigm}$ ,  $\backslash\mathrm{bigr}$ ,  $\backslash\mathrm{Bigl}$ ,  $\backslash\mathrm{Bigm}$ ,  $\backslash\mathrm{Bigr}$ ,  $\backslash\mathrm{biggl}$ ,  $\backslash\mathrm{biggm}$ ,  $\backslash\mathrm{biggr}$ ,  $\backslash\mathrm{Biggl}$ ,  $\backslash\mathrm{Biggm}$ ,  $\backslash\mathrm{Bigg}$ ,  $\backslash\mathrm{Biggr}$  are based on the  $\backslash\mathrm{scalebig}$  macro because we need the dependency on the various sizes of the fonts.

```

373 %{\catcode\active \gdef\scalebig{\not=} % \scalebig is like \ne in math %obsolete
374
375 \def\scalebig#1#2{{\left#1\ vbox to#2\ fontdimen6\ textfont1}%
376 \ kern-\nulldelimiterspace\ right.}}
377 \protected\def\big#1{\scalebig{#1}{.85}}
378 \protected\def\Big#1{\scalebig{#1}{1.15}}
379 \protected\def\bigg#1{\scalebig{#1}{1.45}}
380 \protected\def\Bigg#1{\scalebig{#1}{1.75}}
381 \public \big \Big \bigg \Bigg ;
382
383 \protected\def\bigl{\mathopen\big}
384 \protected\def\bigm{\mathrel\big}
385 \protected\def\bigr{\mathclose\big}
386 \protected\def\Bigl{\mathopen\Big}
387 \protected\def\Bigm{\mathrel\Big}
388 \protected\def\Bigr{\mathclose\Big}
389 \protected\def\biggl{\mathopen\bigg}
390 \protected\def\biggm{\mathrel\bigg}
391 \protected\def\biggr{\mathclose\bigg}
392 \protected\def\Biggl{\mathopen\Bigg}
393 \protected\def\Biggm{\mathrel\Bigg}
394 \protected\def\Biggr{\mathclose\Bigg}
395 \public \bigl \bigm \bigr \Bigl \Bigm \Bigr \biggl \biggm \biggr \Biggl \Biggm \Biggr ;

```

Math relations defined by the  $\backslash\mathrm{jointrel}$  plain  $\TeX$  macro:

```

401 \protected\def\joinrel{\mathrel{\mkern-2.5mu}} % -3mu in plainTeX
402 \protected\def\relbar{\mathrel{\smash{-}}} % \smash, because - has the same height as +
403 \protected\def\Relbar{\mathrel{=}}
404 \mathchardef\lhook="312C
405 \protected\def\hookrightarrow{\lhook\joinrel\rightarrow}
406 \mathchardef\rhook="312D
407 \protected\def\hookleftarrow{\leftarrow\joinrel\rhook}
408 \protected\def\bowtie{\mathrel{\triangleright\joinrel\mathrel{\triangleleft}}}
409 \protected\def\models{\mathrel{|\joinrel=}}
410 \protected\def\Longrightarrow{\Relbar\joinrel\rightarrow}
411 \protected\def\longrightarrow{\relbar\joinrel\rightarrow}
412 \protected\def\longleftarrow{\leftarrow\joinrel\relbar}
413 \protected\def\Longleftarrow{\Leftarrow\joinrel\Relbar}
414 \protected\def\longmapsto{\mapstochar\longrightarrow}
415 \protected\def\longleftarrow{\leftarrow\joinrel\rightarrow}
416 \protected\def\Longleftarrow{\Leftarrow\joinrel\rightarrow}
417 \protected\def\iff{\thicksk\Longleftarrow\thicksk}
418 \private \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
419 \Relbar \rightarrow \relbar \rightarrow \Leftarrow \mapstochar
420 \longrightarrow \Longleftarrow ;
421 \public \joinrel ;

```

`\ldots`, `\cdots`, `\vdots`, `\ddots` from plain T<sub>E</sub>X

```

427 \mathchardef\ldotp="613A % ldot as a punctuation mark
428 \mathchardef\cdotp="6201 % cdot as a punctuation mark
429 \mathchardef\colon="603A % colon as a punctuation mark
430 \public \ldotp \cdotp \colon ;
431
432 \protected\def\ldots{\mathinner{\ldotp\ldotp\ldotp}}
433 \protected\def\cdots{\mathinner{\cdotp\cdotp\cdotp}}
434 \protected\def\vdots{\vbox{\baselineskip=.4em \lineskiplimit=0pt
435   \kern.6em \hbox{.}\hbox{.}\hbox{.}}}
436 \protected\def\ddots{\mathinner{%
437   \mkern1mu\raise.7em\vbox{\kern.7em\hbox{.}\mkern2mu
438   \raise.4em\hbox{.}\mkern2mu\raise.1em\hbox{.}\mkern1mu}}}
439
440 \public \ldots \cdots \vdots \ddots ;

```

Math accents (encoding dependent declarations).

```

446 \protected\def\acute{\mathaccent"7013 }
447 \protected\def\grave{\mathaccent"7012 }
448 \protected\def\ddot{\mathaccent"707F }
449 \protected\def\tilde{\mathaccent"707E }
450 \protected\def\bar{\mathaccent"7016 }
451 \protected\def\breve{\mathaccent"7015 }
452 \protected\def\check{\mathaccent"7014 }
453 \protected\def\hat{\mathaccent"705E }
454 \protected\def\vec{\mathaccent"017E }
455 \protected\def\dot{\mathaccent"705F }
456 \protected\def\widetilde{\mathaccent"0365 }
457 \protected\def\widehat{\mathaccent"0362 }

```

`\overrightarrow`, `\overleftarrow`, `\overbrace`, `\underbrace`, `\skew` macros.

```

463 \def\math{\mathsurround0pt }
464 \protected\def\overrightarrow #1{\vbox{\math\ialign{##\cr
465   \rightarrowfill\cr\noalign{\kern-.1em \nointerlineskip}
466   $\_hfil\_displaystyle{#1}\_hfil$\_cr}}}}
467 \protected\def\overleftarrow #1{\vbox{\math\ialign{##\cr
468   \leftarrowfill\cr\noalign{\kern-.1em \nointerlineskip}
469   $\_hfil\_displaystyle{#1}\_hfil$\_cr}}}}
470 \protected\def\overbrace #1{\mathop{%
471   \vbox{\math\ialign{##\cr\noalign{\kern.3em}
472   \downbracefill\cr\noalign{\kern.3em \nointerlineskip}
473   $\_hfil\_displaystyle{#1}\_hfil$\_cr}}}\limits}
474 \protected\def\underbrace #1{\mathop{\vtop{\math\ialign{##\cr
475   $\_hfil\_displaystyle{#1}\_hfil$\_cr\noalign{\kern.3em \nointerlineskip}
476   \upbracefill\cr\noalign{\kern.3em}}}\limits}

```

```

477 \_protected\_def\_skew #1#2#3{\_muskip0=#1mu\_divide\_muskip0=by2 \_mkern\_muskip0
478 #2{\_mkern-\_muskip0{#3}\_mkern\_muskip0}\_mkern-\_muskip0}{}}
479
480 \_public \overrightarrow \overleftarrow \overbrace \underbrace \skew ;

```

Macros based on `\delimiter`, `\*witdelims` and `\radical` primitives.

math-macros.opm

```

486 \_protected\_def\lmoustache{\_delimiter"437A340 } % top from (, bottom from )
487 \_protected\_def\rmoustache{\_delimiter"537B341 } % top from ), bottom from (
488 \_protected\_def\lgroup{\_delimiter"462833A } % extensible ( with sharper tips
489 \_protected\_def\rgroup{\_delimiter"562933B } % extensible ) with sharper tips
490 \_protected\_def\arrowvert{\_delimiter"26A33C } % arrow without arrowheads
491 \_protected\_def\Arrowvert{\_delimiter"26B33D } % double arrow without arrowheads
492 \_protected\_def\bracevert{\_delimiter"77C33E } % the vertical bar that extends braces
493 \_protected\_def\Vert{\_delimiter"26B30D } \_let\|= \Vert
494 \_protected\_def\vert{\_delimiter"26A30C }
495 \_protected\_def\uparrow{\_delimiter"3222378 }
496 \_protected\_def\downarrow{\_delimiter"3223379 }
497 \_protected\_def\updownarrow{\_delimiter"326C33F }
498 \_protected\_def\Uparrow{\_delimiter"322A37E }
499 \_protected\_def\Downarrow{\_delimiter"322B37F }
500 \_protected\_def\Updownarrow{\_delimiter"326D377 }
501 \_protected\_def\backslash{\_delimiter"26E30F } % for double coset G\_backslash H
502 \_protected\_def\angle{\_delimiter"526930B }
503 \_protected\_def\langle{\_delimiter"426830A }
504 \_protected\_def\rbrace{\_delimiter"5267309 } \_let\|= \rbrace \_let\_rbrace= \rbrace
505 \_protected\_def\lbrace{\_delimiter"4266308 } \_let\|= \lbrace \_let\_lbrace= \lbrace
506 \_protected\_def\rceil{\_delimiter"5265307 }
507 \_protected\_def\lceil{\_delimiter"4264306 }
508 \_protected\_def\rfloor{\_delimiter"5263305 }
509 \_protected\_def\lfloor{\_delimiter"4262304 }
510
511 \_protected\_def\choose{\_atopwithdelims()}
512 \_protected\_def\brack{\_atopwithdelims[]}
513 \_protected\_def\brace{\_atopwithdelims\_lbrace\_rbrace}
514
515 \_protected\_def\_sqrt{\_radical"270370 } \_public \sqrt ;

```

`\mathpalette`, `\vphantom`, `\hphantom`, `\phantom`, `\mathstrut`, and `\smash` macros from plain TeX.

math-macros.opm

```

522 \_def\_mathpalette#1#2{\_mathchoice{#1\_displaystyle{#2}}%
523 {#1\_textstyle{#2}}{#1\_scriptstyle{#2}}{#1\_scriptscriptstyle{#2}}}%
524 \_newbox\_rootbox
525 \_protected\_def\root#1of{\_setbox\_rootbox
526 \_hbox{\_math\_scriptscriptstyle{#1}}}\_mathpalette\_rootA}
527 \_def\_rootA#1#2{\_setbox0=\_hbox{\_math#1\_sqrt{#2}}}\_dimen0=\_ht0
528 \_advance\_dimen0by-\_dp0
529 \_mkern5mu\_raise.6\_dimen0\_copy\_rootbox \_mkern-10mu\_box0 }
530 \_newifi\_ifvp \_newifi\_ifhp
531 \_protected\_def\_vphantom{\_vptrue\_hpfalse\_phant}
532 \_protected\_def\_hphantom{\_vpfalse\_hptrue\_phant}
533 \_protected\_def\_phantom{\_vptrue\_hptrue\_phant}
534 \_def\_phant{\_ifmode\_def\_next{\_mathpalette\_mathphant}}%
535 \_else\_let\_next=\_makephant\_fi\_next}
536 \_def\_makephant#1{\_setbox0\_hbox{#1}\_finphant}
537 \_def\_mathphant#1#2{\_setbox0=\_hbox{\_math#1{#2}}}\_finphant}
538 \_def\_finphant{\_setbox2=\_null
539 \_ifvp \_ht2=\_ht0 \_dp2=\_dp0 \_fi
540 \_ifhp \_wd2=\_wd0 \_fi \_box2 }
541 \_def\_mathstrut{\_vphantom{}}
542 \_protected\_def\_smash{\_relax % \_relax, in case this comes first in \halign
543 \_ifmode\_def\_next{\_mathpalette\_mathsmash}\_else\_let\_next\_makesmash
544 \_fi\_next}
545 \_def\_makesmash#1{\_setbox0=\_hbox{#1}\_finsmash}
546 \_def\_mathsmash#1#2{\_setbox0=\_hbox{\_math#1{#2}}}\_finsmash}
547 \_def\_finsmash{\_ht0=0pt \_dp0=0pt \_box0 }
548 \_public \mathpalette \vphantom \hphantom \phantom \mathstrut \smash ;

```

`\cong`, `\notin`, `\rightleftharpoons`, `\buildrel`, `\doteq`, `\bmod` and `\pmod` macros from plain TeX.



```

555 \protected\def\cong{\mathrel{\mathpalette\overeq\sim}} % congruence sign
556 \def\overeq#1#2{\_lower.05em\_vbox{\_lineskiplimit\_maxdimen\_lineskip=-.05em
557   \_ialign{\_math#1\_hfil#\_hfil$\_crrc#2\_crrc}\_crrc}}
558 \protected\def\notin{\mathrel{\mathpalette\cancel\_in}}
559 \def\cancel#1#2{\_math\_oalign{\_hfil#1\_mkern1mu/\_hfil$\_crrc#1#2$}}
560 \protected\def\rightharpoonups{\mathrel{\mathpalette\_rlhp{}}}
561 \def\_rlhp#1{\_vcenter{\_math\_hbox{\_oalign{\_raise.2em
562   \_hbox{\_#1\_rightharpoonup}\_crrc
563   \_#1\_leftharpoonupdown}}}}
564 \protected\def\buildrel#1\over#2{\mathrel{\mathop{\_kern0pt #2}\_limits^{#1}}}
565 \protected\def\doteq{\_buildrel\_textstyle.\over=}
566 \public \cong \notin \rightharpoonups \buildrel \doteq ;
567
568 \protected\def\bmod{\_nonscript\_mskip-\_medmuskip\_mkern5mu
569   \_mathbin{\_rm mod}\_penalty900\_mkern5mu\_nonscript\_mskip-\_medmuskip}
570 \protected\def\pmod#1{\_allowbreak\_mkern18mu({\_rm mod}\_thinsk\_thinsk#1)}
571 \public \bmod \pmod ;

```

`\matrix`, `\pmatrix` and `\bordermatrix` behave as in Plain TeX, if it is used in the `\displaystyle` or `\textstyle`. On the other hand, it is printed in smaller size (by appropriate amount) in `\scriptstyle` and `\scriptscriptstyle`. This feature is new in OpTeX.

```

581 \protected\def\matrix#1{\_null\_thinsk
582   \_edef\_stylenum{\_the\_numexpr/2\_relax}%
583   \_vcenter{\_matrixbaselines\_math
584     \_ialign{\_hfil$\_matrixstyle##$\_hfil&\_quad\_hfil$\_matrixstyle##$\_hfil\_crrc
585       \_mathstrut\_crrc\_noalign{\_kern-\_baselineskip}
586       #1\_crrc\_mathstrut\_crrc\_noalign{\_kern-\_baselineskip}}}\_thinsk}
587
588 \def\_matrixbaselines{\_normalbaselines \_def\_matrixstyle{}}%
589 \_ifcase\_stylenum \_or \_or
590   \_baselineskip=.7\_baselineskip
591   \_def\_quad {\_hskip.7em\_relax}%
592   \_let\_matrixstyle=\_scriptstyle
593 \_or
594   \_baselineskip=.5\_baselineskip
595   \_def\_quad {\_hskip.5em\_relax}%
596   \_let\_matrixstyle=\_scriptscriptstyle
597 \_fi
598 }
599 \protected\def\pmatrix#1{\_left(\_matrix{#1}\_right)}
600
601 \public \matrix \pmatrix ;

```

The `\cases` and `\bordermatrix` macros are identical from plain TeX.

```

607 \protected\def\cases#1{\_left{\_thinsk\_vcenter{\_normalbaselines\_math
608   \_ialign{##$\_hfil&\_quad##\_hfil\_crrc#1\_crrc}}\_right.}
609
610 \_newdimen\_ptrenwd
611 \_ptrenwd=8.75pt % width of the big left (
612 \protected\def\bordermatrix#1{\_begingroup \_math
613   \_setbox0=\_vbox{\_def\_cr{\_crrc\_noalign{\_kern.2em\_global\_let\_cr\_endline}}}%
614   \_ialign{##$\_hfil\_kern.2em\_kern\_ptrenwd&\_thinspace\_hfil$##$\_hfil
615     &\_quad\_hfil$##$\_hfil\_crrc
616     \_omit\_strut\_hfil\_crrc\_noalign{\_kern-\_baselineskip}%
617     #1\_crrc\_omit\_strut\_cr}}}%
618   \_setbox2=\_vbox{\_unvcopy0 \_global\_setbox1=\_lastbox}%
619   \_setbox2=\_hbox{\_unhbox1 \_unskip\_global\_setbox1=\_lastbox}%
620   \_setbox2=\_hbox{\_kern\_wd1 \_kern-\_ptrenwd\_left(\_kern-\_wd1
621     \_global\_setbox1=\_vbox{\_box1 \_kern.2em}%
622     \_vcenter{\_kern-\_ht1 \_unvbox0 \_kern-\_baselineskip}\_thinsk\_right)$}%
623   \_null\_thicksk\_vbox{\_kern\_ht1 \_box2}\_endgroup}
624
625 \public \cases \bordermatrix ;

```

The `\eqalign` macro behaves like in Plain TeX by default. It creates the `\vcenter` in the math mode. The contents is two column `\halign` with right aligned left column and left aligned right column. The

table items are in `\displaystyle` and the `\baselineskip` is advanced by `\jot` (3pt in plain  $\TeX$ ). It follows from the default settings of `\eqlines` and `\eqstyle` parameters.

In  $\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$ , this macro is more flexible. You can set the `\baselineskip` value by the `\eqlines` parameter and math style by the `\eqstyle` parameter. For example, you need to put the system of equations as an index of sum operator:

```


$$\sum_{\substack{i \in A \\ j \in B \\ k \in C}} i + j + k$$


```

You can write more equation systems one next second:

```


$$\begin{array}{rcl} x + y + z & = & 1 \\ u + v & = & 20 \\ i & = & j \end{array} \quad \begin{array}{rcl} a + b + c & = & -1 \\ f + g & = & -20 \end{array}$$


```

You can use the third column for centered equations without aligning point (as in `\displaylines`, but packed to the `\vcenter` box). For example:

```


$$\left( \begin{array}{rcl} x + y + z & = & 1 \\ u + v & = & 20 \\ i & = & j \end{array} \right) \begin{array}{rcl} a + b + c & = & -1 \\ f + g & = & -20 \end{array}$$


```

The `\eqspace` is additional space used in the third column to separate equation systems one next second.

```

666 \def\eqalign#1{\null\thinskip\vcenter{\the\eqlines\math
667 \ialign{\&\hfil$\the\eqstyle{##}$&$\the\eqstyle{}}{##}$\hfil
668 &\hskip.5\eqspace\hfil$\the\eqstyle{##}$\hskip.5\eqspace\hfil
669 \crcr#1\crcr}}\thinskip}

```

math-macros.opm

`\openup`, `\displaylines` and `\eqalignno` macros are copied from Plain  $\TeX$  unchanged.

```

676 \def\openup{\afterassignment\openupA\dimen0=}
677 \def\openupA{\advance\lineskip by\dimen0
678 \advance\baselineskip by\dimen0
679 \advance\lineskiplimit by\dimen0 }
680 \newif\ifdtop
681 \def\display{\global\dtoptrue\openup\jot\math
682 \everycr{\noalign{\ifdtop \global\dtopfalse \ifdim\prevdepth>-1000pt
683 \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
684 \else \penalty\interdisplaylinepenalty \fi}}}
685 \def\elign{\tabskip=\zskip\everycr{}} % restore inside \display
686 \def\displaylines#1{\display \tabskip=\zskip
687 \halign{\hbox to\displaywidth{$\elign\hfil\displaystyle##\hfil$}\crcr
688 #1\crcr}}
689 \def\eqalignno#1{\display \tabskip=\centering
690 \halign to\displaywidth{\hfil$\elign\displaystyle{##}$\tabskip=\zskip
691 &$\elign\displaystyle{}}{##}$\hfil\tabskip\centering
692 &\llap{$\elign##$}\tabskip\zskip\crcr
693 #1\crcr}}
694 \def\leqalignno#1{\display \tabskip=\centering
695 \halign to\displaywidth{\hfil$\elign\displaystyle{##}$\tabskip=\zskip
696 &$\elign\displaystyle{}}{##}$\hfil\tabskip=\centering
697 &\kern-\displaywidth\rlap{$\elign##$}\tabskip\displaywidth\crcr
698 #1\crcr}}
699 \public \openup \eqalign \displaylines \eqalignno ;

```

math-macros.opm

These macros are inspired from `ams-math.tex` file.

```

706 \def\amsafam{4} \def\amsbfam{5}
707
708 \mathchardef \boxdot "2\amsafam 00
709 \mathchardef \boxplus "2\amsafam 01
710 \mathchardef \boxtimes "2\amsafam 02

```

math-macros.opm

```

711 \mathchardef \square "0\amsafam 03
712 \mathchardef \blacksquare "0\amsafam 04
713 \mathchardef \centerdot "2\amsafam 05
714 \mathchardef \lozenge "0\amsafam 06
715 \mathchardef \blacklozenge "0\amsafam 07
716 \mathchardef \circlearrowright "3\amsafam 08
717 \mathchardef \circlearrowleft "3\amsafam 09
718 \mathchardef \rightleftharpoons "3\amsafam 0A
719 \mathchardef \leftrightharpoons "3\amsafam 0B
720 \mathchardef \boxminus "2\amsafam 0C

```

...etc. (see `math-macros.opm`)

The `\not` macro is re-defined to be smarter than in plain  $\TeX$ . The macro follows this rule:

```

\not< becomes \_nless
\not> becomes \_ngtr
if \_notXXX is defined, \not\XXX becomes \_notXXX;
if \_nXXX is defined, \not\XXX becomes \_nXXX;
otherwise, \not\XXX is done in the usual way.

```

`math-macros.opm`

```

955 \mathchardef \_notchar "3236
956
957 \_protected\_def \_not#1{%
958   \_ifx #1<\_nless \_else
959   \_ifx #1>\_ngtr \_else
960   \_edef\tmpn{\_csstring#1}%
961   \_ifcsname\_not\tmpn\_endcsname \_csname\_not\tmpn\_endcsname
962   \_else \_ifcsname\_n\tmpn\_endcsname \_csname\_n\tmpn\_endcsname
963   \_else \_mathrel{\\_mathord{\\_notchar}\_mathord{#1}}%
964   \_fi \_fi \_fi \_fi}
965 \_private
966 \nleq \ngeq \nless \ngtr \nprec \nsucc \nleqslant \ngeqslant \npreceq
967 \nsucceq \nleqq \ngeqq \nsim \ncong \nsubseteqq \nupseteqq \nsubseteqq
968 \nupseteq \nparallel \nmid \nshortmid \nshortparallel \nvdash \nVdash
969 \nvDash \nVDash \ntrianglerighteq \ntrianglelefteq \ntriangleleft
970 \ntriangleright \nleftarrow \nrightarrow \nLeftarrow \nRightarrow
971 \nLeftrightarrow \leftrightharpoonup \nexists ;
972 \_public \_not ;

```

`\mathstyles{⟨math list⟩}` behaves like `{⟨math list⟩}`, but you can use following commands in the `⟨math list⟩`:

- `\currstyle` which expands to `\displaystyle`, `\textstyle`, `\scriptstyle` or `\scriptscriptstyle` depending on the current math style when `\mathstyles` was opened.
- `\dobystyle{⟨D⟩}{⟨T⟩}{⟨S⟩}{⟨SS⟩}` is expandable macro. It expands to `⟨D⟩`, `⟨T⟩`, `⟨S⟩` or `⟨SS⟩` depending on the current math style when `\mathstyles` was opened.
- The value of the `\stylenum` is 0, 1, 2 or 3 depending on the current math style when `\mathstyles` was opened.

Example of usage of `\mathstyles`: `\def\mathframe#1{\mathstyles{\frame{$\currstyle#1$}}}`.

`math-macros.opm`

```

992 \_newcount\_stylenum
993 \_def\_mathstyles#1{\_mathchoice{\\_stylenum0 #1}{\_stylenum1 #1}%
994   {\\_stylenum2 #1}{\_stylenum1 #1}}
995 \_def\_dobystyle#1#2#3#4{\_ifcase\_stylenum#1\_or#2\_or#3\_or#4\_fi}
996 \_def\_currstyle{\dobystyle\displaystyle\textstyle\scriptstyle\scriptscriptstyle}
997 \_public \mathstyles \dobystyle \currstyle \stylenum ;

```

The `\mathbox{⟨text⟩}` macro is copied from OPmac trick 078. It behaves like `\hbox{⟨text⟩}` but the `⟨text⟩` is scaled to smaller size if it is used in scriptstyle or scriptscript style.

`math-macros.opm`

```

1005 \_def\_mathbox#1{\_mathstyles{\_hbox{
1006   \_ifnum\_stylenum<2 \_everymath{\_currstyle}%
1007   \_else \_typoscale[\_dobystyle{}{}{700}{500}]/\_fi #1}}}%
1008 }
1009 \_public \mathbox ;

```

## 2.15 Unicode-math fonts

The `\loadmath`  $\langle Unicode-math font \rangle$  macro loads math fonts and redefines all default math-codes using `\input unimath-codes.opm`. If Unicode-math font is loaded then `\_mathloadingfalse` is set, so new UnicodeMath font isn't loaded until `\doloadmath` is used.

`\loadboldmath`  $\langle bold-font \rangle$  `\to`  $\langle normal-font \rangle$  loads bold variant only if  $\langle normal-font \rangle$  was successfully loaded by the `\loadmath`. For example:

```
\loadmath      {[xitsmath-regular]}
\loadboldmath {[xitsmath-bold]} \to {[xitsmath-regular]}
```

You can combine more fonts, if you register them to another math families (5, 6, 7, etc.) in the `\normalmath` macro.

The default value of `\normalmath` shows a combination of base Unicode Math font with 8bit Math font at family 4. See definition of `\script` macro where `\fam4` is used. Of course, we need to set `\rmvariables` too, because 8bit font accepts only codes less than 255.

See <http://tex.stackexchange.com/questions/308749/> for more technical details.

The `\loadmath` macro was successfully tested on:

```
\loadmath{[XITSMath-Regular]}      ... XITS MATH
\loadmath{[latinmodern-math]}      ... Latin Modern Math
\loadmath{[texgyretermes-math]}     ... TeXGyre Termes Math
\loadmath{[texgyrebounum-math]}     ... TeXGyre Bonum Math
\loadmath{[texgyrepagella-math]}    ... TeXGyre Pagella Math
\loadmath{[texgyreschola-math]}     ... TeXGyre Schola Math
\loadmath{[texgyredejavu-math]}     ... TeXGyre DeJaVu Math
\loadmath{[LibertinusMath-Regular]} ... Libertinus Math
\loadmath{[FiraMath-Regular]}       ... Fira Math
\loadmath{[Asana-Math]}             ... Asana Math
```

### 2.15.1 Unicode-math macros preloaded in the format

math-unicode.opm

```
3 \_codedecl \loadmath {Unicode Math fonts <2020-02-25>} % preloaded in format
```

`\loadmath`  $\langle Unicode-math font \rangle$  loads given font. It does:

- define `\_unimathfont` as  $\langle Unicode-math font \rangle$ ,
- redefine `\normalmath` and `\boldmath` macros to their Unicode counterparts,
- load the `\_unimathfont` by `\normalmath`,
- print information about loaded font on the terminal,
- redefine all encoding dependent setting by `\input unimath-codes.opm`,
- protect new loading by setting `\_ifmathloading` to false.

`\noloadmath` disallows Unicode-math loading by `\_mathloadingfalse`.

`\doloadmath` allows Unicode-math loading by `\_mathloadingtrue`.

math-unicode.opm

```
19 \_newifi \_ifmathloading \_mathloadingtrue
20
21 \_def\noloadmath{\_mathloadingfalse}
22 \_def\doloadmath{\_mathloadingtrue}
23
24 \_def\loadmath#1{%
25   \_ifmathloading
26   \_initunifonts
27   \_isfont{#1}\_iffalse
28   \_opwarning{Math font "#1" not found, skipped...}%
29   \_else
30     \_def\_unimathfont{#1}%
31     \_let\_normalmath = \_normalunimath \_let\_boldmath = \_boldunimath
32     \_normalmath
33     \_wterm {MATH-FONT: "#1" -- unicode math prepared.}%
34     \_opinput {unimath-codes.opm}%
35     \_mathloadingfalse
36   \_fi\_fi}
37
```

```
38 \public \loadmath \noloadmath \doloadmath ;
```

`\loadboldmath`  $\langle\textit{bold-font}\rangle$   $\rightarrow$   $\langle\textit{normal-font}\rangle$  defines `\unimathboldfont` as  $\langle\textit{bold-font}\rangle$  only if `\unimathfont` is defined as  $\langle\textit{normal-font}\rangle$ . It is used when `\boldmath` macro is run. When no `\unimathboldfont` is defined then the `\boldmath` macro use “fake bold” generated by embolden LuaTeX font feature.

math-unicode.opm

```
48 \def\loadboldmath#1#2\to #3{%
49   \def\tmp{#3}\ifx\unimathfont\tmp % do work only if #3 is loaded as normal Math
50   \isfont"#1"\iffalse
51   \opwarning{Bold-Math font "#1" not found, skipped...}
52   \else
53   \def\unimathboldfont{#1}%
54   \wterm {MATH-FONT: "#1" -- unicode math bold prepared.}%
55   \fi\fi}
56
57 \public \loadboldmath ;
```

The Unicode version of the `\normalmath` and `\boldmath` macros are defined here as `\normalunimath` and `\boldunimath` macros. They are using `\setunimathdimens` in similar sense as `\setmathdimens`.

math-unicode.opm

```
66 \def\normalunimath{%
67   \loadumathfamily 1 {\unimathfont}{} % Base font
68   \loadmathfamily 4 rsfs % script
69   \setunimathdimens
70 }%
71 \def\boldunimath{%
72   \ifx\unimathboldfont \undefined
73   \loadumathfamily 1 {\unimathfont}{embolden=1.7;} % Base faked bold
74   \else
75   \loadumathfamily 1 {\unimathboldfont}{} % Base real bold font
76   \fi
77   \loadmathfamily 4 rsfs % script
78   \setunimathdimens
79 }%
80 \def\setunimathdimens{% PlainTeX sets these dimens for 10pt size only:
81   \delimitershortfall=0.5\fontdimen6\textfont3
82   \nulldelimiterspace=0.12\fontdimen6\textfont3
83   \scriptspace=0.05\fontdimen6\textfont3
84 }
```

`\loadumathfamily`  $\langle\textit{number}\rangle$   $\langle\textit{font}\rangle$   $\langle\textit{font features}\rangle$  loads the given Unicode-math fonts in three sizes given by the `\setmathsizes` macro and sets it as the math family  $\langle\textit{number}\rangle$ . The  $\langle\textit{font features}\rangle$  are added to the default `\mfontfeatures` and to the size dependent features `+ssty=0` if script size is asked or `+ssty=1` if scriptscriptsize is asked. If the fath family 1 is loaded then the family 2 and 3 is set by the same font because TeX needs to read dimension information about generating math formulae from these three math families. All information needed by TeX is collected in single Unicode-math font.

math-unicode.opm

```
99 \def\umathname#1#2{"#1:\mfontfeatures#2"}
100 \def\mfontfeatures{mode=base;script=math;}
101
102 \def\loadumathfamily #1 #2#3 {%
103   \edef\optsizesave{\the\optsize}%
104   \optsize=\sizemtext \font\mF=\umathname{#2}{#3} at\optsize \textfont#1=\mF
105   \ifnum#1=1 \textfont2=\mF \textfont3=\mF \fi
106   \optsize=\sizemscript \font\mF=\umathname{#2}{+ssty=0;#3} at\optsize \scriptfont#1=\mF
107   \ifnum#1=1 \scriptfont2=\mF \scriptfont3=\mF \fi
108   \optsize=\sizemsscript \font\mF=\umathname{#2}{+ssty=1;#3} at\optsize \scriptscriptfont#1=\mF
109   \ifnum#1=1 \scriptscriptfont2=\mF \scriptscriptfont3=\mF \fi
110   \optsize=\optsizesave \relax
111 }
```

Unicode math font includes all typical math alphabets together, user needs not to load more TeX math families. These math alphabets are encoded by different parts of Unicode table. We need auxiliary macros for setting mathcodes by selected math alphabet.

`\umathrange`  $\langle\textit{from-}\rangle\textit{-}\langle\textit{to}\rangle$   $\langle\textit{first}\rangle$  sets `\Umathcodes` of the characters in the interval  $\langle\textit{from}\rangle\textit{-}\langle\textit{to}\rangle$  to  $\langle\textit{first}\rangle$ ,  $\langle\textit{first}\rangle+1$ ,  $\langle\textit{first}\rangle+2$  etc., but `\umathcharholes` are skipped (`\umathcharholes` are parts of

the Unicode table not designed for math alphabets but they causes that the math alphabets are not continuously spread out in the table; I mean that the designers were under the influence of drugs when they created this part of the Unicode table). The  $\langle from \rangle - \langle to \rangle$  clause includes normal letters like A-Z.

$\backslash\_umathrange\greek \langle first \rangle$  is the same as  $\backslash\_umathrange \{ \langle alpha \rangle - \langle omega \rangle \} \langle first \rangle$ .

$\backslash\_umathrange\GREEK \langle first \rangle$  is the same as  $\backslash\_umathrange \{ \langle Alpha \rangle - \langle Omega \rangle \} \langle first \rangle$ .

$\backslash\_greekdef \langle control sequences \rangle \backslash\_relax$  defines each control sequence as a normal character with codes  $\backslash\_umathnumB$ ,  $\backslash\_umathnumB+1$ ,  $\backslash\_umathnumB+2$  etc. It is used for redefining the control sequences for math Greek  $\backslash\alpha$ ,  $\backslash\beta$ ,  $\backslash\gamma$  etc.

math-unicode.opm

```

142 \_newcount\_umathnumA \_newcount\_umathnumB
143
144 \_def\_umathcorr#1#2{\_ea#1\_ea{\_the#2}}
145 \_def\_umathprepare#1{\_def\_umathscanholes##1[#1]##2##3\_relax{##2}}
146 \_def\_umathvalue#1{\_ea\_umathscanholes\_umathcharholes[#1]{#1}\_relax}
147
148 \_def\_umathcharholes{% holes in math alphabets:
149 [119893]{\_210E}[119965]{\_212C}[119968]{\_2130}[119969]{\_2131}%
150 [119971]{\_210B}[119972]{\_2110}[119975]{\_2112}[119976]{\_2133}[119981]{\_211B}%
151 [119994]{\_212F}[119996]{\_210A}[120004]{\_2134}%
152 [120070]{\_212D}[120075]{\_210C}[120076]{\_2111}[120085]{\_211C}[120093]{\_2128}%
153 [120122]{\_2102}[120127]{\_210D}[120133]{\_2115}[120135]{\_2119}
154 [120136]{\_211A}[120137]{\_211D}[120145]{\_2124}%
155 }
156 \_def\_umathrange#1#2{\_umathnumB=#2\_relax \_umathrangeA#1}
157 \_def\_umathrangeA#1-#2{\_umathnumA=#1\_relax
158 \_loop
159 \_umathcorr\_umathprepare\_umathnumB
160 \_Umathcode \_umathnumA = 7 1 \_umathcorr\_umathvalue{\_umathnumB}
161 \_ifnum\_umathnumA<#2\_relax
162 \_advance\_umathnumA by 1 \_advance\_umathnumB by 1
163 \_repeat
164 }
165 \_def\_umathrangeGREEK{\_begingroup
166 \_lccode`A="0391 \_lccode`Z="03A9
167 \_lowercase{\_endgroup \_umathrange{A-Z}}
168 \_def\_umathrangegreek{\_begingroup
169 \_lccode`A="03B1 \_lccode`Z="03D6
170 \_lowercase{\_endgroup \_umathrange{A-Z}}
171 \_def\_greekdef#1{\_ifx#1\_relax \_else
172 \_begingroup \_lccode`X=\_umathnumB \_lowercase{\_endgroup \_def#1{X}}%
173 \_advance\_umathnumB by 1
174 \_expandafter\_greekdef \_fi
175 }

```

## 2.15.2 Macros and codes set when $\backslash\loadmatfont$ is processed

The file `unimath-codes.opm` is loaded when the  $\backslash\loadmath$  is used. The macros here redefines globally all encoding dependent settings declared in the section 2.14.

unimath-codes.opm

```

3 \_codedecl \_ncharrmA {Uni math codes <2020-03-14>} % preloaded on demand by \loadmath

```

The control sequences for  $\backslash\alpha$ ,  $\backslash\beta$  etc are redefined here. The  $\backslash\alpha$  expands to the character with unicode "03B1, this is normal character  $\alpha$ . You can type it directly in your editor, if you know how to do this.

unimath-codes.opm

```

12 \_umathnumB="0391
13 \_greekdef \Alpha \Beta \Gamma \Delta \Epsilon \Zeta \Eta \Theta \Iota \Kappa
14 \Lambda \Mu \Nu \Xi \Omicron \Pi \Rho \varTheta \Sigma \Tau \Upsilon \Phi
15 \Chi \Psi \Omega \_relax
16
17 \_umathnumB="03B1
18 \_greekdef \alpha \beta \gamma \delta \varepsilon \zeta \eta \theta \iota \kappa
19 \lambda \mu \nu \xi \omicron \pi \rho \varsigma \sigma \tau \upsilon
20 \varphi \chi \psi \omega \varDelta \epsilon \vartheta \kappa \phi
21 \varrho \varpi \_relax

```

The math alphabets are declared here using the  $\backslash\_umathrange$  macro.



```

27 \_chardef\_ncharmA="A" \_chardef\_ncharma="a
28 \_chardef\_ncharbA="1D400 \_chardef\_ncharbfa="1D41A
29 \_chardef\_ncharitA="1D434 \_chardef\_ncharita="1D44E
30 \_chardef\_ncharbiA="1D468 \_chardef\_ncharbia="1D482
31 \_chardef\_ncharclA="1D49C \_chardef\_ncharcla="1D4B6
32 \_chardef\_ncharbcA="1D4D0 \_chardef\_ncharbca="1D4EA
33 \_chardef\_ncharfrA="1D504 \_chardef\_ncharfra="1D51E
34 \_chardef\_ncharbrA="1D56C \_chardef\_ncharbra="1D586
35 \_chardef\_ncharbbA="1D538 \_chardef\_ncharbba="1D552
36 \_chardef\_ncharsnA="1D5A0 \_chardef\_ncharsna="1D5BA
37 \_chardef\_ncharbsA="1D5D4 \_chardef\_ncharbsa="1D5EE
38 \_chardef\_ncharsiA="1D608 \_chardef\_ncharsia="1D622
39 \_chardef\_ncharsxA="1D63C \_chardef\_ncharsxa="1D656
40 \_chardef\_ncharttA="1D670 \_chardef\_nchartta="1D68A
41
42 \_protected\_def\_rmvariables {\_umathrange{A-Z}\_ncharmA \_umathrange{a-z}\_ncharma}
43 \_protected\_def\_bfvariables {\_umathrange{A-Z}\_ncharbfa \_umathrange{a-z}\_ncharbfa}
44 \_protected\_def\_nitvariables {\_umathrange{A-Z}\_ncharitA \_umathrange{a-z}\_ncharita}
45 \_protected\_def\_bivvariables {\_umathrange{A-Z}\_ncharbiA \_umathrange{a-z}\_ncharbia}
46 \_protected\_def\_calvariables {\_umathrange{A-Z}\_ncharclA \_umathrange{a-z}\_ncharcla}
47 \_protected\_def\_bcalvariables {\_umathrange{A-Z}\_ncharbcA \_umathrange{a-z}\_ncharbca}
48 \_protected\_def\_frakvariables {\_umathrange{A-Z}\_ncharfrA \_umathrange{a-z}\_ncharfra}
49 \_protected\_def\_bfrakvariables {\_umathrange{A-Z}\_ncharbrA \_umathrange{a-z}\_ncharbra}
50 \_protected\_def\_bbvariables {\_umathrange{A-Z}\_ncharbbA \_umathrange{a-z}\_ncharbba}
51 \_protected\_def\_sansvariables {\_umathrange{A-Z}\_ncharsnA \_umathrange{a-z}\_ncharsna}
52 \_protected\_def\_bsansvariables {\_umathrange{A-Z}\_ncharbsA \_umathrange{a-z}\_ncharbsa}
53 \_protected\_def\_isansvariables {\_umathrange{A-Z}\_ncharsiA \_umathrange{a-z}\_ncharsia}
54 \_protected\_def\_bisansvariables {\_umathrange{A-Z}\_ncharsxA \_umathrange{a-z}\_ncharsxa}
55 \_protected\_def\_ttvariables {\_umathrange{A-Z}\_ncharttA \_umathrange{a-z}\_nchartta}
56
57 \_chardef\_greekmA="0391 \_chardef\_greekma="03B1
58 \_chardef\_greekbfa="1D6A8 \_chardef\_greekbfa="1D6C2
59 \_chardef\_greekitA="1D6E2 \_chardef\_greekita="1D6FC
60 \_chardef\_greekbiA="1D71C \_chardef\_greekbia="1D736
61 \_chardef\_greeksnA="1D756 \_chardef\_greekсна="1D770
62 \_chardef\_greeksiA="1D790 \_chardef\_greekсия="1D7AA
63
64 \_protected\_def\_nitgreek {\_umathrangeGREEK\_greekmA \_umathrangegreek\_greekita}
65 \_protected\_def\_rmgreek {\_umathrangeGREEK\_greekmA \_umathrangegreek\_greekma}
66 \_protected\_def\_bfgreek {\_umathrangeGREEK\_greekbfa \_umathrangegreek\_greekbfa}
67 \_protected\_def\_bigreek {\_umathrangeGREEK\_greekbfa \_umathrangegreek\_greekbia}
68 \_protected\_def\_sansgreek {\_umathrangeGREEK\_greekсна \_umathrangegreek\_greekсна}
69 \_protected\_def\_isansgreek {\_umathrangeGREEK\_greekсия \_umathrangegreek\_greekсия}
70
71 % Another possibility (slanted capitals in \nitgreek, \bigreek, \isansgreek):
72 %\_protected\_def\_nitgreek {\_umathrangeGREEK\_greekitA \_umathrangegreek\_greekita}
73 %\_protected\_def\_rmgreek {\_umathrangeGREEK\_greekmA \_umathrangegreek\_greekma}
74 %\_protected\_def\_bfgreek {\_umathrangeGREEK\_greekbfa \_umathrangegreek\_greekbfa}
75 %\_protected\_def\_bigreek {\_umathrangeGREEK\_greekbiA \_umathrangegreek\_greekbia}
76 %\_protected\_def\_sansgreek {\_umathrangeGREEK\_greekсна \_umathrangegreek\_greekсна}
77 %\_protected\_def\_isansgreek {\_umathrangeGREEK\_greekсия \_umathrangegreek\_greekсия}
78
79 \_chardef\_digitrm0="0
80 \_chardef\_digitbf0="1D7CE
81 \_chardef\_digitbb0="1D7D8
82 \_chardef\_digitsn0="1D7E2
83 \_chardef\_digitbs0="1D7EC
84 \_chardef\_digittt0="1D7F6
85
86 \_protected\_def\_rmdigits {\_umathrange{0-9}\_digitrm0}
87 \_protected\_def\_bfdigits {\_umathrange{0-9}\_digitbf0}
88 \_protected\_def\_bbdigits {\_umathrange{0-9}\_digitbb0}
89 \_protected\_def\_sansdigits {\_umathrange{0-9}\_digitsn0}
90 \_protected\_def\_bsansdigits {\_umathrange{0-9}\_digitbs0}
91 \_protected\_def\_ttdigits {\_umathrange{0-9}\_digittt0}

```

The `\cal`, `\bbchar`, `\frak`, `\script` and the `\rm`, `\bf`, `\it`, `\bi`, `\tt` are defined here. Their “8bit definitions” from the file `math-preload.opm` (section 2.13) are removed.

You can redefine them again if you need different behavior (for example you don't want to use sans serif bold in math). What to do:

```
\_protected\_def\_bf {\_tryloadbf\_tenbf \_inmath{\_bfvariables\_bfgreek\_bfdigits}}
\_protected\_def\_bi {\_tryloadbi\_tenbi \_inmath{\_bivvariables\_bigreek\_bfdigits}}
\_public \bf \bi ;
```

`\_inmath {⟨cmds⟩}` applies `⟨cmds⟩` only in math mode.

```
unimath-codes.opm
109 \_protected\_def\_inmath#1{\_relax \_ifmode#1\_fi} % to keep off \loop processing in text mode
110
111 % You can redefine these macros to follow your wishes.
112 % For example you need upright lowercase greek letters, you don't need
113 % \bf and \bi behaves as sans serif in math, ...
114
115 \_protected\_def\_rm {\_tryloadrm \_tenrm \_inmath{\_rmvariables \_rmdigits}}
116 \_protected\_def\_it {\_tryloadit \_tenit \_inmath{\_nitvariables}}
117 \_protected\_def\_bf {\_tryloadbf \_tenbf \_inmath{\_bsansvariables \_sansgreek \_bsansdigits}}
118 \_protected\_def\_bi {\_tryloadbi \_tenbi \_inmath{\_bisansvariables \_isangreek \_bsansdigits}}
119 \_protected\_def\_tt {\_tryloadtt \_tentt \_inmath{\_ttvariables \_ttdigits}}
120 \_protected\_def\_bbchar {\_bbvariables \_bbdigits}
121 \_protected\_def\_cal {\_calvariables}
122 \_protected\_def\_frak {\_frakvariables}
123 \_protected\_def\_misans {\_isansvariables \_isangreek \_sansdigits}
124 \_protected\_def\_mbisans {\_bisansvariables \_isangreek \_bsansdigits}
125 \_protected\_def\_script {\_rmvariables \_fam4 }
126
127 \_public \rm \it \bf \bi \tt \bbchar \cal \frak \misans \mbisans \script ;
```

Each Unicode slot carries information about math type. This is saved in the file `mathclass.txt` which is copied to `mathclass.opm`. The file has the following format:

```
mathclass.opm
70 002E;P
71 002F;B
72 0030..0039;N
73 003A;P
74 003B;P
75 003C;R
76 003D;R
77 003E;R
78 003F;P
79 0040;N
80 0041..005A;A
81 005B;O
82 005C;B
83 005D;C
84 005E;N
85 005F;N
```

We have to read this information and convert it to the `\Umathcodes`.

```
unimath-codes.opm
137 \_begingroup % \input mathclass.opm (which is a copy of MathClass.txt):
138 \_def\_p#1;#2{\_edef\_tmp{\_pB#2}\_ifx\_tmp\_empty \_else\_pA#1...\_end#2\_fi}
139 \_def\_pA#1..#2..#3\_end#4{%
140 \_ifx\_relax#2\_relax \_pset{"#1}{#4}\_else
141 \_umathnumA="#1
142 \_loop
143 \_pset{\_umathnumA}{#4}%
144 \_ifnum\_umathnumA<"#2 \_advance\_umathnumA by1
145 \_repeat
146 \_fi
147 }
148 \_def\_pB#1{\_if#1L1\_fi \_if#1B2\_fi \_if#1V2\_fi \_if#1R3\_fi \_if#1N0\_fi \_if#1U0\_fi
149 \_if#1F0\_fi \_if#1O4\_fi \_if#1C5\_fi \_if#1P6\_fi \_if#1A7\_fi}
150 \_def\_pset#1#2{\_global\_Umathcode#1=\_tmp\_space 1 #1\_relax
151 \_if#20\_global\_Udelcode#1=1 #1\_relax\_fi
152 \_if#2C\_global\_Udelcode#1=1 #1\_relax\_fi
153 \_if#2F\_global\_Udelcode#1=1 #1\_relax\_fi
154 }
155 \_catcode`#=14
```

```

156 \_everypar={\_setbox0=\_lastbox \_par \_p}
157 \_input mathclass.opm
158 \_endgroup

```

Each math symbol has its declaration in the file `unicode-math-table.tex` which is copied to `unimath-table.opm`. The file has following format:

```

70 \UnicodeMathSymbol{"00397}{\mupEta           }\{\mathalpha}{capital eta, greek}%
71 \UnicodeMathSymbol{"00398}{\mupTheta        }\{\mathalpha}{capital theta, greek}%
72 \UnicodeMathSymbol{"00399}{\mupIota         }\{\mathalpha}{capital iota, greek}%
73 \UnicodeMathSymbol{"0039A}{\mupKappa        }\{\mathalpha}{capital kappa, greek}%
74 \UnicodeMathSymbol{"0039B}{\mupLambda       }\{\mathalpha}{capital lambda, greek}%
75 \UnicodeMathSymbol{"0039C}{\mupMu           }\{\mathalpha}{capital mu, greek}%
76 \UnicodeMathSymbol{"0039D}{\mupNu           }\{\mathalpha}{capital nu, greek}%
77 \UnicodeMathSymbol{"0039E}{\mupXi           }\{\mathalpha}{capital xi, greek}%
78 \UnicodeMathSymbol{"0039F}{\mupOmicron      }\{\mathalpha}{capital omicron, greek}%
79 \UnicodeMathSymbol{"003A0}{\mupPi            }\{\mathalpha}{capital pi, greek}%
80 \UnicodeMathSymbol{"003A1}{\mupRho          }\{\mathalpha}{capital rho, greek}%
81 \UnicodeMathSymbol{"003A3}{\mupSigma        }\{\mathalpha}{capital sigma, greek}%
82 \UnicodeMathSymbol{"003A4}{\mupTau          }\{\mathalpha}{capital tau, greek}%
83 \UnicodeMathSymbol{"003A5}{\mupUpsilon      }\{\mathalpha}{capital upsilon, greek}%
84 \UnicodeMathSymbol{"003A6}{\mupPhi          }\{\mathalpha}{capital phi, greek}%
85 \UnicodeMathSymbol{"003A7}{\mupChi          }\{\mathalpha}{capital chi, greek}%

```

We have to read this information and convert it to the Unicode math codes.

```

167 \_begingroup % \_input unimath-table.opm (it is a copy of unicode-math-table.tex):
168 \_def\UnicodeMathSymbol #1#2#3#4{%
169     \_global\_Umathcharnumdef#2=\_Umathcodenum#1\_relax
170     \_ifx#3\_mathopen \_gdef#2{\\_Udelimter 4 1 #1 }\_fi
171     \_ifx#3\_mathclose \_gdef#2{\\_Udelimter 5 1 #1 }\_fi
172     \_ifx#3\_mathaccent \_gdef#2{\\_Umathaccent fixed 7 1 #1 }\_fi
173 }
174 \_input unimath-table.opm
175 \_endgroup

```

Many special characters must be declared with care...

```

181 \_global\_Udelcode`<=1 "027E8 % these characters have different meaning
182 \_global\_Udelcode`>=1 "027E9 % as normal and as delimiter
183
184 \_nitgreek \_nitvariables \_rmdigits % default setting
185
186 \_Umathcode `~ = 2 1 "2212
187 %\_Umathcode` : = 3 1 "3A % mathclass defines it as 6 1 "3A (punctuation)
188 \_let\{=\_lbrace \_let\}=\_rbrace
189
190 \_protected\_def \_sqrt {\\_Uradical 1 "0221A }
191 \_protected\_def \_cuberoot {\\_Uradical 1 "0221B }
192 \_protected\_def \_fourthroot {\\_Uradical 1 "0221C }
193
194 \_public \_sqrt \_cuberoot \_fourthroot ;
195
196 \_def\_intwithnolimits#1#2 {\_ifx#1\_relax \_else
197     \_ea\_let\_csname\_csstring#1op\_endcsname=#1%
198     \_ea\_def\_ea #1\_ea{\_csname\_csstring#1op\_endcsname \_nolimits}%
199     \_bgroup \_lccode`~=#2 \_lowercase{\_egroup \_mathcode`~="8000 \_let ~=#1}%
200     \_ea \_intwithnolimits \_fi
201 }
202 \_intwithnolimits \_int "0222B \_iint "0222C \_iiint "0222D
203 \_oint "0222E \_oiint "0222F \_oiint "02230
204 \_intclockwise "02231 \_varointclockwise "02232 \_ointctrclockwise "02233
205 \_sumint "02A0B \_iiint "02A0C \_intbar "02A0D \_intBar "02A0E \_fint "02A0F
206 \_pointint "02A15 \_sqint "02A16 \_intlarhk "02A17 \_intx "02A18
207 \_intcap "02A19 \_intcup "02A1A \_upint "02A1B \_lowint "02A1C \_relax "0
208
209 \_protected\_def \_vert {\\_Udelimter 0 1 "07C }
210 \_protected\_def \_Vert {\\_Udelimter 0 1 "02016 }
211 \_protected\_def \_Vvert {\\_Udelimter 0 1 "02980 }
212

```

```

213 \protected\def \overbrace #1{\mathop {\Umathaccent 7 1 "023DE{#1}}\limits}
214 \protected\def \underbrace #1{\mathop {\Umathaccent bottom 7 1 "023DF{#1}}\limits}
215 \protected\def \overparen #1{\mathop {\Umathaccent 7 1 "023DC{#1}}\limits}
216 \protected\def \underparen #1{\mathop {\Umathaccent bottom 7 1 "023DD{#1}}\limits}
217 \protected\def \overbracket #1{\mathop {\Umathaccent 7 1 "023B4{#1}}\limits}
218 \protected\def \underbracket #1{\mathop {\Umathaccent bottom 7 1 "023B5{#1}}\limits}
219
220 \public \overbrace \underbrace \overparen \underparen \overbracket \underbracket ;
221
222 \protected\def \widehat {\Umathaccent 7 1 "00302 }
223 \protected\def \widetilde {\Umathaccent 7 1 "00303 }
224 \protected\def \overleftharpoon {\Umathaccent 7 1 "020D0 }
225 \protected\def \overrightharpoon {\Umathaccent 7 1 "020D1 }
226 \protected\def \overleftarrow {\Umathaccent 7 1 "020D6 }
227 \protected\def \overrightarrow {\Umathaccent 7 1 "020D7 }
228 \protected\def \overleftrightharpoon {\Umathaccent 7 1 "020E1 }
229
230 \mathchardef \ldotp="612E
231 \let \lvert=\Vert
232 \mathcode`\_="8000

```

Aliases are declared here. They are names not mentioned in the `unimath-table.opm` file but commonly used in  $\text{\TeX}$ .

`unimath-codes.opm`

```

239 \let \setminus=\smallsetminus
240 \let \diamond=\smwhtdiamond
241 \let \bullet=\smbkcircle
242 \let \circ=\vysmwhtcircle
243 \let \bigcirc=\mdlghwhtcircle
244 \let \to=\rightarrow
245 \let \le=\leq
246 \let \ge=\geq
247 \let \ne=\neq
248 \protected\def \triangle {\mathord{\bigtriangleup}}
249 \let \emptyset=\varnothing
250 \let \hbar=\hslash
251 \let \land=\wedge
252 \let \lor=\vee
253 \let \owns=\ni
254 \let \gets=\leftarrow
255 \let \mathring=\ocirc
256 \let \lnot=\neg
257 \let \longdivision=\longdivisionsign
258 \let \backepsilon=\upbackepsilon
259 \let \eth=\matheth
260 \let \dbkarow=\dbkarrow
261 \let \drbkarow=\drbkarow
262 \let \hksearrow=\hksearrow
263 \let \hkswarrow=\hkswarrow
264
265 \let \upalpha=\mupalpha
266 \let \upbeta=\mupbeta
267 \let \upgamma=\mupgamma
268 \let \updelta=\mupdelta
269 \let \upepsilon=\mupvarepsilon
270 \let \upvarepsilon=\mupvarepsilon
271 \let \upzeta=\mupzeta
272 \let \upeta=\mupeta
273 \let \uptheta=\muptheta
274 \let \upiota=\mupiota
275 \let \upkappa=\mupkappa
276 \let \uplambda=\muplambda
277 \let \upmu=\mupmu
278 \let \upnu=\mupnu
279 \let \upxi=\mupxi
280 \let \upomicron=\mupomicron
281 \let \uppi=\muppi
282 \let \uprho=\muprho
283 \let \upvarrho=\mupvarrho

```

```

284 \_let \upvarsigma=\mupvarsigma
285 \_let \upsigma=\mupsigma
286 \_let \uptau=\muptau
287 \_let \upupsilon=\mupupsilon
288 \_let \upvarphi=\mupvarphi
289 \_let \upchi=\mupchi
290 \_let \uppsi=\muppsi
291 \_let \upomega=\mupomega
292 \_let \upvartheta=\mupvartheta
293 \_let \upphi=\mupphi
294 \_let \upvarpi=\mupvarpi

```

The `\not` macro is redefined here.

unimath-codes.opm

```

300 \_protected\_def\_not#1{%
301   \_ifcsname\_not!\_csstring#1\_endcsname\_csname\_not!\_csstring#1\_endcsname
302   \_else\_mathrel{\_mathord{\_rlap{\_kern1pt/}}\_mathord{#1}}}%
303   \_fi
304 }
305 \_def\_negationof #1#2{\_ea\_let\_csname\_not!\_csstring#1\_endcsname=#2}
306 \_negationof = \neq
307 \_negationof < \less
308 \_negationof > \ngtr
309 \_negationof \gets \nleftarrow
310 \_negationof \simeq \nsime
311 \_negationof \equal \ne
312 \_negationof \le \nleq
313 \_negationof \ge \ngeq
314 \_negationof \greater \ngtr
315 \_negationof \forksnot \forks
316 \_negationof \in \notin
317
318 \_public \not ;

```

Newly declared public control sequences are used in internal macros by OpTeX. We need to get new meanings of these control sequences in private name space.

unimath-codes.opm

```

326 \_private
327 \ldotp \cdotp \bullet \triangleleft \triangleright \mapstochar \rightarrow
328 \prime \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
329 \relbar \rightarrow \relbar \rightarrow \Leftarrow \mapstochar
330 \longrightarrow \Longleftarrow \vdots \ddots ;

```

### 2.15.3 A few observations

You can combine more fonts in math, if you register them to another math families (5, 6, 7, etc.) in the `\normalmath` macro.

The default value of `\normalmath` shows a combination of base Unicode Math font with 8bit Math font at family 4. See definition of the `\script` macro where `\fam4` is used. Of course, we need to set `\rmvariables` too, because 8bit font accepts only codes less than 255.

XITSmath-bold needs correction: the norm symbol  $||x||$  is missing here. So, you can define:

```

\def\_boldmath{%
  \loadumathfamily 1 {[xitsmath-bold]}{} % Base font
  \loadmathfamily 4 rsfs % script
  \loadumathfamily 5 {[xitsmath-regular]}{}
  \def\|\{\Udelimter 0 5 "02016 %} % norm delimiter from family 5
  \setmathdimens
}

```

## 2.16 Scaling fonts in document (high-level macros)

These macros are documented in section 1.3.2 from user point of view.

fonts-opmac.opm

```

3 \_codedecl \typsize {Font managing macros from OPmac <2020-04-28>} % loaded in format

```

**\typosize** [ $\langle font-size \rangle / \langle baseline-skip \rangle$ ] sets given parameters. It sets text font size by the **\setfontsize** macro and math font sizes by setting internal macros **\\_sizemtext**, **\\_sizemscript** and **\\_sizemsscript**. It uses common concept font theses sizes: 100 %, 70 % and 50 %. The **\\_setmainvalues** sets the parameters as main values when the **\typosize** is called first.

fonts-opmac.opm

```

15 \_protected\_def \_typosize [#1/#2]{%
16   \_textfontsize{#1}\_mathfontsize{#1}\_setbaselineskip{#2}%
17   \_setmainvalues \_ignorespaces
18 }
19 \_protected\_def \_textfontsize #1{\_if$#1$\_else \_setfontsize{at#1\_ptunit}\_fi}
20
21 \_def \_mathfontsize #1{\_if$#1$\_else
22   \_tmpdim=#1\_ptunit
23   \_edef\_sizemtext{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
24   \_tmpdim=0.7\_tmpdim
25   \_edef\_sizemscript{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
26   \_tmpdim=#1\_ptunit \_tmpdim=0.5\_tmpdim
27   \_edef\_sizemsscript{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
28   \_fi
29 }
30 \_public \typosize ;

```

**\typoscale** [ $\langle font-factor \rangle / \langle baseline-factor \rangle$ ] scales font size and baselineskip by given factors in respect to current values. It calculates the **\typosize** parameters and runs the **\typosize**.

fonts-opmac.opm

```

38 \_protected\_def \_typoscale [#1/#2]{%
39   \_if$#1$\_def\_tmp{[/]\_else
40     \_settmpdim{#1}\_optsize
41     \_edef\_tmp{[\_ea\_ignorept\_the\_tmpdim/]\_fi
42   \_if$#2$\_edef\_tmp{\_tmp}\_else
43     \_settmpdim{#2}\_baselineskip
44     \_edef\_tmp{\_tmp \_ea\_ignorept\_the\_tmpdim}\_fi
45   \_ea\_typosize\_tmp
46 }
47 \_def\_settmpdim#1#2{%
48   \_tmpdim=#1pt \_divide\_tmpdim by1000
49   \_tmpdim=\_ea\_ignorept \_the#2\_tmpdim
50 }
51 \_public \typoscale ;

```

**\\_setbaselineskip** { $\langle baseline-skip \rangle$ } sets new **\baselineskip** and more values of registers which are dependent on the  $\langle baseline-skip \rangle$  including the **\strutbox**.

fonts-opmac.opm

```

59 \_def \_setbaselineskip #1{\_if$#1$\_else
60   \_tmpdim=#1\_ptunit
61   \_baselineskip=\_tmpdim \_relax
62   \_bigskipamount=\_tmpdim plus.33333\_tmpdim minus.33333\_tmpdim
63   \_medskipamount=.5\_tmpdim plus.16666\_tmpdim minus.16666\_tmpdim
64   \_smallskipamount=.25\_tmpdim plus.08333\_tmpdim minus.08333\_tmpdim
65   \_normalbaselineskip=\_tmpdim
66   \_jot=.25\_tmpdim
67   \_maxdepth=.33333\_tmpdim
68   \_setbox\_strutbox=\_hbox{\_vrule height.709\_tmpdim depth.291\_tmpdim width0pt}%
69   \_fi
70 }

```

**\\_setmainvalues** sets the current font size and **\baselineskip** values to the **\mainfosize** and **\mainbaselineskip** registers. It redefines itself in order to set the main values only first.

**\scalemain** returns to these values if they were set. Else they are set to 10/12pt.

fonts-opmac.opm

```

81 \_newskip \_mainbaselineskip \_mainbaselineskip=0pt \_relax
82 \_newdimen \_mainfosize \_mainfosize=0pt
83
84 \_def\_setmainvalues {%
85   \_mainbaselineskip=\_baselineskip
86   \_mainfosize=\_optsize
87   \_topskip=\_mainfosize \_splittopskip=\_topskip
88   \_ifmmode \_else \_bf \_it \_bi \_rm \_fi % load all basic variants of the family

```



```

89 \normalmath % load fonts if \typsize is running first
90 \let \setmainvalues=\setmainvaluesL
91 }
92 \def\setmainvaluesL {\ifmode \normalmath \else
93 \rm \everymath={\normalmath}\everydisplay={\normalmath}\fi}
94 \def\scalemain {%
95 \ifdim \mainfosize=0pt
96 \mainfosize=10pt \mainbaselineskip=12pt
97 \let \setmainvalues=\setmainvaluesL
98 \fi
99 \optsize=\mainfosize \baselineskip=\mainbaselineskip
100 }
101 \public \scalemain \mainfosize \mainbaselineskip ;

```

`\thefontsize` [*size*] and `\thefontscale` [*factor*] do modification of the size of the current font. They are implemented by the `\newcurrfontsize` macro.

fonts-opmac.opm

```

109 \protected\def\thefontsize[#1]{\if$#1$\else
110 \tmpdim=#1\ptunit
111 \newcurrfontsize{at\tmpdim}%
112 \fi
113 \ignorespaces
114 }
115 \protected\def\thefontscale[#1]{\ifx$#1$\else
116 \tmpdim=#1pt \divide\tmpdim by1000
117 \tmpdim=\ea\ea\ea\ignorept \pdffontsize\font \tmpdim
118 \newcurrfontsize{at\tmpdim}%
119 \fi
120 \ignorespaces
121 }
122 \public \thefontsize \thefontscale ;

```

`\em` keeps the weight of the current variant and switches roman ↔ italic. It adds the italic correction by the `\_additcorr` and `\_afteritcorr` macros. The second does not add italic correction if the next character is dot or comma.

fonts-opmac.opm

```

131 \protected\def\em {%
132 \ea\ifx \the\font \tenit \_additcorr \rm \else
133 \ea\ifx \the\font \tenbf \_bi\_aftergroup\_afteritcorr\else
134 \ea\ifx \the\font \tenbi \_additcorr \bf \else
135 \it \_aftergroup\_afteritcorr\fi\fi\fi
136 }
137 \def\_additcorr{\ifdim\_lastskip>0pt
138 \skip0=\lastskip \unskip\italcorr \hskip\skip0 \else\italcorr \fi}
139 \def\_afteritcorr{\futurelet\_next\_afteritcorrA}
140 \def\_afteritcorrA{\ifx\_next.\_else\ifx\_next,\_else \italcorr \fi\fi}
141 \let\_italcorr=\/

```

The `\boldify` macro does `\let\it\bi` and `\let\normalmath=\boldmath`.

fonts-opmac.opm

```

147 \protected\def \boldify {%
148 \let \setmainvalues=\setmainvaluesL
149 \let\it=\bi \let\rm=\bf \let\normalmath=\boldmath \bf
150 }
151 \public \em \boldify ;

```

We need to use a font selector for default pagination. Because we don't know what default font size will be selected by the user, we use this `\rmfixed` macro. It sets the `\rm` font from default font size (declared by first `\typsize` command and redefines itself to be only the font switch for next pages.

fonts-opmac.opm

```

161 \def \rmfixed {% used in default \footline
162 {\ifdim\mainfosize=0pt \mainfosize=10pt \fi
163 \fontdef\rmfixed{\setfontsize{at\mainfosize}\resetmod\rm}%
164 \global\let\rmfixed=\rmfixed} % next use will be font switch only
165 \rmfixed
166 }
167 \let \rmfixed = \tenrm % user can redefine it

```

## 2.17 Output routine

The output routine `\_optexoutput` is similar as in plain T<sub>E</sub>X. It does:

- `\_begoutput` which does:
  - increments `\gpageno`,
  - prints `\_Xpage{\gpageno}{\pageno}` to the `.ref` file (if `\openref` is active),
  - calculates `\hoffset`,
  - sets local meaning of macros used in headlines/footlines (see `\regmacro`).
- `\shipout\_completepage`, which is `\vbox` of –
  - background box, if `\pgbackground` is non-empty,
  - headline box by `\_makeheadline`, if the `\headline` is nonempty,
  - `\vbox` to `\vsize` of `\_pagecontents` which consists of –
    - `\_pagedest`, the page destination `pg:\gpageno` for hyperlinks is created here,
    - `\topins` box if non-empty (from `\topinserts`),
    - `\box255` with completed vertical material from main vertical mode,
    - `\_footnoterule` and `\footins` box if nonempty (from `\fnote`, `\footnote`),
    - `\pgbottomskip` (default is 0 pt).
  - footline box by `\_makefootline`, if the `\footline` is nonempty
- `\_endoutput` which does:
  - increments `\pageno` using `\advancepageno`
  - runs output routine repeatedly if `\dosupereject` is activated.

```
3 \_codedecl \nopagenumbers {Output routine <2020-03-28>} % preloaded in format
```

output.opm

`\_optexoutput` is default output routine. You can create another...

```
9 \_output={\_optexoutput}
10 \_def \_optexoutput{\_begoutput \_shipout\_completepage \_endoutput}
```

output.opm

Default `\_begoutput` and `\_endoutput` is defined. If you need another functionality implemented in the output routine, you can `\addto\_begoutput{...}` or `\addto\_endoutput{...}`. The settings here is local in the `\output` group.

The `\_preppoffsets` can set `\hoffset` differently for left or right page. It is re-defined by the `\margins` macro..

The `\_regmark` tokens list includes accumulated #2 from the `\regmacro`. Logos and another macros are re-defined here (locally) for their usage in headlines or footlines.

```
26 \_def \_begoutput{\_incr\_gpageno
27 \_immediate\_wref\_Xpage{\\_the\_gpageno}{\_folio}}%
28 \_preppoffsets \_the\_regmark} %
29 \_def \_endoutput{\_advancepageno
30 {\_globaldefs=1 \_the\_nextpages \_nextpages={}}%
31 \_ifnum\_outputpenalty>-20000 \_else\_dosupereject\_fi
32 }
33 \_def \_preppoffsets {}
```

output.opm

`\gpageno` counts pages from one in whole document

```
39 \_newcount\_gpageno
40 \_public \gpageno ;
```

output.opm

The `\_completepage` is similar what plain T<sub>E</sub>X does in its output routine. New is only `\_backgroundbox`. It is `\vbox` with zero height with its contents (from `\pgbackground`) laped down. It is shifted directly to the left-upper corner of the paper.

The `\_ensureblack` sets the typesetting of its parameter locally to `\Black` color. We needn't do this if colors are never used in the document. So, default value of the `\_ensureblack` macro is empty. But first usage of color macros in the document re-defines `\_ensureblack`. See the section 2.19 for more details.

```
55 \_def\_completepage{\_vbox{%
56 \_istoksemtty \pgbackground
57 \_iffalse \_ensureblack{\_backgroundbox{\_the\_pgbackground}}\_nointerlineskip \_fi
58 \_ensureblack{\_makeheadline}%
```

output.opm

```

59 \vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
60 \ensureblack{\_makefootline}}%
61 }
62 \def \_ensureblack #1{#1} % will be re-defined by color macros
63 \_let \_openfnoteSTACK = \_relax % will be re-defined by color macros
64 \def \_backgroundbox #1{\_moveleft\_hoffset\vbox to0pt{\_kern-\_voffset #1\_vss}}

```

`\_makeheadline` creates `\vbox to0pt` with its contents (the `\headline`) shifted by `\headlinedist` up.

output.opm

```

71 \def\_makeheadline {\_istoksempy \_headline \_iffalse
72 \vbox to0pt{\_vss
73 \baselineskip=\_headlinedist \lineskiplimit=-\_maxdimen
74 \line{\_the\_headline}\_hbox{}}\_nointerlineskip
75 \_fi
76 }

```

The `\_makefootline` appends the `\footline` to the page-body box.

output.opm

```

82 \def\_makefootline{\_istoksempy \_footline \_iffalse
83 \baselineskip=\_footlinedist
84 \lineskiplimit=-\_maxdimen \line{\_the\_footline}
85 \_fi
86 }

```

The `\_pagecontents` is similar as in plain T<sub>E</sub>X. The only difference is that the `\_pagedest` is inserted at the top of `\_pagecontents` and `\_ensureblack` is applied to the `\topins` and `\footins` material.

The `\_footnoterule` is defined here.

output.opm

```

95 \def\_pagecontents{\_pagedest % destination of the page
96 \_ifvoid\_topins \_else \_ensureblack{\_unvbox\_topins}\_fi
97 \_dimen0=\_dp255 \_unvbox255 % open up \box255
98 \_ifvoid\_footins \_else % footnote info is present
99 \_vskip\_skip\_footins
100 \_ensureblack{\_footnoterule \_openfnoteSTACK \_unvbox\_footins}\_fi
101 \_kern-\_dimen0 \_vskip \_pgbottomskip
102 }
103 \def \_pagedest {\_def\_destheight{25pt}\_dest[pg:\_the\_pageno]}
104 \def \_footnoterule {\_kern-3pt \_hrule width 2truein \_kern 2.6pt }

```

`\pageno`, `\folio`, `\nopagenumbers`, `\advancepageno` and `\normalbottom` used in the context of the output routine from plain T<sub>E</sub>X is defined here. Only the `\raggedbottom` macro is defined differently. We use the `\pgbottomskip` register here which is set to 0pt by default.

output.opm

```

115 \countdef\_pageno=0 \_pageno=1 % first page is number 1
116 \def \_folio {\_ifnum\_pageno<0 \_romannumeral-\_pageno \_else \_number\_pageno \_fi}
117 \def \_nopagenumbers {\_footline={}}
118 \def \_advancepageno {%
119 \_ifnum\_pageno<0 \_global\_advance\_pageno by-1 \_else \_incr\_pageno \_fi
120 } % increase |pageno|
121 \def \_raggedbottom {\_topskip=\_dimexpr\_topskip plus60pt \_pgbottomskip=0pt plus1fil\_relax}
122 \def \_normalbottom {\_topskip=\_dimexpr\_topskip \_pgbottomskip=0pt\_relax}
123
124 \_public \pageno \folio \nopagenumbers \advancepageno \raggedbottom \normalbottom ;

```

Macros for footnotes are the same as in plain T<sub>E</sub>X. There is only one difference: `\vfootnote` is implemented as `\_opfootnote` with empty parameter #1. This parameter should do a local settings inside the `\footins` group and it does it when `\fnote` macro is used.

The `\_opfootnote` nor `\vfootnote` don't take the footnote text as a parameter. This is due to user can do catcode settings (like inline verbatim) in the footnote text. This idea is adapted from plain T<sub>E</sub>X.

The `\footnote` and `\footstrut` is defined as in plain T<sub>E</sub>X.

output.opm

```

137 \_newinsert\_footins
138 \def \_footnote #1{\_let\_osf=\_empty % parameter #2 (the text) is read later
139 \_ifhmode \_edef\_osf{\_spacefactor\_the\_spacefactor}\\_fi
140 #1\_osf\vfootnote{#1}}
141 \def \_vfootnote{\_opfootnote{}}
142 \def \_opfootnote #1#2{\_insert\_footins\_bgroup
143 \_interlinepenalty=\_interfootnotelinepenalty
144 \_leftskip=0pt \_rightskip=0pt \_spaceskip=0pt \_xspaceskip=0pt \_relax

```

```

145 \let\colorstackcnt=\fnotestack % special color stack for footnotes
146 #1\relax % local settings used by \fnote macro
147 \splittopskip=\ht\strutbox % top baseline for broken footnotes
148 \splitmaxdepth=\dp\strutbox \floatingpenalty=20000
149 \textindent{#2}\footstrut
150 \isnextchar \bgroup
151 {\bgroup \aftergroup\vfootA \afterassignment\ignorespaces \let\next={}\vfootB}%
152 }
153 \def\vfootA{\unskip\strut\isnextchar\colorstackpop\closefncolor\vfootF}
154 \def\vfootB #1{#1\unskip\strut\vfootF}
155 \def\vfootF{\egroup} % close \insert\footins\bgroup
156 \def\closefncolor#1{#1\isnextchar\colorstackpop\closefncolor\vfootF}
157 \def \footstrut {\vbox to\splittopskip{}}
158 \skip\footins=\bigskipamount % space added when footnote is present
159 \count\footins=1000 % footnote magnification factor (1 to 1)
160 \dimen\footins=8in % maximum footnotes per page
161 \public
162 \footins \footnote \vfootnote \footstrut ;

```

The `\topins` macros `\topinsert`, `\midinsert`, `\pageinsert`, `\endinsert` are the same as in plain  $\TeX$ . output.opm

```

170 \newinsert\topins
171 \newif\ifupage \newif\ifumid
172 \def \topinsert {\umidfalse \upagefalse \oins}
173 \def \midinsert {\umidtrue \oins}
174 \def \pageinsert {\umidfalse \upagetrue \oins}
175 \skip\topins=\zskip % no space added when a topinsert is present
176 \count\topins=1000 % magnification factor (1 to 1)
177 \dimen\topins=\maxdimen % no limit per page
178 \def \oins {\par \begingroup\setbox0=\vbox\bgroup} % start a \vbox
179 \def \endinsert {\par\egroup % finish the \vbox
180 \ifumid \dimen0=\ht0 \advance\dimen0 by\dp0 \advance\dimen0 by\baselineskip
181 \advance\dimen0 by\pagetotal \advance\dimen0 by-\pageshrink
182 \ifdim\dimen0>\pagegoal \umidfalse \upagefalse \fi \fi
183 \ifumid \bigskip \box0 \bigbreak
184 \else \insert \topins {\penalty100 % floating insertion
185 \splittopskip=0pt
186 \splitmaxdepth=\maxdimen \floatingpenalty=0
187 \ifupage \dimen0=\dp0
188 \vbox to\vsizel{\unvbox0 \kern-\dimen0}% depth is zero
189 \else \box0 \nobreak \bigskip \fi}\fi\endgroup}
190
191 \public \topins \topinsert \midinsert \pageinsert \endinsert ;

```

The `\draft` macro is an example of usage `\pgbackground` to create water color marks.

```

198 \def \draft {\pgbackground={\draftbox{\draftfont DRAFT}}}%
199 \fontdef\draftfont{\setfontsize{at10pt}\bf}%
200 \global\let\draftfont=\draftfont
201 }
202 \def \draftbox #1{\setbox0=\hbox{#1}%
203 \kern.5\vsizel\kern4.5\wd0
204 \hbox to0pt{\kern.5\hsizel\kern-1\wd0
205 \pdfsave \pdfrotate{55}\pdfscale{10}{10}%
206 \hbox to0pt{\localcolor\LightGrey \box0\hss}%
207 \pdfrestore
208 \hss}%
209 }
210 \public \draft ;

```

## 2.18 Margins

The `\margins` macro is documented in the section 1.2.1.

```

3 \codedecl \margins {Macros for margins setting <2020-03-14>} % preloaded in format

```

`\margins`/`<pg>` `<fmt>` (`<left>`), (`<right>`), (`<top>`), (`<bot>`) (`<unit>`) takes its parameters, does calculation and sets `\hoffset`, `\voffset`, `\hsizel` and `\vsizel` registers. Note that  $\text{Op}\TeX$  sets the page origin at the top left corner of the paper, not at the obscure position 1 in, 1 in. It is much more comfortable for macro writers.

```

13 \_newdimen\_pgwidth \_newdimen\_pgheight \_pgwidth=0pt
14 \_newdimen\_shiftoffset
15
16 \_def\_margins/#1 #2 (#3,#4,#5,#6)#7 {\_def\_tmp{#7}%
17   \_ifx\_tmp\_empty
18     \_opwarning{\_string\_margins: missing unit, mm inserted}\_def\_tmp{mm}\_fi
19   \_setpagedimens #2 % setting \_pgwidth, \_pgheight
20   \_ifdim\_pgwidth=0pt \_else
21     \_hoffset=0pt \_voffset=0pt
22     \_if$#3$_\_if$#4$_\_hoffset =\_dimexpr (\_pgwidth -\_hsize)/2 \_relax
23     \_else \_hoffset =\_dimexpr \_pgwidth -\_hsize - #4\_tmp \_relax % only right margin
24     \_fi
25     \_else \_if$#4$_\_hoffset = #3\_tmp \_relax % only left margin
26     \_else \_hsize =\_dimexpr \_pgwidth - #3\_tmp - #4\_tmp \_relax % left+right margin
27     \_hoffset = #3\_tmp \_relax
28     \_fi\_fi
29     \_if$#5$_\_if$#6$_\_voffset =\_dimexpr (\_pgheight -\_vsize)/2 \_relax
30     \_else \_voffset =\_dimexpr \_pgheight -\_vsize - #6\_tmp \_relax % only bottom margin
31     \_fi
32     \_else \_if$#6$_\_voffset = #5\_tmp \_relax % only top margin
33     \_else \_vsize =\_dimexpr \_pgheight - #5\_tmp - #6\_tmp \_relax % top+bottom margin
34     \_voffset = #5\_tmp \_relax
35     \_fi\_fi
36     \_if 1#1\_shiftoffset=0pt \_def\_preppoffsets{}\_else \_if 2#1% double-page layout
37       \_shiftoffset = \_dimexpr \_pgwidth -\_hsize -2\_hoffset \_relax
38       \_def\_preppoffsets{\_ifodd\_pageno \_else \_advance\_hoffset \_shiftoffset \_fi}%
39     \_else \_opwarning{use \_string\_margins/1 or \_string\_margins/2}%
40     \_fi\_fi\_fi
41 }
42 \_def\_setpagedimens{\_isnextchar{\\_setpagedimensB}{\_setpagedimensA}}
43 \_def\_setpagedimensA#1 {\_ifcsname\_pgs:#1\_endcsname
44   \_ea\_ea\_ea\_setpagedimensB \_csname\_pgs:#1\_ea\_endcsname\_space
45   \_else \_opwarning{page specification "#1" is undefined}\_fi}
46 \_def\_setpagedimensB (#1,#2)#3 {\_setpagedimensC\_pgwidth=#1:#3
47   \_setpagedimensC\_pgheight=#2:#3
48   \_pdfpagewidth=\_pgwidth \_pdfpageheight=\_pgheight
49 }
50 \_def\_setpagedimensC #1=#2:#3 {#1=#2\_ifx^#3\_tmp\_else#3\_fi\_relax\_truedimen#1}
51
52 \_public \margins ;

```

The common page dimensions are defined here.

```

58 \_sdef{pgs:a3}{(297,420)mm} \_sdef{pgs:a4}{(210,297)mm} \_sdef{pgs:a5}{(148,210)mm}
59 \_sdef{pgs:a3l}{(420,297)mm} \_sdef{pgs:a4l}{(297,210)mm} \_sdef{pgs:a5l}{(210,148)mm}
60 \_sdef{pgs:b5}{(176,250)mm} \_sdef{pgs:letter}{(8.5,11)in}

```

`\magscale` [*factor*] does `\mag=<factor>` and recalculates page dimensions to their true values.

```

67 \_def\_trueunit{}
68 \_def\_magscale[#1]{\_mag=#1\_def\_trueunit{true}%
69   \_ifdim\_pgwidth=0pt \_else \_truedimen\_pgwidth \_truedimen\_pgheight \_fi
70   \_truedimen\_pdfpagewidth \_truedimen\_pdfpageheight
71 }
72 \_def\_truedimen#1{\_ifx\_trueunit\_empty \_else#1=\_ea\_ignorept\_the#1truept \_fi}
73
74 \_public \magscale ;

```

## 2.19 Colors

The colors have different behavior than fonts. A marks (whatsits) with color information are stored into PDF output and T<sub>E</sub>X doesn't interpret them. The PDF viewer (or PDF interpreter in a printer) reads these marks and switches colors according to them. This is totally independent on T<sub>E</sub>X group mechanism. You can declare `\nolocalcolor` at the beginning of the document, if you want this behavior. In this case, if you set a color then you must to return back to black color using `\Black` manually.

By default, OpT<sub>E</sub>X sets `\localcolor`. It means that the typesetting returns back to a previous color at the end of current group, so you cannot write `\Black` explicitly. This is implemented using

`\aftergroup` feature. There is a limitation of this feature: when a color selector is used in a group of a box, which is saved by `\setbox`, then the activity or reconstruction of previous color are processed at `\setbox` time, not in the box itself. You must correct it by double group:

```
\setbox0=\hbox{\Red text} % bad: \Black is done after \setbox
\setbox0=\hbox{{\Red text}} % good: \Black is done after group inside the box
```

The implementation of colors is based on colorstack, so the current color can follow across more pages. It is not so obvious because PDF viewer (or PDF interpreter) manipulates with colors locally at each PDF page and it initializes each PDF page with black on white color.

Macros `\setcmykcolor{<C> <M> <Y> <K>}` or `\setrgbcolor{<R> <G> <B>}` or `\setgreycolor{<Grey>}` should be used in color selectors or user can specify these macros explicitly.

The color mixing processed by the `\colordef` is done in the subtractive color model CMYK. If the result has a component greater than 1 then all components are multiplied by a coefficient in order to maximal component is equal to 1.

You can move a shared amount of CMY components (i.e. their minimum) to the *K* component. This saves the color tonners and the result is more true. This should be done by `\useK` command at the end of a linear combination used in `\colordef`. For example

```
\colordef \myColor {.3\Green + .4\Blue \useK}
```

The `\useK` command exactly does:

$$\begin{aligned} k' &= \min(C, M, Y), \\ C &= (C - k') / (1 - k'), \quad M = (M - k') / (1 - k'), \quad Y = (Y - k') / (1 - k'), \\ K &= \min(1, K + k'). \end{aligned}$$

You can use minus instead plus in the linear combination in `\colordef`. The given color is subtracted in such case and the negative components are rounded to zero immediately. For example

```
\colordef \Color {\Brown-\Black}
```

can be used for removing black component from the color. You can use the `-\Black` trick after `\useK` command in order to remove grey components occurred during color mixing.

Finally, you can use `^` immediately preceeded before macro name of the color. Then the complementary color is used here.

```
\colordef\mycolor{\Grey+.6^\Blue} % the same as \colordef\mycolor{\Grey+.6\Yellow}
```

The `\rgbcolordef` can be used to mix colors in additive color model RGB. If `\onlyrgb` is declared, then `\colordef` works as `\rgbcolordef`.

If a CMYK to RGB or RGB to CMYK conversion is needed then the following simple formulae are used (ICC profiles are not supported):

CMYK to RGB:

$$R = (1 - C)(1 - K), \quad G = (1 - M)(1 - K), \quad B = (1 - Y)(1 - K).$$

RGB to CMYK:

$$K' = \max(R, G, B), \quad C = (K' - R) / K', \quad M = (K' - G) / K', \quad Y = (K' - B) / K', \quad K = 1 - K'.$$

The RGB to CMYK conversion is invoked when a color is declared using `\setrgbcolor` and it is used in `\colordef` or if it is printed when `\onlycmyk` is declared. The CMYK to RGB conversion is invoked when a color is declared using `\setcmykcolor` and it is used in `\rgbcolordef` or if it is printed when `\onlyrgb` is declared.

colors.opm

```
3 \_codedecl \colordef {Colors <2020-03-18>} % loaded in format
```

We declare internal boolean value `\_iflocalcolor` and do `\localcolor` as default.

colors.opm

```
10 \_newif \_iflocalcolor \_localcolortrue
11 \_protected\_def \_localcolor {\_localcolortrue}
12 \_protected\_def \_nolocalcolor {\_localcolorfalse}
13 \_public \localcolor \nolocalcolor ;
```



The basic colors in CMYK `\Blue` `\Red` `\Brown` `\Green` `\Yellow` `\Cyan` `\Magenta` `\Grey` `\LightGrey` `\White` and `\Black` are declared here.

```
colors.opm
```

```

22 \_def\Blue      {\_setcmykcolor{1 1 0 0}}
23 \_def\Red       {\_setcmykcolor{0 1 1 0}}
24 \_def\Brown     {\_setcmykcolor{0 0.67 0.67 0.5}}
25 \_def\Green     {\_setcmykcolor{1 0 1 0}}
26 \_def\Yellow    {\_setcmykcolor{0 0 1 0}}
27 \_def\Cyan      {\_setcmykcolor{1 0 0 0}}
28 \_def\Magenta   {\_setcmykcolor{0 1 0 0}}
29 \_def\Grey      {\_setcmykcolor{0 0 0 0.5}}
30 \_def\LightGrey {\_setcmykcolor{0 0 0 0.2}}
31 \_def\White     {\_setgreycolor{1}}
32 \_def\Black     {\_setgreycolor{0}}
```

By default, the `\setcmykcolor` `\setrgbcolor` and `\setgreycolor` macros with  $\langle\text{componets}\rangle$  parameter expand to `\setcolor{pdf-primitive}` using `\formatcmyk` or `\formatrgb` or `\formatgrey` expandable macros. For example `\setrgbcolor{1 0 0}` expands to `\setcolor{1 0 0 rg 1 0 0 RG}`. We set both types of colors (for lines (K or RG or G) and for fills (r or rg or g) together in the  $\langle\text{pdf-primitive}\rangle$  command. This is the reason why the `\fillstroke` uses both its parameters. If only fills are needed you can do `\def\fillstroke#1#2{#1}`. If only strokes are needed you can do `\def\fillstroke#1#2{#2}`.

```
colors.opm
```

```

47 \_def\_setcmykcolor#1{\_setcolor{\formatcmyk{#1}}}
48 \_def\_setrgbcolor#1{\_setcolor{\formatrgb{#1}}}
49 \_def\_setgreycolor#1{\_setcolor{\formatgrey{#1}}}
50 \_def\_formatcmyk#1{\_fillstroke{#1 k}{#1 K}}
51 \_def\_formatrgb#1{\_fillstroke{#1 rg}{#1 RG}}
52 \_def\_formatgrey#1{\_fillstroke{#1 g}{#1 G}}
53 \_def\_fillstroke#1#2{#1 #2}
54 \_public \setcmykcolor \setrgbcolor \setgreycolor ;
```

The `\onlyrgb` declaration redefines `\formatcmyk` in order it expands to its conversion to RGB  $\langle\text{pdf-primitive}\rangle$ . This conversion is done by the `\cmyktorgb` macro. Moreover, `\onlyrgb` re-defines three basic RGB colors for RGB color space and re-declares `\colordef` as `\rgbcolordef`. The `\onlycmyk` macro does a similar work, it re-defines `\formatrgb` macro. The Grey color space is unchanged and works in both main settings (RGB or CMYK) without collisions.

```
colors.opm
```

```

66 \_def\_onlyrgb{\_def\Red{\_setrgbcolor{1 0 0}}%
67   \_def\Green{\_setrgbcolor{0 1 0}}\_def\Blue{\_setrgbcolor{0 0 1}}%
68   \_let\_colordef=\rgbcolordef
69   \_def\_formatrgb##1{\_fillstroke{##1 rg}{##1 RG}}%
70   \_def\_formatcmyk##1{\_fillstroke{\cmyktorgb ##1 ; rg}{\cmyktorgb ##1 ; RG}}
71 \_def\_onlycmyk{\_def\_formatcmyk##1{\_fillstroke{##1 k}{##1 K}}%
72   \_def\_formatrgb##1{\_fillstroke{\rgbtocmyk ##1 ; k}{\rgbtocmyk ##1 ; K}}
73 \_public \onlyrgb \onlycmyk ;
```

The `\setcolor` macro redefines empty `\ensureblack` macro (used in output routine for headers and footers) to `\ensureblackA` which sets Black at the start of its parameter and returns to the current color at the end of its parameter.

The current color is saved into `\currentcolor` macro and colorstack is pushed. Finally, the `\colorstackpop` is initialized by `\aftergroup` if `\localcolor` is declared.

You can save current color to your macro by `\let\yourmacro=\currentcolor` and you can return to this color by the command `\setcolor\yourmacro`.

```
colors.opm
```

```

89 \_protected\_def \setcolor #1{\_global\_let\ensureblack=\ensureblackA
90   \_iflocalcolor \_edef\_currentcolor{#1}\_colorstackpush\_currentcolor
91   \_aftergroup\_colorstackpop
92   \_else \_xdef\_currentcolor{#1}\_colorstackset\_currentcolor \_fi
93 }
94 \_def\_pdfblackcolor{0 g 0 G}
95 \_edef\_currentcolor{\_pdfblackcolor}
96 \_def\_ensureblackA#1{\_global\_let\_opennotestack=\_opennotestackA
97   \_colorstackpush\_pdfblackcolor #1\_colorstackpop}
```

The colorstack is initialized here and the basic macros `\colorstackpush`, `\colorstackpop` and `\colorstackset` are defined here.

```

105 \mathchardef\colorstackcnt=0 % Implicit stack usage
106 \def\colorstackpush#1{\pdfcolorstack\colorstackcnt push{#1}}
107 \def\colorstackpop{\pdfcolorstack\colorstackcnt pop}
108 \def\colorstackset#1{\pdfcolorstack\colorstackcnt set{#1}}

```

We need to open a special color stack for footnotes, because footnotes can follow on next pages and their colors are independent on colors used in the main page-body. The `\_openfnestack` is defined as `\_openfnestackA` when the `\_setcolor` is used first. The `\_fnestack` is initialized in `\everyjob` because the initialization is not saved to the format.

```

119 %\mathchardef\_fnestack=\pdfcolorstackinit page {0 g 0 G} % must be in \everyjob
120 \def\_openfnestackA {\pdfcolorstack\_fnestack current}

```

We use lua codes for RGB to CMYK or CMYK to RGB conversions and for addition color components in the `\colordef` macro. The `\_rgbtocmyk`  $\langle R \rangle \langle G \rangle \langle B \rangle$  ; expands to  $\langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle$  and the `\_cmyktorgb`  $\langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle$  ; expands to  $\langle R \rangle \langle G \rangle \langle B \rangle$ . The `\_colorcrop`, `\_colordefFin` and `\_douseK` are auxiliary macros used in the `\colordef`. The `\_colorcrop` rescales color components in order to they are in  $[0,1]$  interval. The `\colordefFin` expands to the values accumulated in Lua code `color_C`, `color_M`, `color_Y` and `color_K`. The `\_douseK` applies `\useK` to CMYK components.

```

134 \def\_rgbtocmyk #1 #2 #3 ;{%
135   \_ea \_stripzeros \_detokenize \_ea{\_directlua{
136     local kr = math.max(#1,#2,#3)
137     if (kr==0) then
138       tex.print('0. 0. 0. 1 ;')
139     else
140       tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
141         (kr-#1)/kr, (kr-#2)/kr, (kr-#3)/kr, 1-kr))
142     end
143   }}
144 \def\_cmyktorgb #1 #2 #3 #4 ;{%
145   \_ea \_stripzeros \_detokenize \_ea{\_directlua{
146     local kr = 1-#4
147     tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
148       (1-#1)*kr, (1-#2)*kr, (1-#3)*kr))
149   }}
150 \def\_colorcrop{\_directlua{
151   local m=math.max(color_C, color_M, color_Y, color_K)
152   if (m>1) then
153     color_C=color_C/m color_M=color_M/m color_Y=color_Y/m color_K=color_K/m
154   end
155 }}
156 \def\_colordefFin{\_colorcrop \_ea \_stripzeros \_detokenize \_ea{\_directlua{
157   tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
158     color_C, color_M, color_Y, color_K))
159 }}
160 \def\_douseK{\_colorcrop \_directlua{
161   kr=math.min(color_C, color_M, color_Y)
162   if (kr>=1) then
163     color_C=0 color_M=0 color_Y=0 color_K=1
164   else
165     color_C=(color_C-kr)/(1-kr) color_M=(color_M-kr)/(1-kr)
166     color_Y=(color_Y-kr)/(1-kr) color_K=math.min(color_K+kr,1)
167   end
168 }}

```

We have a problem with the `%.3f` directive in Lua code. It prints trailed zeros: (0.300 instead desired 0.3) but we want to save PDF file space. The macro `\_stripzeros` removes these trailing zeros at expand processor level. So `\_stripzeros 0.300 0.400 0.560 ;` expands to `.3 .4 .56`.

```

177 \def\_stripzeros #1.#2 #3{\_ifx0#1\_else#1\_fi.\_stripzeroA #2 0 :%
178   \_ifx;#3\_else \_space \_ea\_stripzeros\_ea#3\_fi}
179 \def\_stripzeroA #10 #2:{\_ifx^#2^\_stripzeroC#1:\_else \_stripzeroB#1 0 :\_fi}
180 \def\_stripzeroB #10 #2:{\_ifx^#2^\_stripzeroC#1:\_else #1\_fi}
181 \def\_stripzeroC #1 #2:{#1}

```

The `\rgbcolordef` and `\cmykcolordef` use common macro `\_commoncolordef` with different first four parameters. The `\_commoncolordef`  $\langle selector \rangle \langle K \rangle \langle R \rangle \langle G \rangle \langle what-define \rangle \{ \langle data \rangle \}$  does the real work. It

initializes the Lua variables for summation. It expands  $\langle data \rangle$  in the group where color selectors have special meaning, then it adjusts the resulting string by `\replstring` and runs it. Example shows how the  $\langle data \rangle$  are processed:

```
input <data>: ".3\Blue + .6~\KhakiC \useK -\Black"
expanded to: ".3 !=K 1 1 0 0 +.6~=R .804 .776 .45 \_useK -!=G 0"
adjusted to: "\_addcolor .3!=K 1 1 0 0 \_addcolor .6!~R .804 .776 .45
              \_useK \_addcolor -1!=G 0"
and this is processed.
```

`\_addcolor`  $\langle coef \rangle$   $! \langle mod \rangle \langle type \rangle$  expands to `\_addcolor:  $\langle mod \rangle \langle type \rangle \langle coef \rangle$`  for example it expands to `\_addcolor:=K  $\langle coef \rangle$`  followed by one or three or four numbers (depending on  $\langle type \rangle$ ).  $\langle mod \rangle$  is = (use as is) or ~ (use complementary color).  $\langle type \rangle$  is K for CMYK, R for RGB and G for GREY color space. Uppercase  $\langle type \rangle$  informs that `\cmkykcolordef` is processed and lowercase  $\langle type \rangle$  informs that `\rgbcolordef` is processed. All variants of commands `\_addcolor:  $\langle mod \rangle \langle type \rangle$`  are defined. All of them expand to `\_addcolorA  $\langle v1 \rangle \langle v2 \rangle \langle v3 \rangle \langle v4 \rangle$`  which adds the values of Lua variables. The `\rgbcolordef` uses `\_addcolorA  $\langle R \rangle \langle G \rangle \langle B \rangle 0$`  and `\cmkykcolordef` uses `\_addcolorA  $\langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle$` . So the Lua variable names are a little confusing when `\rgbcolordef` is processed.

Next, `\_commoncolordef` saves resulting values from Lua to `\_tmpb` using `\_colordefFin`. If `\rgbcolordef` is processed, then we must to remove the last  $\langle K \rangle$  component which is in the format .0 in such case. The `\_stripK` macro does it. Finally, the  $\langle what-define \rangle$  is defined as  $\langle selector \rangle \{ \langle expanded\_tmpb \rangle \}$ , for example `\_setcmkykclor{1 0 .5 .3}`.

colors.opm

```
218 \_def\_rgbcolordef {\_commoncolordef \_setrgbcolor krg}
219 \_def\_cmkykcolordef {\_commoncolordef \_setcmkykcolor KRG}
220 \_def\_commoncolordef#1#2#3#4#5#6{%
221   \_begingroup
222     \_directlua{color_C=0 color_M=0 color_Y=0 color_K=0}%
223     \_def\_setcmkykcolor##1{!=#2 ##1 }%
224     \_def\_setrgbcolor ##1{!=#3 ##1 }%
225     \_def\_setgreycolor##1{!=#4 ##1 }%
226     \_let\_useK=\_relax
227     \_edef\_tmpb{+##6}%
228     \_replstring\_tmpb{+ }{+}\_replstring\_tmpb{- }{-}%
229     \_replstring\_tmpb{+}\_addcolor\_replstring\_tmpb{-}\_addcolor-}%
230     \_replstring\_tmpb{^!=}{!^}\_replstring\_tmpb{-!}{-1!}%
231     \_ifx K#2\_let\_useK=\_douseK \_fi
232     \_tmpb
233     \_edef\_tmpb{\_colordefFin}%
234     \_ifx k#2\_edef\_tmpb{\_ea\_stripK \_tmpb;}\_fi
235   \_ea\_endgroup
236   \_ea\_def\_ea#5\_ea{\_ea#1\_ea{\_tmpb}}%
237 }
238 \_def\_addcolor#1!#2#3{\_cs{addcolor:#2#3}#1}
239 \_def\_addcolorA #1 #2 #3 #4 #5 {%
240   \_def\_tmpa{#1}\_ifx\_tmpa\_empty \_else \_edef\_tmpa{\_tmpa*}\_fi
241   \_directlua{color_C=math.max(color_C+\_tmpa#2,0)
242               color_M=math.max(color_M+\_tmpa#3,0)
243               color_Y=math.max(color_Y+\_tmpa#4,0)
244               color_K=math.max(color_K+\_tmpa#5,0)
245 }%
246 \_sdef{addcolor:=K}#1 #2 #3 #4 #5 {\_addcolorA #1 #2 #3 #4 #5 }
247 \_sdef{addcolor:=K}#1 #2 #3 #4 #5 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) #5 }
248 \_sdef{addcolor:=G}#1 #2 {\_addcolorA #1 0 0 0 #2 }
249 \_sdef{addcolor:=G}#1 #2 {\_addcolorA #1 0 0 0 (1-#2) }
250 \_sdef{addcolor:=R}#1 #2 #3 #4 {%
251   \_edef\_tmpa{\_noexpand\_addcolorA #1 \_rgbtocmyk #2 #3 #4 ; }\_tmpa
252 }%
253 \_sdef{addcolor:=R}#1 #2 #3 #4 {\_cs{addcolor:=R}#1 (1-#2) (1-#3) (1-#4) }
254
255 \_sdef{addcolor:=k}#1 #2 #3 #4 #5 {%
256   \_edef\_tmpa{\_noexpand\_addcolorA #1 \_cmkyktorgb #2 #3 #4 #5 ; 0 }\_tmpa
257 }%
258 \_sdef{addcolor:=k}#1 #2 #3 #4 #5 {\_cs{addcolor:=k}#1 (1-#2) (1-#3) (1-#4) #5 }
259 \_sdef{addcolor:=g}#1 #2 {\_addcolorA #1 (1-#2) (1-#2) (1-#2) 0 }
260 \_sdef{addcolor:=g}#1 #2 {\_addcolorA #1 #2 #2 #2 0 }
```

```

261 \_sdef{addcolor:=r}#1 #2 #3 #4 {\_addcolorA #1 #2 #3 #4 0 }
262 \_sdef{addcolor:=r}#1 #2 #3 #4 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) 0 }
263 \_def\_stripK#1 .0;{#1}
264 \_let\_colordef=\_cmkycolordef % default \_colordef is \_cmkycolordef

```

Public versions of `\colordef` and `\useK` macros are declared using `\_def`, because the internal versions `\_colordef` and `\_useK` are changed during processing.

```

272 \_def \useK{\_useK}
273 \_def \colordef {\_colordef}
274 \_public \cmkycolordef \rgbcolordef ;

```

colors.opm

The  $\text{\LaTeX}$  file `x11nam.def` is read by `\morecolors`. The numbers 0,1,2,3,4 are transformed to letters O,  $\langle none \rangle$ , B, C, D in the name of the color. Colors defined already are not re-defined. The empty `\_showcolor` macro should be re-defined for color catalog printing. For example:

```

\def\vr{\vrule height10pt depth2pt width20pt}
\def\_showcolor{\hbox{\tt\_bslash\_tmpb: \csname\_tmpb\endcsname \vr}\space\space}
\beginmulti 4 \typsize[11/14]
\morecolors
\endmulti

```

colors.opm

```

290 \_def\_morecolors{%
291   \_long\_def\_tmp##1\preparecolorset##2##3##4##5{\_tmpa ##5;,,,}
292   \_def\_tmpa##1,##2,##3,##4;{\_ifx,##1,\_else
293     \_def\_tmpb{##1}\_replstring\_tmpb{1}{}\\_replstring\_tmpb{2}{B}%
294     \_replstring\_tmpb{3}{C}\\_replstring\_tmpb{4}{D}\\_replstring\_tmpb{0}{0}%
295     \_ifcsname \_tmpb\endcsname \_else
296       \_sdef{\_tmpb}{\_setrgbcolor{##2 ##3 ##4}}\_showcolor\_fi
297     \_ea\_tmpa\_fi
298   }
299   \_ea\_tmp\_input x11nam.def
300 }
301 \_let\_showcolor=\_relax % re-define it if you want to print a color catalog
302 \_public \morecolors ;

```

## 2.20 The .ref file

The `.ref` file has the name `\jobname.ref` and it saves information about references, TOC lines, etc. All data needed in next  $\text{\TeX}$  run are saved here.  $\text{OpTeX}$  reads this file at the beginning of the document (using `\everyjob`) if such file exists. The `.ref` file looks like:

```

\Xrefversion{\ref-version}
\_Xpage{\gpageno}{\pageno}
\_Xtoc{\level}{\type}{\text}{\title}
\_Xlabel{\label}{\text}
\_Xlabel{\label}{\text}
...
\_Xpage{\gpageno}{\pageno}
\_Xlabel{\label}{\text}
...

```

where  $\langle gpageno \rangle$  is internal page number globally numbered from one and  $\langle pageno \rangle$  is a page number (`\the\pageno`) used in pagination (they may be differ). Each page begins with `\_Xpage`. The  $\langle label \rangle$  is a label used by user in `\label[\label]` and  $\langle text \rangle$  is a text which should be referenced (the number of section or table, for example 2.3.14). The  $\langle title \rangle$  is a title of the chapter ( $\langle level \rangle=1$ ,  $\langle type \rangle=\text{chap}$ ), section ( $\langle level \rangle=2$ ,  $\langle type \rangle=\text{sec}$ ), subsection ( $\langle level \rangle=3$ ,  $\langle type \rangle=\text{secc}$ ). The `\_Xpage` is written at beginning of each page, the `\_Xtoc` is written when chapter or section or subsection title exists on the page and `\_Xlabel` when labeled object prefixed by `\label[\label]` exists on the page.

The `.ref` file is read when the processing of the document starts using `\everyjob`. It is read, removed and opened to writing immediately. But the `.ref` file should be missing. If none forward references are needed in the document then `.ref` file is not created. For example, you only want to test a simple plain

TeX macro, you create `test.tex` file, you do `optex test` and you don't need to see empty `test.ref` file in your directory.

```
3 \_codedecl \openref {File for references <2020-02-14>} % preloaded in format
```

ref-file.opm

The `\_inputref` macro is used in `\everyjob`. It reads `\jobname.ref` file if it exists. After the file is read then it is removed and opened to write a new contents to this file.

```
11 \_newwrite\_reffile
12
13 \_def\_inputref {%
14   \_isfile{\_jobname.ref}\_iftrue
15     \_input {\_jobname.ref}
16     \_gfnotenum=0 \_lfnotenum=0 \_mnotenum=0
17     \_openrefA{\_string\_inputref}%
18   \_fi
19 }
```

ref-file.opm

If the file does not exists then it is not created by default. It means that if you process a document without any forward references then no `\jobname.ref` file is created because it is unusable. The `\_wref` macro is dummy in such case.

```
28 \_def\_wrefrelax#1#2{}
29 \_let\_wref=\_wrefrelax
```

ref-file.opm

If a macro needs to create and to use `.ref` file then such macro must use `\openref`. When the file is created (using internal `\_openrefA`) then the `\_wref \<macro>\<data>` is redefined in order to save the line `\<macro>\<data>` to the `.ref` file using asynchronous `\write` primitive. Finally, the `\_openref` destroys itself, because we need not to open the file again.

```
40 \_def\_openref {%
41   \_ifx \_wref\_wrefrelax \_openrefA{\_string\_openref}\_fi
42   \_gdef\_openref{}%
43 }
44 \_def\_openrefA #1{%
45   \_immediate\_openout\_reffile="\_jobname.ref"\_relax
46   \_gdef\_wref ##1##2{\_write\_reffile{\_string##1##2}}%
47   \_immediate\_write\_reffile {\_pcent\_pcent\_space OPTeX <\_optexversion> - REF file (#1)}%
48   \_immediate\_wref \Xrefversion{\_REFversion}%
49 }
50 \def\openref{\_openref}
```

ref-file.opm

We are using convention that the macros used in `.ref` file are named `\_X<foo>`. If there is a new version of OpTeX with different collection of such macros then we don't want to read the `.ref` files produced by an old version of OpTeX or by OPmac. So first line of `.ref` file is in the form

```
\Xrefversion{\_version}
```

We can check the version compatibility by this macro. Because OPmac does not understand `\_Xrefversion` we use `\Xrefversion` (with different number of `\_version` form OPmac) here. The result: OPmac skips the `.ref` files produced by OpTeX and vice versa.

```
68 \_def\_REFversion{3} % actual version of .ref files in OpTeX
69 \_def\_Xrefversion#1{\_ifnum #1=\_REFversion\_relax \_else \_endinput \_fi}
70 \_public \Xrefversion ; % we want to ignore .ref files generated by OPmac
```

ref-file.opm

You cannot define your special `.ref` macros before `.ref` file is read because it is read in `\everyjob`. But you can define such macros using `\refdecl{\<definitions of your ref macros>}`. This command sends to `.ref` file your `\<definitions of your ref macros>` immediately. Next lines in `.ref` file should include our macros. Example from CTUstyle2:

```
\refdecl{%
  \def\totlist{} \def\toflist{}^^J
  \def\Xtab#1#2#3{\addto\totlist{\totline{#1}{#2}{#3}}}^^J
  \def\Xfig#1#2#3{\addto\toflist{\tofline{#1}{#2}{#3}}}
}
```

We must read *<definition of your ref macros>* when catcode of # is 12 because we needn't to duplicate each # in the .ref file.

```
ref-file.opm
```

```

90 \_def\_refdecl{\_bgroup \_catcode\#=12 \_refdeclA}
91 \_def\_refdeclA #1{\_egroup\_openref
92   \_immediate\_write\_reffile {\\_pcent\_space \_string \_refdecl:}%
93   \_immediate\_write\_reffile {\\_detokenize{#1}}%
94 }
95 \_public \_refdecl ;

```

## 2.21 References

If the references are “forward” (i. e. the \ref is used first, the destination is created later) or if the reference text is page number then we must read .ref file first in order to get appropriate information. See section 2.20 for more information about .ref file concept.

```
references.opm
```

```

3 \_codedecl \_ref {References <2020-03-03>} % preloaded in format

```

\\_Xpage {\<pgageno>}{\<pageno>} saves the parameter pair into \\_currpage. Resets \\_lfnotenun; it is used if footnotes are numbered from one at each page.

```
references.opm
```

```

10 \_def\_Xpage#1#2{\_def\_currpage{{#1}{#2}}\_lfnotenun=0 }

```

\\_Xlabel {\<label>}{\<text>} saves the \<text> to \\_lab:\<label> and saves [pg:\<pgageno>]{\<pageno>} to \\_pgref:\<label>.

```
references.opm
```

```

17 \_def\_Xlabel#1#2{\_sdef{\_lab:#1}{#2}\_sxdef{\_pgref:#1}{\_ea\_bracketspg\_currpage}}
18 \_def\_bracketspg#1#2{[pg:#1]{#2}}

```

\\_label[\<label>] saves the decalred label to \\_lastlabel and \\_wlabel{\<text>} uses the \\_lastlabel and activates \\_wref\\_Xlabel{\<label>}{\<text>}.

```
references.opm
```

```

26 \_def\_label[#1]{\_isempty{#1}\_iftrue \_global\_let \_lastlabel=\_undefined
27   \_else \_isdefined{10:#1}%
28   \_iftrue \_opwarning{duplicated label [#1], ignored}\_else \_xdef\_lastlabel{#1}\_fi
29   \_fi \_ignorespaces
30 }
31 \_def\_wlabel#1{%
32   \_ifx\_lastlabel\_undefined \_else
33     \_dest[ref:\_lastlabel]%
34     \_printlabel\_lastlabel
35     \_edef\_tmp{{\_lastlabel}{#1}}%
36     \_ea\_wref \_ea\_Xlabel \_ea{\_tmp}%
37     \_sxdef{\_lab:\_lastlabel}{#1}\_sxdef{10:\_lastlabel}{}%
38     \_global\_let\_lastlabel=\_undefined
39   \_fi
40 }
41 \_public \_label \_wlabel ;

```

\\_ref[\<label>] uses saved \\_lab:\<label> and prints (linked) \<text>. If the reference is backwarded then we know \\_lab:\<label> without any need to read REF file. On the other hand, if the reference is forwarded, then we doesn't know \\_lab:\<label> in first run of T<sub>E</sub>X and we print warning and do \\_openref.

\\_pgref[\<label>] uses {\<pgageno>}{\<pageno>} from \\_pgref:\<label> and prints (linked) \<pgageno> using \\_ilink macro.

```
references.opm
```

```

54 \_def\_ref[#1]{\_isdefined{\_lab:#1}%
55   \_iftrue \_ilink[ref:#1]{\_csname \_lab:#1\_endcsname}%
56   \_else ??\_opwarning[label [#1] unknown. Try to TeX me again]\_openref
57   \_fi
58 }
59 \_def\_pgref[#1]{\_isdefined{\_pgref:#1}%
60   \_iftrue \_ea\_ea\_ea\_ilink \_csname \_pgref:#1\_endcsname
61   \_else ??\_opwarning[pg-label [#1] unknown. Try to TeX me again]\_openref
62   \_fi
63 }
64 \_public \_ref \_pgref ;

```



Default `\_printlabel` is empty macro (labels are not printed). The `\showlabels` redefines it as box with zero dimensions and with left lapped `[<label>]` in blue 10pt `\tt` font shifted up by 1.7ex.

references.opm

```
72 \_def\_printlabel#1{
73 \_def\_showlabels {%
74 \_def\_printlabel##1{\_vbox to0pt{\_vss\_llap{\_labelfont{##1}}\_kern1.7ex}}%
75 \_fontdef\_labelfont{\_setfontsize{at10pt}\setfontcolor{blue}\_tt}
76 }
77 \_public \showlabels ;
```

## 2.22 Hyperlinks

There are four types of the internal links and one type of external link:

- `ref:<label>` – the destination is created when `\label[<label>]` is used, see also the section 2.21.
- `toc:<tocrefnum>` – the destination is created at chap/sec/secc titles, see also the section 2.23.
- `pg:<gpageno>` – the destination is created at beginning of each page, see also the section 2.17.
- `cite:<bibnum>` – the destination is created in bibliography reference, see also the section 2.31.1.
- `url:<url>` – used by `\url` or `\link`, see also the end of this section.

The `<tocrefnum>`, `<gpageno>` and `<bibnum>` are numbers starting from one and globally incremented by one in whole document. The registers `\tocrefnum`, `\gpageno` and `\bibnum` are used for these numbers.

When a chap/sec/secc title is prefixed by `\label[<label>]`, then both types of internal links are created at the same destination place: `toc:<tocrefnum>` and `ref:<label>`.

hyperlinks.opm

```
3 \_codedecl \link {Hyperlinks <2020-04-22>} % preloaded in format
```

`\dest[<type>:<spec>]` creates a destination of internal links. The destination is declared in the format `<type>:<spec>`. If the `\hyperlinks` command is not used, then `\dest` does nothing else it is set to `\_destactive`. The `\_destactive` is implemented by `\_pdfdest` primitive. It creates a box in which the destination is shifted by `\_destheight`. The reason is that the destination is exactly at top border of the PDF viewer but we want to see the line where destination is. The destination box is positioned by different way dependent on current vertical or horizontal mode.

hyperlinks.opm

```
16 \_def\_destheight{1.4em}
17 \_def\_destactive[#1:#2]{\_if$#2$\_else\_ifvmode
18 \_tmpdim=\_prevdepth \_prevdepth=-1000pt
19 \_destbox[#1:#2]\_prevdepth=\_tmpdim
20 \_else \_destbox[#1:#2]%
21 \_fi\_fi
22 }
23 \_def\_destbox[#1]{\_vbox to0pt{\_kern-\_destheight \_pdfdest name{#1} xyz\_vss}}
24 \_def\_dest[#1]{
25 \_public \dest ;
```

`\link[<type>:<spec>]{<color>}{<text>}` creates an internal link to `\dest` with the same `<type>:<spec>`. You can have more links with the same `<type>:<spec>` but only one `\dest` in the document. If `\hyperlinks` command is not used, then `\link` only prints `<text>` else it is set to `\_linkactive`. The `\_linkactive` is implemented by `\_pdfstartlink..._pdfendlink` primitives.

`\ilink[<type>:<spec>]{<text>}` is equivalent to `\link` but the `<color>` is used from `\hyperlinks` declaration.

hyperlinks.opm

```
40 \_protected\_def\_linkactive[#1:#2]#3#4{\_leavevmode\_pdfstartlink height.9em depth.3em
41 \_pdfborder{#1} goto name{#1:#2}\_relax {#3#4}\_pdfendlink
42 }
43 \_protected\_def\_link[#1]#2#3{\_leavevmode{#3}}
44 \_protected\_def\_ilink[#1]#2{\_leavevmode{#2}}
45 \_public \ilink \link ;
```

`\urlink[<url>]{<text>}` creates external link. It prints only the `<text>` by default but the `\hyperlinks` declaration defines it as `\_urlactive[url:<url>]{<text>}`. The external link is created by the `\_pdfstartlink..._pdfendlink` primitives. The `<url>` is detokenized with `\escapechar=-1` before it is used, so `\%`, `\#` etc. can be used in the `<url>`.

hyperlinks.opm

```
55 \_protected\_def\_urlactive[#1:#2]#3#4{\_leavevmode{\_escapechar=-1
```

```

56 \pdfstartlink height.9em depth.3em \pdfborder{#1}%
57   user{/Subtype/Link/A <</Type/Action/S/URI/URI(\_detokenize{#2})>>}\_relax
58   {#3#4}\_pdfendlink}%
59 }
60 \_def\_ulink[#1]#2{\_leavevmode{#2}}
61 \_def\_urlcolor{}
62 \_public \ulink ;

```

The `\pdfstartlink` primitive uses `\pdfborder{<type>}` in its parameter (see `\linkactive` or `\urlactive` macros). The `\pdfborder{<type>}` expands to `attr{/C[? ? ?] /Border[0 0 .6]}` if the `\<type>border` (i.e. `\refborder`, `\citeborder`, `\tocborder`, `\pgborder`, `\urlborder`, `\fntborder` or `\fnfborder`) is defined. User can define it in order to create colored frames around active links. For example `\def\tocborder{1 0 0}` causes red frames in TOC (not printed, only visible in PDF viewers).

hyperlinks.opm

```

76 \_def\_pdfborder#1{\_ifcstype #1border\_endcstype
77   attr{/C[\_cstype #1border\_endcstype] /Border[0 0 .6]}%
78   \_else attr{/Border[0 0 0]}\_fi
79 }

```

`\hyperlinks{<link_color>}{<ulink_color>}` activates `\dest`, `\link`, `\ilink`, `\ulink` in order they create links. These macros are redefined here to their “active” version.

hyperlinks.opm

```

87 \_def\_hyperlinks#1#2{%
88   \_let\_dest=\_destactive \_let\_link=\_linkactive
89   \_def\_ilink[#1]##2{\_link[#1]{\_localcolor#1}{##2}}%
90   \_def\_ulink[#1]##2{\_urlactive[url:##1]{\_localcolor#2}{##2}}%
91   \_public \dest \ilink \ulink ;%
92 }
93 \_public \hyperlinks ;

```

`\url{<url>}` does approximately the same as `\ulink[<url>]{<url>}`, but more work is done before the `\ulink` is processed. The link-version of `<url>` is saved to `\_tmpa` and the printed version in `\_tmpb`. The printed version is modified in order to set breakpoints to special places of the `<url>`. For example `//` is replaced by `\_urlskip/\_urlskip/\_urlbskip` where `\urlskip` adds a small nobreakable glue between these two slashes and before them and `\urlbskip` adds a breakable glue after them.

The text version of the `<url>` is printed in `\_urlfont`.

hyperlinks.opm

```

107 \_def\_url#1{%
108   \_def\_tmpa{#1}\_replstring\_tmpa {\|}{}%
109   {\_escapechar=-1 \_ea\_ea\_edef\_ea\_tmpa\_ea{\_detokenize\_ea{\_tmpa}}}%
110   \_def\_tmpb{#1}\_replstring\_tmpb {\|}{\_urlbskip}%
111   \_replstring\_tmpb {//} {{\_urlskip\_urlslashslash\_urlbskip}}%
112   \_replstring\_tmpb {/} {{\_urlskip/\_urlbskip}}%
113   \_replstring\_tmpb {.} {{\_urlskip.\_urlbskip}}%
114   \_replstring\_tmpb {?} {{\_urlskip?\_urlbskip}}%
115   \_replstring\_tmpb {=} {{\_urlskip=\_urlbskip}}%
116   \_ea\_replstring\_ea\_tmpb \_ea{\_string &} {{\_urlbskip\_char`& \_urlskip}}%
117   \_ea\_replstring\_ea\_tmpb \_ea{\_bslash|} {{\_penalty0}}%
118   \_ea\_ulink \_ea[\_tmpa] {\_urlfont\_tmpb\_null}%
119 }}
120 \_def\_urlfont{\_tt}
121 \_def\_urlskip{\_null\_nobreak\_hskip0pt plus0.05em\_relax}
122 \_def\_urlbskip{\_penalty100 \_hskip0pt plus0.05em\_relax}
123 \_def\_urlslashslash{/\_urlskip/}
124
125 \_public \url ;

```

## 2.23 Making table of contents

maketoc.opm

```

3 \_codedecl \maketoc {Macros for maketoc <2020-03-12>} % preloaded in format

```

`\_Xtoc {<level>}{<type>}{<number>}{<title>}` (in `.ref` file) reads the specified data and appends them to the `\_toclist` as `\_tocline{<level>}{<type>}{<number>}{<title>}{<gpageno>}{<pageno>}` where:

- $\langle level \rangle$ : 0 reserved, 1: chapter, 2: section, 3: subsection
- $\langle type \rangle$ : the type of the level, i.e. chap, sec, secc
- $\langle number \rangle$ : the number of the chapter/section/subsection in the format 1.2.3
- $\langle title \rangle$ : the title text
- $\langle gpageno \rangle$ : the page number numbered from 1 independently of pagination
- $\langle pageno \rangle$ : the page number used in the pagination

The last two parameters are restored from previous  $\backslash\_Xpage\{\langle pageno \rangle\}\{\langle gpageno \rangle\}$ , data were saved in the  $\backslash\_currrpage$  macro.

```

22 \_def\__toclist{
23 \_newifi \_ifischap \_ischapfalse
24
25 \_def\__Xtoc#1#2#3#4{\_ifnum#1=0 \_ischaptrue\_fi
26 \_addto\__toclist{\_tocline{#1}{#2}{#3}{#4}}
27 \_ea\_addto\_ea\__toclist\_ea{\_currrpage}%
28 }
```

maketoc.opm

$\backslash\_tocline\{\langle level \rangle\}\{\langle type \rangle\}\{\langle number \rangle\}\{\langle title \rangle\}\{\langle gpageno \rangle\}\{\langle pageno \rangle\}$  prints the record to the table of contents. It opens group, reduces  $\backslash\_leftskip$ ,  $\backslash\_rightskip$ , runs the  $\backslash\_everytocline$  (user can customise the design of TOC here) and runs  $\backslash\_tocl:\langle level \rangle \ \{\langle number \rangle\}\{\langle title \rangle\}\{\langle pageno \rangle\}$  macro. This macro starts with vertical mode, inserts one record with given  $\langle level \rangle$  and it should end by  $\backslash\_tocpar$  which returns to horizontal mode. The  $\backslash\_tocpar$  appends  $\backslash\_nobreak \ \backslash\_hskip-2\backslash\_iindent\backslash\_null \ \backslash\_par$ . This causes that the last line of the record is shifted outside the margin given by  $\backslash\_rightskip$ . A typical record (with long  $\langle title \rangle$ ) looks like:

```

      |
\llap{<number>} text text text text text
      text text text text text
      text text ..... <pageno>
```

Margins given by  $\backslash\_leftskip$  and  $\backslash\_rightskip$  are denoted by | in the example above.

$\backslash\_tocrefnum$  is global counter of all TOC records (used by hyperlinks).

```

53 \_newcount \_tocrefnum
54 \_def\__tocline#1#2#3#4#5#6{
55 \_advance\_tocrefnum by1
56 \_bgroup
57 \_leftskip=\_iindent \_rightskip=2\_iindent
58 \_ifischap \_advance\_leftskip by \_iindent \_fi
59 \_def\__pgn{\_ilink[pg:#5]}%
60 \_the\_everytocline
61 \_ifcsname \_tocl:#1\_endcsname
62 \_cs{\_tocl:#1}{#3}{#4}{#6}\_par
63 \_fi
64 \_egroup
65 }
66 \_public \_tocrefnum ;
```

maketoc.opm

You can re-define default macros for each level of tocline if you want.

Parameters are  $\{\langle number \rangle\}\{\langle title \rangle\}\{\langle pageno \rangle\}$ .

```

73 \_sdef{\_tocl:1}#1#2#3{\_nofirst\_bigskip \_bf\_llaptoclink{#1}{#2}\_hfill \_pgn{#3}\_tocpar}
74 \_sdef{\_tocl:2}#1#2#3{\_llaptoclink{#1}{#2}\_tocdotfill \_pgn{#3}\_tocpar}
75 \_sdef{\_tocl:3}#1#2#3{\_advance\_leftskip by\_iindent \_cs{\_tocl:2}{#1}{#2}{#3}}
```

maketoc.opm

The auxiliary macros are:

- $\backslash\_llaptoclink\langle text \rangle$  does  $\backslash\_noindent\backslash\_llap\{\langle linked text \rangle\}$ .
- $\backslash\_tocdotfill$  creates dots in the TOC.
- $\backslash\_nofirst$  macro applies the macro only if we don't print the first record of the TOC.
- $\backslash\_tocpar$  finalizes one TOC record which is lapped  $\langle pageno \rangle$ .
- $\backslash\_pgn\{\langle pageno \rangle\}$  creates  $\langle pageno \rangle$  as link to real  $\langle gpage \rangle$  saved in #6 of  $\backslash\_tocline$ . This is temporarily defined in the  $\backslash\_tocline$ .

```
90 \def\llaptoclink#1{\noindent
91   \llap{\_ilink[toc:\_the\_tocrefnum]{\_enspace#1\_kern.4em}\_kern.1em}}
92 \def\__tocdotfill{\_nobreak\_leaders\_hbox to.8em{\_hss.\_hss}\_hskip 1em plusifill\_relax}
93 \def\__nofirst #1{\_ifnum \_lastpenalty=11333 \_else #1\_fi}
94 \def\__tocpar{\_nobreak \_hskip-2\_iindent\_null \_par}
```

`\maketoc` prints warning if TOC data is empty, else it creates TOC by running `\_toclist`

```
101 \def\__maketoc{\_par \_ifx\_toclist\_empty
102   \_opwarning{\_noexpand\maketoc -- data unavailable, TeX me again}\_openref
103   \_else \_begingroup
104     \_tocrefnum=0 \_penalty11333
105     \_the\_regtoc \_toclist
106   \_endgroup \_fi
107 }
```

`\regmacro` appends its parameters to `\_regtoc`, `\_regmark` and `\_regoul`. These token lists are used in `\maketoc`, `\_begoutput` and `\pdfunidef`.

```
115 \_newtoks \_regtoc \_newtoks \_regmark \_newtoks \_regoul
116
117 \def\__regmacro #1#2#3{%
118   \_toksapp\_regtoc{#1}\_toksapp\_regmark{#2}\_toksapp\_regoul{#3}%
119 }
120 \_public \maketoc \regmacro ;
```

## 2.24 PDF outlines

### 2.24.1 Nesting PDF outlines

The problem is that PDF format needs to know the number of direct descendants of each outline if we need to create the tree of structured outlines. But we know only the level of each outline. The required data should be calculated from TOC data. We use two steps over TOC data saved in the `\_toclist` where each record is represented by one `\_tocline`.

First step, the `\outlines` macro sets `\_tocline` to `\_outlinesA` and calculates the number of direct descendants of each record. Second step, the `\outlines` macro sets `\_tocline` to `\_outlinesB` and it uses prepared data and create outlines.

Each outline is mapped to the control sequence of the type `\_ol:<num>` or `\_ol:<num>:<num>` or `\_ol:<num>:<num>:<num>` or etc. The first one is reserved for level 0, the second one for level 1 (chapters), third one for level 2 (sections) etc. The number of direct descendants will be stored in these macros after first step is finished. Each new outline of given level increases the `<num>` at given level. When the first step is processed then (above that) the `\_ol:...` sequence of the parent increases its value too. The `\_ol:...` sequences are implemented by `\_ol:\_count0:\_count1:\_count2` etc. For example, when section (level 2) is processed in the first step then we do:

```
\advance \count2 by 1
      % increases the mapping pointer of the type
      % \_ol:<\_count0:\_count1:\_count2> of this section
\advance \<\_ol:\_count0:\_count1> by 1
      % increases the number of descendants connected
      % to the parent of this section.
```

When second step is processed, then we only read the stored data about the number of descendants. And we use it in `count` parameter of `\_pdfoutline` primitive.

For linking, we use the same links as in TOC, i.e. the `toc:\_the\_tocrefnum` labels are used.

`\insertoutline {<text>}` inserts one outline with zero direct descendants. It creates link destination of the type `oul:<num>` into the document (where `\insertoutline` is used) and the link itself is created too in the outline.

```
3 \_codedecl \outlines {PDF outlines <2020-03-12>} % preloaded in format
4
5 \def\__outlines#1{\_pdfcatalog{/PageMode/UseOutlines}\_openref
6   \_ifx\_toclist\_empty
7   \_opwarning{\_noexpand\outlines -- data unavailable. TeX me again}%
```

```

8 \_else
9 \_ifx\_dest\_destactive \_else
10 \_opwarning{\_noexpand\outlines doesn't work when \_noexpand\hyperlinks isn't declared}\_fi
11 {\_let\_tocline=\_outlinesA
12 \_count0=0 \_count1=0 \_count2=0 \_count3=0 \_toclist % calculate numbers o childs
13 \_def\_outlinelevel{#1}\_let\_tocline=\_outlinesB
14 \_tocrefnum=0 \_count0=0 \_count1=0 \_count2=0 \_count3=0
15 \_toclist}% create outlines
16 \_fi
17 }
18 \_def\_outlinesA#1#2#3#4#5#6{%
19 \_advance\_count#1 by1
20 \_ifcase#1\_or
21 \_addoneol{ol:\_the\_count0}\_or
22 \_addoneol{ol:\_the\_count0:\_the\_count1}\_or
23 \_addoneol{ol:\_the\_count0:\_the\_count1:\_the\_count2}\_or
24 \_addoneol{ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}\_fi
25 }
26 \_def\_addoneol#1{%
27 \_ifcsname #1\_endcsname
28 \_tmpnum=\_csname#1\_endcsname\_relax
29 \_advance\_tmpnum by1 \_sxdef{#1}{\_the\_tmpnum}%
30 \_else \_sxdef{#1}{1}%
31 \_fi
32 }
33 \_def\_outlinesB#1#2#3#4#5#6{%
34 \_advance\_count#1 by1
35 \_ifcase#1%
36 \_tmpnum=\_trycs{ol:\_the\_count0}{0}\_or
37 \_tmpnum=\_trycs{ol:\_the\_count0:\_the\_count1}{0}\_relax\_or
38 \_tmpnum=\_trycs{ol:\_the\_count0:\_the\_count1:\_the\_count2}{0}\_relax\_or
39 \_tmpnum=\_trycs{ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}{0}\_relax\_or
40 \_tmpnum = 0\_relax\_fi
41 \_pdfunidef\_tmp{#4}%
42 \_advance\_tocrefnum by1
43 \_outlinesC{#1}{toc:\_the\_tocrefnum}{\_ifnum#1<\_outlinelevel\_space\_else-\_fi}{\_tmpnum}{\_tmp}%
44 }
45 \_def\_outlinesC#1#2#3#4#5{\_pdfoutline goto name{#2} count #3#4{#5}\_relax}
46
47 \_newcount\_oulnum
48 \_def\_insertoutline#1{\_global\_advance\_oulnum by1
49 \_pdfdest name{oul:\_the\_oulnum} xyz\_relax
50 \_pdfunidef\_tmp{#1}%
51 \_pdfoutline goto name{oul:\_the\_oulnum} count0 {\_tmp}\_relax
52 }
53 \_public \outlines \insertoutline ;

```

## 2.24.2 Strings in PDF outlines

There are only two encodings for PDF strings (used in PDFoutlines, PDFinfo etc.). First one is PDFDocEncoding which is one-byte encoding, but most Czech or Slovak characters are missing here.

The second encoding is PDFUnicode encoding which is implemented in this file. This encoding is T<sub>E</sub>X-discomfortable, because it looks like

```

\376\377\000C\000v\000i\001\015\000e\000n\000\355\000\040\000j\000e\000\040
\000z\000\341\000t\001\033\001\176

```

This example is real encoding of the string "Cvičení je zátěž". You can see that this is UTF-16 encoding (two bytes per character) with two starting bytes FEFF. Moreover, each byte is encoded by three octal digits preceded by backslash. The only exception is the visible ASCII character encoding: such a character is encoded by its real byte preceded by \000.

The command `\pdfunidef\macro{string}` is implemented here using `\directlua`. The input string is preprocessed: detokenized, converted `\word /` to `\word/` (used in logos) preprocessed spaces using `\replstring` and then the `\pdfunidefB` is repeated on each character. It calls the `\directlua` chunk to print octal numbers in the macro `\_octalprint`.

The `\regmacro` is used in order to set the values of macros `\em`, `\rm`, `\bf`, `\it`, `\bi`, `\tt`, `\/` and `~` to values usable in PDF outlines.

```

3 \_codedecl \pdfunidef {PDFUnicode strings for outlines <2020-03-12>} % preloaded in format
4
5 \_edef\__octalprint#1#2{\_noexpand\_directlua{% #1=character-code #2=character
6   if ('#2'>='A' and '#2'<='Z') or ('#2'>='a' and '#2'<='z') then
7     tex.print(string.format('000\_pcent s',"#2"))
8   else
9     local num=#1\_pcent256
10    tex.print(string.format('\_pcent 03o\_nbb\_pcent03o',(#1-num)/256,num))
11  end
12 }}
13 \_def\__pdfunidef#1#2{%
14   \_begingroup
15     \_catcode\__=12 \_let\__=\_bslash
16     \_the\_regoul \_relax % \_regmacro alternatives of logos etc.
17     \_escapechar=-1
18     \_edef#1{\_empty}%
19     \_escapechar=\__
20     \_ea\_edef \_ea#1\_ea{\_ea\_removeoutmath #1$\_end$}% $x$ -> x
21     \_ea\_edef \_ea#1\_ea{\_ea\_removeoutbraces #1{\_end}}% {x} -> x
22     \_edef#1{\_detokenize\_ea{#1}}%
23     \_replstring#1{ }{ }% text text -> text{ }text
24     \_edef\__out{\_376\377}%
25     \_ea\_pdfunidefB#1^% text -> \_out in octal
26     \_ea
27   \_endgroup
28   \_ea\_def\__ea#1\_ea{\_out}
29 }
30 \_def\__pdfunidefB#1{%
31   \_ifx^#1\_else
32     \_tmpnum=#1
33     \_pdfunidefC{\_luaescapestring{#1}}%
34     \_ea\_pdfunidefB \_fi
35 }
36 \_def\__pdfunidefC #1{\_edef\__out{\_out \_\_ea\_octalprint\_ea{\_the\_tmpnum}{#1}}}
37
38 \_def\__removeoutbraces #1#{#1\_removeoutbracesA}
39 \_def\__removeoutbracesA #1{\_ifx\_end#1\_else #1\_ea\_removeoutbraces\_fi}
40 \_def\__removeoutmath #1$#2$#{#1\_ifx\_end#2\_else #2\_ea\_removeoutmath\_fi}
41
42 \_regmacro {}{ }\_let\em=\_empty \_let\rm=\_empty \_let\bf=\_empty
43   \_let\it=\_empty \_let\bi=\_empty \_let\tt=\_empty \_let\=/=\_empty
44   \_let-=\_space
45 }
46 \public \pdfunidef ;

```

## 2.25 Chapters, sections, subsections

```

3 \_codedecl \chap {Titles, chapters, sections, subsections <2020-03-28>} % preloaded in format

```

We are using scaled fonts for titles `\_titfont`, `\_chapfont`, `\_secfont` and `\_seccfont`. They are scaled from main fonts size of the document, which is declared by first `\_typosize[⟨fo-size⟩/⟨b-size⟩]` command.

```

13 \_def \_titfont {\_scalemain\_typoscale[\_magstep4/\_magstep5]\_boldify}
14 \_def \_chapfont {\_scalemain\_typoscale[\_magstep3/\_magstep3]\_boldify}
15 \_def \_secfont {\_scalemain\_typoscale[\_magstep2/\_magstep2]\_boldify}
16 \_def \_seccfont {\_scalemain\_typoscale[\_magstep1/\_magstep1]\_boldify}

```

The `\_tit` macro is defined by `\_eoldef`, it means that the parameter is separated by end of line. The macros `\_chap`, `\_sec` and `\_secc` use `\_eoldef` too.

```

24 \_eoldef\_tit #1{\_vglue\_titskip
25   {\_leftskip=0pt plus1fill \_rightskip=\_leftskip % centering
26     \_titfont \_noindent #1\_par}%
27   \_nbreak\_bigskip
28 }
29 \_public \_tit ;

```



You can re-define `\_printchap`, `\_printsec` or `\_printsecc` macros if another design of section titles is needed. These macros gets the `\langle title \rangle` text in its parameter. The common recommendations for these macros are:

- Use `\_abovetitle{\langle penaltyA \rangle}{\langle skipA \rangle}` and `\_belowtitle{\langle skipB \rangle}` for inserting vertical material above and below the section title. The arguments of these macros are normally used, i.e. `\_abovetitle` inserts `\langle penaltyA \rangle \langle skipA \rangle` and `\_belowtitle` inserts `\langle skipB \rangle`. But there is an exception: if `\_belowtitle{\langle skipB \rangle}` is immediately followed by `\_abovetitle{\langle penaltyA \rangle}{\langle skipA \rangle}` (for example section title is immediately followed by subsection title), then only `\langle skipA \rangle` is generated, i.e. `\langle skipB \rangle \langle penaltyA \rangle \langle skipA \rangle` is reduced only to `\langle skipA \rangle`. The reason of such behavior: we don't want to duplicate vertical skip and we don't want to use negative penalty in such cases. Moreover, `\_abovetitle{\langle penaltyA \rangle}{\langle skipA \rangle}` takes previous whatever vertical skip (other than from `\_belowtitle`) and generates only greater from this pair of skips. It means that `\langle whatever-skip \rangle \langle penaltyA \rangle \langle skipA \rangle` is transformed to `\langle penaltyA \rangle \max(\langle whatever-skip \rangle \langle skipA \rangle)`. The reason of such behavior: we don't want to duplicate vertical skips (from `\_belowlistskip`, for example) above the title.
- Use `\_printrefnum[\langle pre \rangle @ \langle post \rangle]` in horizontal mode. It prints `\langle pre \rangle \langle ref-num \rangle \langle post \rangle`. The `\langle ref-num \rangle` is `\_thechapnum` or `\_theseccnum` or `\_theseccnum` depending on what type of title is processed. If `\_nonum` prefix is used then `\_printrefnum` prints nothing. The macro `\_printrefnum` does more work: it creates destination of hyperlinks (if `\_hyperlinks{\}{}` is used) and saves references from label (if `\_label[\langle label \rangle]` precedes) and saves references for table of contents (if `\_maketoc` is used).
- Use `\_nbp` for closing the paragraph for printing title. This command inserts `\_nobreak` between each line of such paragraph, so the title cannot be broken to more pages.
- You can use `\_firstnoindent` in order to the first paragraph after the title is not indented.

sections.opm

```

69 \_def\_printchap #1{\_vfill\_supereject
70   {\_chapfont \_noindent \_mtext{chap} \_printrefnum[@]\_par
71    \_nobreak\_smallskip
72    \_noindent \_raggedright #1\_nbp}\\_mark{}}%
73   \_nobreak \_belowtitle{\_bigskip}%
74   \_firstnoindent
75 }
76 \_def\_printsec#1{\_par
77   \_abovetitle{\_penalty-400}\_bigskip
78   {\_secfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbp}\\_insertmark{#1}%
79   \_nobreak \_belowtitle{\_medskip}%
80   \_firstnoindent
81 }
82 \_def\_printsecc#1{\_par
83   \_abovetitle{\_penalty-200}\_medskip
84   {\_seccfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbp}%
85   \_nobreak \_belowtitle{\_medskip}%
86   \_firstnoindent
87 }

```

The `\_sectionlevel` is the level of the printed section:

- `\_sectionlevel=0` – reserved for parts of the book (unused by default)
- `\_sectionlevel=1` – chapters (used in `\chap`)
- `\_sectionlevel=2` – sections (used in `\sec`)
- `\_sectionlevel=3` – subsections (used in `\secc`)
- `\_sectionlevel=4` – subsubsections (unused by default)

sections.opm

```

100 \_newcount\_sectionlevel
101 \_def \_secinfo {\_ifcase \_sectionlevel
102   part\_or chap\_or sec\_or secc\_or seccc\_fi
103 }

```

The `\_chapx` initializes counters used in chapters, the `\_secx` initializes counters in sections and `\_seccx` initializes counters in subsections. If you have more types of numbered objects in your document then you can declare appropriate counters and do `\addto\_chapx{\yourcounter=0}` for example. If you have another concept of numbering objects used in your document, you can re-define these macros. All settings here are global because it is used by `\_globaldefs=1 \_chapx`.

Default concept: Tables, figures and display maths are numbered from one in each section – subsections doesn't reset these counters. Footnotes declared by `\fnotenumchapters` are numbered in each chapter from one.

The `\the*` macros `\_thechapnum`, `\_theseccnum`, `\_theseccnum`, `\_thetnum`, `\_thefnum` and `\_thednum` include the format of numbers used when the object is printing. If chapter is never used in the document then `\_chapnum=0` and `\_othe\_chapnum` expands to empty. Sections have numbers  $\langle num \rangle$  and subsections  $\langle num \rangle.\langle num \rangle$ . On the other hand, if chapter is used in the document then `\_chapnum>0` and sections have numbers  $\langle num \rangle.\langle num \rangle$  and subsections have numbers  $\langle num \rangle.\langle num \rangle.\langle num \rangle$ .

sections.opm

```

132 \_newcount \_chapnum % chapters
133 \_newcount \_secnum % sections
134 \_newcount \_seccnum % subsections
135 \_newcount \_tnum % table numbers
136 \_newcount \_fnum % figure numbers
137 \_newcount \_dnum % numbered display maths
138
139 \_def \_chapx {\_secx \_secnum=0 \_lfnnotenum=0 }
140 \_def \_secx {\_seccx \_seccnum=0 \_tnum=0 \_fnum=0 \_dnum=0 \_resetABCDE }
141 \_def \_seccx {}
142
143 \_def \_thechapnum {\_the\_chapnum}
144 \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum}
145 \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum.\_the\_seccnum}
146 \_def \_thetnum {\_othe\_chapnum.\_the\_secnum.\_the\_tnum}
147 \_def \_thefnum {\_othe\_chapnum.\_the\_secnum.\_the\_fnum}
148 \_def \_thednum {\_the\_dnum}
149
150 \_def \_othe #1.{\_ifnum#1>0 \_the#1.\_fi}
151 \_def \_incr #1{\_global\_advance#1by1 }

```

The `\notoc` and `\nonum` prefixes are implemented by internal `\_ifnotoc` and `\_ifnonum`. They are reset after each chapter/section/subsection by the `\_resetnonumnotoc` macro.

sections.opm

```

159 \_newif \_ifnotoc \_notocfalse \_def \_notoc {\_global\_notoctrue}
160 \_newif \_ifnonum \_nonumfalse \_def \_nonum {\_global\_nonumtrue}
161 \_def \_resetnonumnotoc{\_global\_notocfalse \_global\_nonumfalse}
162 \_public \notoc \nonum ;

```

The `\chap`, `\sec` and `\secc` macros are implemented here. The `\_inchap`, `\_insec` and `\_insecc` macros does the real work, First, we read the optional parameter  $[\langle label \rangle]$ , if it exists.

sections.opm

```

170 \_optdef \_chap []{\_trylabel \_inchap}
171 \_optdef \_sec []{\_trylabel \_insec}
172 \_optdef \_secc []{\_trylabel \_insecc}
173 \_def \_trylabel{\_istoksempy\_opt\_iffalse \_label[\_the\_opt]\_fi}
174
175 \_eoldef \_inchap #1{\_par \_sectionlevel=1
176 \_def \_savedtitle {#1}% saved to .ref file
177 \_ifnonum \_else {\_globaldefs=1 \_incr\_chapnum \_chapx}\_fi
178 \_edef \_therefnum {\_ifnonum \_space \_else \_thechapnum \_fi}%
179 \_printchap{#1}%
180 \_resetnonumnotoc
181 }
182 \_eoldef \_insec #1{\_par \_sectionlevel=2
183 \_def \_savedtitle {#1}% saved to .ref file
184 \_ifnonum \_else {\_globaldefs=1 \_incr\_secnum \_secx}\_fi
185 \_edef \_therefnum {\_ifnonum \_space \_else \_theseccnum \_fi}%
186 \_printsec{#1}%
187 \_resetnonumnotoc
188 }
189 \_eoldef \_insecc #1{\_par \_sectionlevel=3
190 \_def \_savedtitle {#1}% saved to .ref file
191 \_ifnonum \_else {\_globaldefs=1 \_incr\_seccnum \_seccx}\_fi
192 \_edef \_therefnum {\_ifnonum \_space \_else \_theseccnum \_fi}%
193 \_printsecc{#1}%
194 \_resetnonumnotoc
195 }
196 \_public \chap \sec \secc ;

```

The `\_printrefnum[⟨pre⟩@⟨post⟩]` macro is used in `\_print*` macros.

The `\_wtotoc{⟨level⟩}{⟨info⟩}{⟨ref-num⟩}{⟨title-text⟩}` macro expands its parameters and does `\_wref`.

Note that the `⟨title-text⟩` is `\detokenized` before `\_wref`, so the problem of “fragile macros” from old L<sup>A</sup>T<sub>E</sub>X never occurs.

sections.opm

```
208 \_def \_printrefnum [#1@#2]{\_leavevmode % we must be in horizontal mode
209 \_ifnonum \_else #1\_therefnun #2\_fi
210 \_wlabel \_therefnun % references, if \_label[<label>]` is declared
211 \_ifnotoc \_else \_incr \_tocrefnum
212 \_dest[toc:\_the\_tocrefnum]%
213 \_wtotoc{\_the\_sectionlevel}{\_secinfo}%
214 {\_therefnun}{\_detokenize\_ea{\_savedtitle}}%
215 \_fi
216 }
217 \_def \_wtotoc #1#2#3#4{\_edef\_tmp{#1}{#2}{#3}{#4}}\_ea\_wtotocA\_tmp}
218 \_def \_wtotocA #1#2#3#4{\_wref\_Xtoc{#1}{#2}{#3}{#4}}}
```

The `\_abovetitle{⟨penaltyA⟩}{⟨skipA⟩}` and `\_belowtitle{⟨skipB⟩}` pair communicates using a special penalty 11333 in vertical mode. The `\_belowtitle` puts the vertical skip (its value is saved in `\_savedtitleskip`) followed by this special penalty. The `\_abovetitle` reads `\_lastpenalty` and if it has this special value then it removes the skip used before and don’t use the parameter. The `\_abovetitle` creates `⟨skipA⟩` only if whatever previous skip is less or equal than `⟨skipA⟩`. We must save `⟨whatever-skip⟩`, remove it, create `⟨penaltyA⟩` (if `\_belowtitle` does not preceded) and create `⟨whatever-skip⟩` or `⟨skipA⟩` depending on what is greater. The amount of `⟨skipA⟩` is measured using `\setbox0=\vbox`.

sections.opm

```
234 \_newskip \_savedtitleskip
235 \_newskip \_savedlastskip
236 \_def\_abovetitle #1#2{\_savedlastskip=\_lastskip % <whatever-skip>
237 \_ifdim\_lastskip>0pt \_vskip-\_lastskip \_fi
238 \_ifnum\_lastpenalty=11333 \_vskip-\_savedtitleskip \_else #1\_fi
239 \_ifdim\_savedlastskip>0pt \_setbox0=\_vbox{#2\_global\_tmpdim=\_lastskip}%
240 \_else \_tmpdim=\_maxdimen \_fi
241 \_ifdim\_savedlastskip>\_tmpdim \_vskip\_savedlastskip \_else #2\_fi
242 }
243 \_def\_belowtitle #1#1\_global\_savedtitleskip=\_lastskip \_penalty11333 }
```

`\_nbpar` sets `\_interlinepenaty` value. `\_nl` is “new line” in text (or titles), but space in toc or headlines or outlines.

sections.opm

```
250 \_def\_nbpar{{\_interlinepenalty=10000\_endgraf}}
251
252 \_protected\_def\_nl{\_hfil\_break}
253 \_regmacro {\_def\_nl{\_unskip\_space}} {\_def\_nl{\_unskip\_space}} {\_def\_nl{ }}
254 \_regmacro {\_def\_nl{\_unskip\_space}} {\_def\_nl{\_unskip\_space}} {\_def\_nl{ }}
255
256 \_public \_nbpar \_nl ;
```

`\_firstnoindent` puts a material to `\_everypar` in order to next paragraph will be without indentation. It is useful after titles. If you dislike this feature then you can say `\let\_firstnoindent=\_relax`. The `\_wipepar` removes the material from `\_everypar`.

sections.opm

```
265 \_def \_firstnoindent {\_global\_everypar={\_wipepar \_setbox7=\_lastbox}}
266 \_def \_wipepar {\_global\_everypar={}}
```

The `\_mark` (for running heads) is used in `\_printsection` only. We suppose that chapters will be printed after `\_vfil\_break`, so user can implement chapter titles for running headers directly by macros, no `\_mark` mechanism is needed. But sections need `\_marks`. And they can be mixed with chapter’s running heads, of course.

The `\_insertmark{⟨title text⟩}` saves `\_mark` in the format `{⟨title-num⟩} {⟨title-text⟩}`, so it can be printed “as is” in `\_headline` (see the space between them), or you can define a formatting macro with two parameters for processing these data, if you need it.

sections.opm

```
281 \_def\_insertmark#1{\_mark{\_ifnonum\_else\_therefnun\_fi} {\_unexpanded{#1}}}
```

OpT<sub>E</sub>X sets `\_headline={}` by default, so no running headings are printed. You can activate the running headings by following code, for example:

```

\addto\chapx {\_edef\_runningchap {\_thecapnum: \_unexpanded\_ea{\_savedtitle}}}
\def \formathead #1#2{\ifempty{#1}\iffalse #1: #2\fi}
\headline = {%
  \ifodd \pageno
    \hfil \ea\formathead\firstmark{}\fi}%
  \else
    Chapter: \runningchap \hfil
  \fi
}

```

The `\caption`/`\letter` uses `\_<letter>num` counter. The group opened by `\caption` is finalized by first `\par` from empty line or from `\vskip` or from `\endinsert`. The `\_printcaption<letter>` is called, it starts with printing of the caption.

The `\cskip` macro inserts nobreakable vertical space between caption and the object.

sections.opm

```

306 \_def\_caption/#1{\_def\_tmpa{#1}\_nospaceafter \_capA}
307 \_optdef\_capA []{\_trylabel \_incaption}
308 \_def\_incaption {\_bgroup
309   \_ifcsname \_tmpa num\_endcsname \_ea\_incr \_csname \_tmpa num\_endcsname
310   \_else \_opwarning{Unknown caption /\_tmpa}\_fi
311   \_edef\_thecapnum {\_csname \_the\_tmpa num\_endcsname}%
312   \_edef\_thecaptitle{\_mtext{\_tmpa}}%
313   \_ifcsname \_everycaption\_tmpa\_endcsname
314   \_ea\_the \_csname \_everycaption\_tmpa\_endcsname \_fi
315   \_def\_par{\_nbp\_egroup}\_let\par=\_par
316   \_cs{\_printcaption\_tmpa}%
317 }
318 \_def \_cskip {\_par\_nobreak\_medskip} % space between caption and the object
319
320 \_public \_caption \_cskip ;

```

The `\_printcaptiont` and `\_printcaptionf` macros start in vertical mode. They switch to horizontal mode and use `\_wlabel\_thecapnum` (in order to make reference and hyperlink destination) a they can use:

- `\_thecaptitle ...` expands to the word Table or Figure (depending on the current language).
- `\_thecapnum ...` expands to `\the<letter>num` (caption number).

sections.opm

```

333 \_def \_printcaptiont {%
334   \_noindent \_wlabel\_thecapnum {\_bf\_thecaptitle~\_thecapnum}\_enspace
335   \_narrowlastlinecentered\_iindent
336 }
337 \_let \_printcaptionf = \_printcaptiont % caption of figures = caption of tables

```

The default format of `\caption` text is paragraph in block narrower by `\_iindent` and with the last line is centered. This setting is done by the `\_narrowlastlinecentered` macro.

sections.opm

```

345 \_def\_narrowlastlinecentered#1{%
346   \_leftskip=#1plus1fil
347   \_rightskip=#1plus-1fil
348   \_parfillskip=0pt plus2fil
349 }

```

`\eqmark` is processed in display mode (we add `\eqno` primitive) or in internal mode when `\eqaligno` is used (we don't add `\eqno`).

sections.opm

```

356 \_optdef\_eqmark []{\_trylabel \_ineqmark}
357 \_def\_ineqmark{\_incr\_dnum
358   \_ifinner\_else\_eqno \_fi
359   \_wlabel\_thednum \_thednum
360 }
361 \_public \eqmark ;

```

The `\numberedpar <letter>\{<name>\}` is implemented here.

sections.opm

```

367 \_newcount\_counterA \_newcount\_counterB \_newcount\_counterC
368 \_newcount\_counterD \_newcount\_counterE

```

```

369
370 \def\resetABCDE {\_counterA=0 \_counterB=0 \_counterC=0 \_counterD=0 \_counterE=0 }
371
372 \def \_theAnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterA}
373 \def \_theBnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterB}
374 \def \_theCnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterC}
375 \def \_theDnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterD}
376 \def \_theEnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterE}
377
378 \def\_\_numberedpar#1#2{\_ea \_incr \_csname \_counter#1\_endcsname
379 \_def\_tmpa{#1}\_def\_tmpb{#2}\_\_numberedparparam}
380 \_optdef\_\_numberedparparam[]{%
381 \_ea \_printnumberedpar \_csname \_the\_tmpa num\_ea\_endcsname\_ea{\_tmpb}}
382
383 \_public \_\_numberedpar ;

```

The `\_\_printnumberedpar \_theXnum {\langle name \rangle}` opens numbered paragraph and prints it. The optional parameter is in `\_the\_opt`. You can re-define it if you need another design.

`\_\_printnumberedpar` needs not to be re-defined if you only want to print Theorems in italic and to insert vertical skips (for example). You can do this by the following code:

```

\def\theorem {\medskip\bgroup\it \_\_numberedpar A{Theorem}}
\def\endtheorem {\par\egroup\medskip}

\theorem Let  $M$  be... \endtheorem

```

sections.opm

```

401 \_def \_\_printnumberedpar #1#2{\_par
402 \_noindent\_wlabel #1%
403 {\_bf #2 #1\_\_istokseempty\_opt\_iffalse \_space \_the\_opt \_fi.}\_space
404 \_ignorespaces
405 }

```

## 2.26 Lists, items

lists.opm

```

3 \_codedecl \begitems {Lists: begitems, enditems <2020-04-21>} % preloaded in format

```

`\_aboveliskip` is used above the list of items,

`\_belowliskip` is used below the list of items and

`\_interliskip` is used between items.

`\_listskipA` is used as `\listskipamount` at level 1 of items.

`\_listskipB` is used as `\listskipamount` at other levels.

`\_setlistskip` sets the skip dependent on the current level of items

lists.opm

```

14 \_def\_\_aboveliskip {\_removelastskip \_penalty-100 \_vskip\_listskipamount}
15 \_def\_\_belowliskip {\_penalty-200 \_vskip\_listskipamount}
16 \_def\_\_interliskip {}
17 \_def\_\_listskipA {\_medskipamount}
18 \_def\_\_listskipB {0pt plus.5\_smallskipamount}
19
20 \_def\_\_setlistskip {%
21 \_ifnum \_ilevel = 1 \_listskipamount = \_listskipA \_relax
22 \_else \_listskipamount = \_listskipB \_relax
23 \_fi}

```

The `\itemnum` is locally reset to zero in each group declared by `\begitems`. So nested lists are numbered independently. User can set initial value of `\itemnum` to another value after `\beitems` if he/she want.

Each level of nested lists is indented by new `\iindent` from left. Default item mark is `\_printitem`.

The `\begitems` runs `\_aboveliskip` only if we are not near below a title, where a vertical skip is placed already and where the `\penalty 11333` is. It activates `*` and defines it as `\_startitem`.

The `\enditems` runs `\_isnextchar\_par{}\_noindent` thus the next paragraph is without indentation if there is no empty line between the list and this paragraph (it is similar behavior as after display math).

lists.opm

```

42 \_newcount\_itemnum \_itemnum=0
43 \_newtoks\_printitem

```

```

44
45 \def\beginitems{\par
46   \bgroup
47   \advance \ilevel by1
48   \setlistskip
49   \ifnum\lastpenalty<10000 \abovelispenalty \fi
50   \itemnum=0 \edef*\startitem{
51   \advance\leftskip by\iindent
52   \printitem=\defaultitem
53   \the\everylist \relax
54 }
55 \def\enditems{\par\belowlisp\egroup \isnextchar\par{}\noindent}}
56
57 \def\startitem{\par \ifnum\itemnum>0 \interlispenalty \fi
58   \advance\itemnum by1
59   \the\everyitem \noindent\llap{\the\printitem}\ignorespaces
60 }
61 \public \beginitems \enditems \itemnum ;

```

`\novspaces` sets `\listskipamount` to 0pt.

lists.opm

```

67 \def\novspaces {\removelastskip \listskipamount=0pt \relax}
68 \public \novspaces ;

```

Various item marks are saved in `\_item:<letter>` macros. You can re-define them or define more such macros. The `\style <letter>` does `\_printitem={\_item:<letter>}`. More exactly: `\beginitems` does `\_printitem=\defaultitem` first, then `\style <letter>` does `\_printitem={\_item:<letter>}` when it is used and finally, `\startitem` alias `*` uses `\_printitem`.

lists.opm

```

79 \def\style#1{%
80   \ifcsname _item:#1\endcsname \_printitem=\ea{\csname _item:#1\endcsname}%
81   \else \_printitem=\defaultitem \fi
82 }
83 \sdef{item:o}{\raise.4ex\hbox{$\scriptscriptstyle\bullet$} }
84 \sdef{item:-}{- }
85 \sdef{item:n}{\the\itemnum. }
86 \sdef{item:N}{\the\itemnum) }
87 \sdef{item:i}{(\romannumeral\itemnum) }
88 \sdef{item:I}{\uppercase\ea{\romannumeral\itemnum}\kern.5em}
89 \sdef{item:a}{\athe\itemnum) }
90 \sdef{item:A}{\uppercase\ea{\athe\itemnum)} }
91 \sdef{item:x}{\raise.3ex\fullrectangle{.6ex}\kern.4em}
92 \sdef{item:X}{\raise.2ex\fullrectangle{1ex}\kern.5em}

```

`\athe{<num>}` returns the `<num>`s lowercase letter from the alphabet.

`\fullrectangle{<dimen>}` prints full rectangle with given `<dimen>`.

lists.opm

```

99 \def\fullrectangle#1{\hbox{\vrule height#1 width#1}}
100
101 \def\athe#1{\ifcase#1?\_or a\_or b\_or c\_or d\_or e\_or f\_or g\_or h\_or
102   i\_or j\_or k\_or l\_or m\_or n\_or o\_or p\_or q\_or r\_or s\_or t\_or
103   u\_or v\_or w\_or x\_or y\_or z\_else ?\_fi
104 }
105 \public \style ;

```

## 2.27 Verbatim, listings

### 2.27.1 Inline and “display” verbatim

verbatim.opm

```

3 \codeldecl \begtt {Verbatim <2020-04-22>} % preloaded in format

```

The internal parameters `\ttskip`, `\ttpenalty`, `\viline`, `\vifile` and `\ttfont` for verbatim macros are set.

verbatim.opm

```

11 \def\ttskip{\medskip}           % space above and below \begtt, \verinput
12 \mathchardef\ttpenalty=100      % penalty between lines in \begtt, \verinput
13 \newcount\viline                % last line number in \verinput

```



```

14 \_newread\_vifile           % file given by \verinput
15 \_def\_ttfont{\_tt}         % default tt font

```

`\code{<text>}` expands to `\detokenize{<text>}` when `\escapechar=-1`. In order to do it more robust when it is used in `\write` then it expands as noexpanded `\code{<space>}` (followed by space in its csname). This macro does the real work.

The `\_printinverbatim{<text>}` macro is used for `\code{<text>}` printing and for ``<text>`` printing. It is defined as `\hbox`, so the in-verbatim `<text>` will be never broken. But you can re-define this macro.

When `\code` occurs in PDF outlines then it does the same as `\detokenize`. The macro for preparing outlines sets `\escapechar` to `-1` and uses `\_regoul` token list before `\edef`.

The `\code` is not `\protected` because we want it expands to `\unexpanded{\code{<space>}{<text>}}` in `\write` parameters. This protect the expansions of the `\code` parameter (like `\\`, `\^` etc.).

verbatim.opm

```

36 \_def\_code#1{\_unexpanded\_ea{\_csname\_code\_endcsname{#1}}}
37 \_protected\_sdef{\_code}#1{{\_escapechar=-1\_ttfont\_the\_everyintt\_relax
38 \_ea\_printinverbatim\_ea{\_detokenize{#1}}}}
39 \_def\_printinverbatim#1{\_leavevmode\_hbox{#1}}
40
41 \_regmacro {}{}{\_let\_code=\_detokenize\_let\_code=\_detokenize}
42 \_public \code ;

```

The `\_setverb` macro sets all catcodes to “verbatim mode”. It should be used only in a group, so we prepare a new catcode table with “verbatim” catcodes and we define it as

`\_catcodetable\_verbatimcatcodes`. After the group is finished then original catcode table is restored.

verbatim.opm

```

51 \_newcatcodetable \_verbatimcatcodes
52 \_def\_setverb{\_begingroup
53 \_def\do##1{\_catcode`##1=12 }
54 \_dospecials
55 \_savecatcodetable\_verbatimcatcodes % all characters are normal
56 \_endgroup
57 }
58 \_setverb
59 \_def\_setverb{\_catcodetable\_verbatimcatcodes}%

```

`\activettchar<char>` saves original catcode of previously declared `<char>` (if such character was declared) using `\_savedttchar` and `\_savedttcharc` values. Then new such values are stored. The declared character is activated by `\_adeft` as a macro (active character) which opens a group, does `\_setverb` and other settings and reads its parameter until second the same character. This is done by the `\_readverb` macro. Finally it prints scanned `<text>` by `\_printinverbatim` and closes group. Suppose that `\activettchar` is used. Then the following work is schematically done:

```

\_def "{\_begingroup\_setverb ...\_readverb}
\_def \_readverb #1{"\_printinverbatim{#1}\_endgroup}

```

Note that the second occurrence of `"` is not active because `\_setverb` deactivates it.

verbatim.opm

```

78 \_def\_activettchar#1{%
79 \_ifx\_savedttchar\_undefined\_else \_catcode\_savedttchar=\_savedttcharc\_fi
80 \_chardef\_savedttchar=`#1
81 \_chardef\_savedttcharc=\_catcode`#1
82 \_adeft{#1}{\_begingroup\_setverb\_adeft{}{}\_ttfont\_the\_everyintt\_relax\_readverb}%
83 \_def\_readverb ##1#1{\_printinverbatim{##1}\_endgroup}%
84 }
85 \_public \activettchar ;

```

`\begtt` is defined only as public. We don't need private `\_begtt` variant. This macro is defined by `\eoldef`, so user can put a parameter at the same line where `\begtt` is. This `#1` parameter is used after `\_everytt` parameters settings, so user can change them locally.

The `\begtt` macro opens group, does `\_setverb` and another preprocessing, sets `\endlinechar` to `^^J` and reads the following text in verbatim mode until `\endtt` occurs. This scanning is done by `\_startverb` macro which is defined as:

```

\_def\_startverb #1\endtt #2^^J{...}

```

We must to ensure that the backslash in `\endtt` has category 12 (this is a reason of the `\ea` chain in real code). The `#2` is something between `\endtt` and end of the same line and it is simply ignored.

The `\_startverb` puts the scanned data to `\_prepareverldata`. It sets the data to `\_tmpb` without changes by default, but you should re-define it in order to do special changes, if you want. (For example, `\hisyntax` redefines this macro.) The scanned data have `^^J` at each end of line and all spaces are active characters (defined as `\_`). Other characters have normal category 11 or 12.

When `\_prepareverldata` finishes then `\_startverb` runs `\_printverb` loop over each line of the data and does a final work: last skip plus `\noindent` in the next paragraph.

The `\_printverb` macro calls `\_printverblines{<line>}` to each scanned line of verbatim text. This macro expect that it starts in vertical mode and it must do `\par` in order to return the vertical mode. The `\_printverblines` is used here: it does nothing when `\_ttline<0` else it prints the line number using `\_llap`.

verbatim.opm

```

123 \_eoldef \begtt#1{\_par \_wipepar
124 \_vskip\_parskip \_ttskip
125 \_begingroup
126 \_setverb
127 \_ifnum\_ttline<0 \_let\_printverblines=\_relax \_else \_initverblines \_fi
128 \_adef{ }{\ } \_adef{^^I{\t}\_parindent=\_ttindent \_parskip=0pt
129 \_def\t{\_hskip \_dimexpr\_tabspace em/2\relax}%
130 \_the\_everytt \_relax #1\relax \_ttfont
131 \_endlinechar=^^J
132 \_startverb
133 }
134 \_ea\_def\_ea\_startverb \_ea#\_ea1\_csstring\endtt#2^^J{%
135 \_prepareverldata\_tmpb{#1^^J}%
136 \_ea\_printverb \_tmpb\_end
137 \_par
138 \_endgroup \_ttskip
139 \_isnextchar\_par{}{\_noindent}%
140 }
141 \_def\_printverb #1^^J#2{\_ifx\_end#2
142 \_bgroup \_adef{ }{\ } \_def\t{}%
143 \_ifcat&#1&\_egroup \_else\_egroup \_printverblines{#1}\_fi
144 \_else
145 \_printverblines{#1}%
146 \_ea \_printverb \_ea #2%
147 \_fi
148 }
149 \_def\_prepareverldata#1#2{\_def#1{#2}}
150 \_def\_printverblines#1{\_penalty \_tptpenalty
151 \_indent \_printverblines \_kern\_ttshift #1\par}
152 \_def\_initverblines{\_tenrm \_thefontscale[700]\_ea\_let\_ea\_sevenrm\_the\_font}
153 \_def\_printverblines{\_global\_advance\_ttline by1 \_llap{\_sevenrm \_the\_ttline\_kern.9em}}

```

Macro `\_verbininput` uses a file read previously or opens the given file. Then it runs the parameter scanning by `\_viscanparameter` and `\_viscanminus`. Finally the `\_doverbininput` is run. At beginning of `\_doverbininput`, we have `\_vilines=` number of lines already read using previous `\_verbininput`, `\_vinolines=` the number of lines we need to skip and `\_vidolines=` the number of lines we need to print. Similar preparation is done as in `\begtt` after the group is opened. Then we skip `\_vinolines` lines in a loop `a` and we read `\_vidolines` lines. The read data is accumulated into `\_tmpb` macro. The next steps are equal to the steps done in `\_startverb` macro: data are processed via `\_prepareverldata` and printed via `\_printverb` loop.

verbatim.opm

```

169 \_def\_verbininput #1(#2) #3 {\_par \_def\_tmpa{#3}%
170 \_def\_tmpb{#1}% cmds used in local group
171 \_ifx\_vifilename\_tmpa \_else
172 \_openin\_vifile={#3}%
173 \_global\_vilines=0 \_global\_let\_vifilename=\_tmpa
174 \_ifeof\_vifile
175 \_opwarning{\_noexpand\_verbininput - file "#3" is unable to reading}
176 \_ea\_ea\_ea\_skiptorelax
177 \_fi
178 \_fi
179 \_viscanparameter #2+\_relax
180 }
181 \_def\_skiptorelax#1\_relax{}
182

```

```

183 \_def \_viscanparameter #1+#2\_relax{%
184   \_if$#2$_\_viscanminus(#1)\_else \_viscanplus(#1+#2)\_fi
185 }
186 \_def\_viscanplus(#1+#2){%
187   \_if$#1$_\_tmpnum=\_viline
188   \_else \_ifnum#1<0 \_tmpnum=\_viline \_advance\_tmpnum by-#1
189     \_else \_tmpnum=#1
190       \_advance\_tmpnum by-1
191       \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi % (0+13) = (1+13)
192   \_fi \_fi
193   \_edef\_vinolines{\_the\_tmpnum}%
194   \_if$#2$_\_def\_vidolines{0}\_else\_edef\_vidolines{#2}\_fi
195   \_doverbininput
196 }
197 \_def\_viscanminus(#1-#2){%
198   \_if$#1$_\_tmpnum=0
199   \_else \_tmpnum=#1 \_advance\_tmpnum by-1 \_fi
200   \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi % (0-13) = (1-13)
201   \_edef\_vinolines{\_the\_tmpnum}%
202   \_if$#2$_\_tmpnum=0
203   \_else \_tmpnum=#2 \_advance\_tmpnum by-\_vinolines \_fi
204   \_edef\_vidolines{\_the\_tmpnum}%
205   \_doverbininput
206 }
207 \_def\_doverbininput{%
208   \_tmpnum=\_vinolines
209   \_advance\_tmpnum by-\_viline
210   \_ifnum\_tmpnum<0
211     \_openin\_vfile={\_vifilename}%
212     \_global\_viline=0
213   \_else
214     \_edef\_vinolines{\_the\_tmpnum}%
215     \_fi
216   \_vskip\_parskip \_ttskip \_wipeepar
217   \_begingroup
218   \_ifnum\_ttline<-1 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
219   \_setverb \_adeft{ }{ } \_adeft{^I}{\_parindent=\_ttindent \_parskip=0pt}
220   \_def\t{\_hskip \_dimexpr\_tabspace em/2\_relax}%
221   \_the\_everytt\_relax \_tmpb\_relax \_ttfont
222   \_endlinechar=^^J \_tmpnum=0
223   \_loop \_ifeof\_vfile \_tmpnum=\_vinolines\_space \_fi
224     \_ifnum\_tmpnum<\_vinolines\_space
225       \_vireadline \_advance\_tmpnum by1 \_repeat %% skip lines
226     \_edef\_ttlinesave{\_ttline=\_the\_ttline}%
227     \_ifnum\_ttline=-1 \_ttline=\_viline \_fi
228     \_tmpnum=0 \_def\_tmpb{}%
229     \_ifnum\_vidolines=0 \_tmpnum=-1 \_fi
230     \_ifeof\_vfile \_tmpnum=\_vidolines\_space \_fi
231     \_loop \_ifnum\_tmpnum<\_vidolines\_space
232       \_vireadline
233       \_ifnum\_vidolines=0 \_else\_advance\_tmpnum by1 \_fi
234       \_ifeof\_vfile \_tmpnum=\_vidolines\_space \_else \_visaveline \_fi %% save line
235     \_repeat
236     \_ea\_prepareverbdata \_ea \_tmpb\_ea{\_tmpb^^J}%
237     \_ea\_printverb \_tmpb\_end
238     \_global\_ttlinesave
239     \_par
240     \_endgroup
241     \_ttskip
242     \_isnextchar\_par{}{\_noindent}%
243 }
244 \_def\_vireadline{\_read\_vfile to \_tmp \_global\_advance\_viline by1 }
245 \_def\_visaveline{\_ea\_addto\_ea\_tmpb\_ea{\_tmp}}
246
247 \_public \_verbininput ;

```

The `\visiblep` sets spaces as visible characters `□`. It redefines `\_` primitive, so it is useful for verbatim modes only.

```

254 \_def \_visiblesp{\_ifx\_initunifonts\_relax \_def\ {\_char9251 }%
255         \_else \_def\ {\_char32 }\_fi}
256
257 \_public \_visiblesp ;

```

## 2.27.2 Listings with syntax highlighting

The user can write

```

\_begtt \_hisyntax{C}
...
\_endtt

```

and the code is colorized by C syntax. The user can write `\everytt={\_hisyntax{C}}` and all verbatim listings are colorized.

The `\_hisyntax{<name>}` reads the file `hisyntax-<name>.opm` where the colorization is declared. The parameter `<name>` is case insensitive and the file name must include it in lowercase letters. For example the file `hisyntax-c.opm` looks like:

hisyntax-c.opm

```

3 \_codedecl \_hisyntaxc {Syntax highlighting for C sources <2020-04-03>}
4
5 \_newtoks \_hisyntaxc \_newtoks \_hicolorsc
6
7 \_global\_hicolorsc={%      colors for C language
8   \_hicolor K \Red      % Keywords
9   \_hicolor S \Magenta  % Strings
10  \_hicolor C \Green     % Comments
11  \_hicolor N \Cyan      % Numbers
12  \_hicolor P \Blue      % Preprocessor
13  \_hicolor O \Blue      % Non-letters
14 }
15 \_global\_hisyntaxc={%
16   \_the\_hicolorsc
17   \_let\c=\_relax \_let\e=\_relax \_let\o=\_relax
18   \_replfromto {/}{*}{/}      {\x C{/##1*/}}% /*...*/
19   \_replfromto {/}{^~J}      {\z C{/##1}^~J}% //...
20   \_replfromto {\_string#}{^~J} {\z P{\_string#}^~J}% #include ...
21   \_replthis {\_string"}      {\_string"}% \" protected inside strings
22   \_replfromto {"}{"}      {\x S{"#1"}}% "...
23   %
24   \_edef\_tmpa {()\_string{\_string}+~*/=[<>,;\_pcent\_string&\_string^|!~}% non-letters
25   \_ea \_foreach \_tmpa
26     \_do {\_replthis{#1}{\n\o#1\n}}
27   \_foreach                                     % keywords
28     {auto}{break}{case}{char}{continue}{default}{do}{double}%
29     {else}{entry}{enum}{extern}{float}{for}{goto}{if}{int}{long}{register}%
30     {return}{short}{sizeof}{static}{struct}{switch}{typedef}{union}%
31     {unsigned}{void}{while}
32     \_do {\_replthis{\n#1\n}{\z K{#1}}}
33   \_replthis{.}{\n.\n}                                     % numbers
34   \_foreach 0123456789
35     \_do {\_replfromto{\n#1}{\n}{\c#1##1\e}}
36   \_replthis{e.\c}{.}
37   \_replthis{e.\n}{.e}
38   \_replthis{n.\c}{\c.}
39   \_replthis{e\o+\c}{e+}\_replthis{e\o-\c}{e-}
40   \_replthis{E\o+\c}{E+}\_replthis{E\o-\c}{E-}
41   \_def\o#1{\z O{#1}}
42   \_def\c#1{\z N{#1}}
43 }

```

OpTeX provides `hisyntax-{c,python,tex,html}.opm` files. You can take inspiration from these files and declare more languages.

User can re-declare colors by `\_hicolors={...}`. This value has precedence before `\_hicolors<name>` values declared in the `hicolors-<name>.opm` file. What exactly to do: copy `\_hicolors<name>={...}` from `hicolors-<name>.opm` to your document, rename it as `\_hicolors={...}` and do you own colors modifications.

Another way to set non-default colors is to declare `\newtoks\hicolors<name>` (without the `_` prefix) and set the colors palette here. It has precedence before `\_hicolors<name>` (with the `_` prefix) declared in the `hicolors-<name>.opm` file. This is useful when there are more hi-syntax languages used in one document.

### Notes for hi-syntax macro writers

The file `hisyntax-<name>.opm` is read only once in the  $\TeX$  group. If there are definitions then they must be declared as global.

The `hisyntax-<name>.opm` file must (globally) declare `\_hisyntax<name>` tokens string where the action over verbatim text is declared typically by `\replfromto` or `\replthis` macros.

The verbatim text is prepared by *pre-processing phase*, then the `\_hisyntax<name>` is applied and then *post-processing phase* does final corrections. Finally, the verbatim text is printed line by line.

The pre-processing phase does:

- Each space is replaced by `\n\_\n`, so `\n<word>\n` should be a pattern to finding whole words (no subwords). The `\n` control sequence is removed in the post-processing phase.
- Each end of line is represented by `\n^^J\n`.
- The `\_start` control sequence is added before the verbatim text and `\_end` control sequence is appended to the end of the verbatim text. These control sequences are removed in post-processing phase.

There are special macros working only in a group when processing the verbatim text.

- `\n` means noting but it should be used as a boundary of words as mentioned above.
- `\t` means a tabulator. It is prepared as `\n\t\n` because it can be at the boundary of a word.
- `\x <letter>{<text>}` can be used as replacing text. Suppose the example

`\replfromto{/{*}/{*/}{\x C{/*#1*/}}`

This replaces all C comments `/*...*/` by `\x C{/*...*/}`. But the C comments may span more lines, i.e. the `^^J` should be inside it.

The macro `\x <letter>{<text>}` is replaced by one or more `\z <letter>{<text>}` in post-processing phase where each parameter `<text>` of `\z` keeps inside one line. Inside-line parameters are represented by `\x C{<text>}` and they are replaced to `\z C{<text>}` without any change. But:

`\x C{<text1>}^^J<text3>^^J<text3>`  
is replaced by  
`\z C{<text1>}^^J\z C{<text2>}^^J\z C{<text3>}`

The `\z <letter>{<text>}` is expanded to `\_z:<letter>{<text>}` and if `\hicolor <letter> <color>` is declared then `\_z:<letter>{<text>}` expands to `{<color><text>}`. So, required color is activated at all lines (separately) where C comment spans.

- `\y {<text>}` is replaced by `\<text>` in the post processing phase. It should be used for macros without a parameter. You cannot use unprotected macros as replacement text before the post-processing phase, because the post-processing phase is based on expansion whole verbatim text.

hi-syntax.opm

```
3 \_codedecl \hisyntax {Syntax highlighting of verbatim listings <2020-04-04>} % preloaded in format
```

The following macros `\replfromto` and `\replthis` manipulate with the verbatim text which has been read already and stored in the `\_tmpb` macro.

The `\replfromto {<from>}{<to>}{<what>}` finds first `<from>` then the first `<to>` following by `<from>` pattern and the `<text>` between them is packed to `#1`. Then `<from><text><to>` is replaced by `<what>`. The `<what>` parameter can use `#1` which is replaced by the `<text>`.

The `\replfromto` continues by finding next `<from>`, then, next `<to>` repeatedly over the whole verbatim text. If the verbatim text is ended by opened `<from>` but not closing by `<to>` then `<to>` is appended to the verbatim text automatically and the last part of verbatim text is replaced too.

First two parameters are expanded before usage of `\replfromto`. You can use `\csstring%` or something else here.

hi-syntax.opm

```
24 \_def\replfromto #1#2{\_edef\_tmpa{{#1}{#2}}\_ea\_replfromtoE\_tmpa}
25 \_def\replfromtoE#1#2#3{% #1=from #2=to #3=what to replace
26 \_def\replfrom##1#1##2{\_addto\_tmpb{##1}%
27 \_ifx\_end##2\_ea\_replstop \_else \_afterfi{\_replto##2}\_fi}%
```

```

28 \def\replto##1#2##2{%
29 \ifx\end##2\afterfi{\replfin##1}\else
30 \addto\_tmpb{#3}%
31 \afterfi{\replfrom##2}\fi}%
32 \def\replfin##1#1\end{\_addto\_tmpb{#3}\replstop}%
33 \edef\_tmpb{\_ea\_ea\_replfrom\_tmpb#1\end#2\end\end\_relax
34 }
35 \def\replstop#1\end\_relax{}
36 \def\_finrepl{}

```

The `\replthis`  $\langle pattern \rangle \langle what \rangle$  replaces each  $\langle pattern \rangle$  by  $\langle what \rangle$ . Both parameters of `\replthis` are expanded first.

hi-syntax.opm

```

43 \def\replthis#1#2{\edef\_tmpa{{#1}{#2}}\_ea\_replstring\_ea\_tmpb\_tmpa}
44
45 \_public \replfromto \replthis ;

```

The patterns  $\langle from \rangle$ ,  $\langle to \rangle$  and  $\langle pattern \rangle$  are not found when they are hidden in braces  $\{...\}$ . Example:

```
\replfromto{/{*}/{*/}}{\x C{/*#1/*}}
```

replaces all C comments by `\x C{...}`. The patterns inside  $\{...\}$  are not used by next usage of `\replfromto` or `\replthis` macros.

The `\_xscan` macro does replacing `\x` by `\z` in the post-processing phase. The `\x`  $\langle letter \rangle \langle text \rangle$  expands to `\_xscan`  $\langle letter \rangle \langle text \rangle \textasciicircum J$ . If #3 is `\_end` then it signals that something wrong happens, the  $\langle from \rangle$  was not terminated by legal  $\langle to \rangle$  when `\replfromto` did work. We must to fix it by the `\_xscanR` macro.

hi-syntax.opm

```

63 \def\_xscan#1#2\textasciicircum J#3{\ifx\end#3\_ea\_xscanR\_fi
64 \z{#1}{#2}%
65 \ifx\end#3\_else \textasciicircum J\_afterfi{\_xscan{#1}{#3}\_fi}
66 \def\_xscanR#1\_fi#2\textasciicircum J}

```

The `\hicolor`  $\langle letter \rangle \langle color \rangle$  defines `\_z` $\langle letter \rangle \langle text \rangle$  as  $\langle color \rangle \langle text \rangle$ . It should be used in the context of `\x`  $\langle letter \rangle \langle text \rangle$  macros.

hi-syntax.opm

```
74 \def\_hicolor #1#2{\_sdef\_z:#1##1{{#2##1}}}
```

The `\hisyntax`  $\langle name \rangle$  re-defines default `\_prepareverbdata`  $\langle macro \rangle \langle verbtex \rangle$  in order to it does more things: It saves  $\langle verbtex \rangle$  to `\_tmpb`, appends `\n` around spaces and `\textasciicircum J` characters in pre-processing phase, it opens `hisyntax- $\langle name \rangle$ .opm` file if `\_hisyntax`  $\langle name \rangle$  is not defined. Then `\_the\_isyntax`  $\langle name \rangle$  is processed. Finally, the post-processing phase is realized by setting appropriate values to `\x` and `\y` macros and doing `\_edef\_tmpb{\_tmpb}`.

hi-syntax.opm

```

87 \def\_hisyntax#1{\def\_prepareverbdata##1##2{%
88 \_let\n=\_relax \_def\t{\n\_noexpand\t\n}\_let\_start=\_relax
89 \_adef\ }{\n \n}\_edef\_tmpb{\_start\textasciicircum J##2\_end}%
90 \_replthis{\textasciicircum J}{\n\textasciicircum J\n}\_replthis{\n\_end}{\_end}%
91 \_let\x=\_relax \_let\y=\_relax \_let\z=\_relax \_let\t=\_relax
92 \_endlinechar=\textasciicircum M
93 \_lowercase{\_def\_tmpa{#1}}%
94 \_ifcsname\_hialias:\_tmpa\_endcsname \_edef\_tmpa{\_cs{hialias:\_tmpa}}\_fi
95 \_ifx\_tmpa\_empty \_else
96 \_unless\_ifcsname\_hisyntax\_tmpa\_endcsname
97 \_isfile{hisyntax-\_tmpa.opm}\_iftrue \_opinput {hisyntax-\_tmpa.opm} \_fi\_fi
98 \_ifcsname\_hisyntax\_tmpa\_endcsname
99 \_ifcsname hicolors\_tmpa\_endcsname
100 \_cs{hicolors\_tmpa}=\_cs{hicolors\_tmpa}%
101 \_else
102 \_if\textasciicircum\_the\_hicolors\_else
103 \_ifcsname hicolors\_tmpa\_endcsname
104 \_global\_cs{hicolors\_tmpa}=\_hicolors\_global\_hicolors={}%
105 \_fi\_fi\_fi
106 \_ea\_the\_csname\_hisyntax\_tmpa\_endcsname % \_the\_hisyntax<name>
107 \_else\_opwarning{Syntax highlighting "\_tmpa" undeclared (no file hisyntax-\_tmpa.opm)}
108 \_fi\_fi
109 \_replthis{\_start\n\textasciicircum J}{\_replthis{\textasciicircum J\_end}{\textasciicircum J}%
110 \_def\n{}}%

```



```

111 \def\x####1###2{\_xscan{####1}###2^^J}%
112 \def\y####1{\_ea \noexpand\_csname ####1\_endcsname}%
113 \edef\_tmpb{\_tmpb}%
114 \def\z####1{\_cs{z:####1}}%
115 \def\t{\_hskip\_dimexpr\_tabspace em/2\_relax}%
116 \_localcolor
117 }}
118 \_public \hisyntax \hicolor ;

```

Aliases for languages can be declared like this. When `\hisyntax{xml}` is used then this is the same as `\hisyntax{html}`.

hi-syntax.opm

```

125 \_sdef\_hialias:xml}{html}
126 \_sdef\_hialias:json}{c}

```

## 2.28 Graphics

graphics.opm

```

3 \_codedecl \inspic {Graphics <2020-04-12>} % preloaded in format

```

`\inspic` accepts old syntax `\inspic <filename><space>` or new syntax `\inspic{<filename>}`. So, we need to define two auxiliary macros `\_inspicA` and `\_inspicB`.

You can include more `\pdfximage` parameters (like `page<number>`) in the `\_picparams` macro.

All `\inspic` macros are surrounded in `\hbox` in order user can write `\moveright\inspic ...` or something similar.

graphics.opm

```

17 \_def\_inspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inspicB\_inspicA}
18 \_def\_inspicA #1 {\_inspicB {#1}}
19 \_def\_inspicB #1{%
20 \_pdfximage \_ifdim\_picwidth=0pt \_else width\_picwidth\_fi
21 \_ifdim\_picheight=0pt \_else height\_picheight\_fi
22 \_picparams {\_the\_picdir#1}%
23 \_pdfrefximage\_pdflastximage\_egroup}
24
25 \_def\_picparams{}
26
27 \_public \inspic ;

```

Inkscape is able to save a picture to `*.pdf` file and labels for the picture to `*.pdf_tex` file. The second file is in  $\text{\LaTeX}$  format (unfortunately) and it is intended to read immediately it after `*.pdf` in included in order to place labels of this picture in the same font as document is printed. We need to read this  $\text{\LaTeX}$  file by plain  $\text{\TeX}$  macros when `\inkinspic` is used. These macros are stored in the `\_inkdefs` tokens list and it is used locally in the group. The solution is borrowed from OPmac trick 0032.

graphics.opm

```

39 \_def\_inkinspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}
40 \_def\_inkinspicA #1 {\_inkinspicB {#1}}
41 \_def\_inkinspicB #1{%
42 \_ifdim\_picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
43 \_the\_inkdefs
44 \_opinput {\_the\_picdir #1\_tex}% file with labels
45 \_egroup}
46
47 \_newtoks\_inkdefs \_inkdefs={%
48 \_def\makeatletter#1\makeatother{%
49 \_def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{#2}\_hss}%
50 \_def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
51 \_def\put(#1,#2)#3{\_nointerlineskip\_vbox to0pt{\_vss\_hbox to0pt{\_kern#1\_picwidth
52 \_pdfsave\_hbox to0pt{#3}\_pdfrestore\_hss}\_kern#2\_picwidth}}%
53 \_def\begin#1{\_csname\_begin#1\_endcsname}%
54 \_def\_beginpicture(#1,#2){\_vbox\_bgroup
55 \_hbox to\_picwidth{\_kern#2\_picwidth \_def\end##1{\_egroup}}%
56 \_def\_beginatubular[#1]#2#3\end#4{%
57 \_vtop{\_def\\{\_cr}\_tabiteml{\\_tabitemr}\\_table{#2}{#3}}}%
58 \_def\color[#1]#2{\_scancolor #2,}%
59 \_def\_scancolor#1,#2,#3,{\_pdfliteral{#1 #2 #3 rg}}%
60 \_def\makebox(#1)[#2]#3{\_hbox to0pt{\_csname\_mbx:#2\_endcsname{#3}}}%
61 \_sdef{\_mbx:lb}#1{#1\_hss}\_sdef{\_mbx:rb}#1{#1\_hss#1}\_sdef{\_mbx:b}#1{#1\_hss#1\_hss}%

```

```

62 \_sdef{<mbx:lt>#1{#1\_hss}}\_sdef{<mbx:rt>#1{\_hss#1}}\_sdef{<mbx:t>#1{\_hss#1\_hss}}%
63 \_def\rotatebox#1#2{\_pdfrotate{#1}#2}%
64 \_def\lineheight#1{}%
65 \_def\setlength#1#2{}%
66 }
67 \_public \inkinspic ;

```

`\pdfscale{<x-scale>}{<y-scale>}` and `\pdfrotate{<degrees>}` macros are implemented by `\pdfsetmatrix` primitive. We need to know values of sin, cos function in the `\pdfrotate`. We use Lua code for this.

graphics.opm

```

76 \_def\_pdfscale#1#2{\_pdfsetmatrix{#1 0 0 #2}}
77
78 \_def\_gonfunc#1#2{%
79 \_directlua{tex.print(string.format('\_pcent.4f',math.#1(3.14159265*(#2)/180)))}%
80 }
81 \_def\_sin{\_gonfunc{sin}}
82 \_def\_cos{\_gonfunc{cos}}
83
84 \_def\_pdfrotate#1{\_pdfsetmatrix{\_cos{#1} \_sin{#1} \_sin{(#1)-180} \_cos{#1}}}
85
86 \_public \pdfscale \pdfrotate ;

```

The `\transformbox{<transformation>}{<text>}` is copied from OPmac trick 0046.

The `\rotbox{<degrees>}{<text>}` is a combination of `\rotsimple` from OPmac trick 0101 and the `\transformbox`. Note, that `\rotbox{-90}` puts the rotated text to the height of the outer box (depth is zero) because code from `\rotsimple` is processed. But `\rotbox{-90.0}` puts the rotated text to the depth of the outer box (height is zero) because `\transformbox` is processed.

graphics.opm

```

100 \_def\_multiplyMxV #1 #2 #3 #4 {% matrix * (vvalX, vvalY)
101 \_tmpdim = #1\_vvalX \_advance\_tmpdim by #3\_vvalY
102 \_vvalY = #4\_vvalY \_advance\_vvalY by #2\_vvalX
103 \_vvalX = \_tmpdim
104 }
105 \_def\_multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
106 \_vvalX=#1pt \_vvalY=#2pt \_ea\_multiplyMxV \_currmatrix
107 \_edef\_tmpb{\_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY}%
108 \_vvalX=#3pt \_vvalY=#4pt \_ea\_multiplyMxV \_currmatrix
109 \_edef\_currmatrix{\_tmpb\_space
110 \_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY\_space}%
111 }
112 \_def\_transformbox#1#2{\_hbox{\_setbox0=\_hbox{#2}}}%
113 \_dimendef\_vvalX 11 \_dimendef\_vvalY 12 % we use these variables
114 \_dimendef\_newHt 13 \_dimendef\_newDp 14 % only in this group
115 \_dimendef\_newLt 15 \_dimendef\_newRt 16
116 \_pretransform{#1}%
117 \_kern-\_newLt \_vrule height\_newHt depth\_newDp width0pt
118 \_setbox0=\_hbox{\_box0}\_ht0=0pt \_dp0=0pt
119 \_pdfsave#1\_rlap{\_box0}\_pdfrestore \_kern\_newRt}%
120 }
121 \_def\_pretransform #1{\_def\_currmatrix{1 0 0 1 }%
122 \_def\_pdfsetmatrix##1{\_edef\_tmpb{##1 }\_ea\_multiplyMxM \_tmpb\_unskip}%
123 \_let\_pdfsetmatrix=\_pdfsetmatrix #1%
124 \_setnewHtDp Opt \_ht0 \_setnewHtDp Opt -\_dp0
125 \_setnewHtDp \_wd0 \_ht0 \_setnewHtDp \_wd0 -\_dp0
126 \_protected\_def \_pdfsetmatrix {\_pdfextension setmatrix}%
127 \_let\_pdfsetmatrix=\_pdfsetmatrix
128 }
129 \_def\_setnewHtDp #1 #2 {%
130 \_vvalX=#1\_relax \_vvalY=#2\_relax \_ea\_multiplyMxV \_currmatrix
131 \_ifdim\_vvalX<\_newLt \_newLt=\_vvalX \_fi \_ifdim\_vvalX>\_newRt \_newRt=\_vvalX \_fi
132 \_ifdim\_vvalY>\_newHt \_newHt=\_vvalY \_fi \_ifdim\_vvalY>\_newDp \_newDp=-\_vvalY \_fi
133 }
134
135 \_def\_rotbox#1#2{%
136 \_isequal{90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_ht0 \_tmpdim=\_dp0}{\_vfill}{#2}%
137 \_else \_isequal{-90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_dp0 \_tmpdim=\_ht0}{#2}%
138 \_else \_transformbox{\_pdfrotate{#1}}{#2}%
139 \_fi \_fi

```

```

140 }
141 \def\rotboxA #1#2#3#4{\hbox{\setbox0=\hbox{#4}}#2%
142 \vbox to\wd0{#3\wd0=0pt \dp0=0pt \ht0=0pt
143 \pdfsave\pdfrotate{#1}\box0\pdfrestore\vfil}%
144 \kern\tmpdim
145 }}
146 \public \transformbox \rotbox ;

```

**\scantwodimens** scans two objects with the syntactic rule  $\langle \text{dimen} \rangle$  and returns  $\{\langle \text{number} \rangle\}\{\langle \text{number} \rangle\}$  in sp unit.

**\puttext**  $\langle \text{right} \rangle$   $\langle \text{up} \rangle \{\langle \text{text} \rangle\}$  puts the  $\langle \text{text} \rangle$  to desired place: From current point moves  $\langle \text{down} \rangle$  and  $\langle \text{right} \rangle$ , puts the  $\langle \text{text} \rangle$  and returns back. The current point is unchanged after this macro ends.

**\putpic**  $\langle \text{right} \rangle$   $\langle \text{up} \rangle$   $\langle \text{width} \rangle$   $\langle \text{height} \rangle$   $\{\langle \text{image-file} \rangle\}$  does **\puttext** with the image scaled to desired  $\langle \text{width} \rangle$  and  $\langle \text{height} \rangle$ . If  $\langle \text{width} \rangle$  or  $\langle \text{height} \rangle$  is zero, natural dimension is used. The **\nospec** is a shortcut to such natural dimension.

**\backgroundpic** $\{\langle \text{image-file} \rangle\}$  puts the image to the background of each page. It is used in the slides style, for example.

graphics.opm

```

165 \def\scantwodimens{%
166 \directlua{tex.print(string.format('{\pcent d}{\pcent d}',
167 token.scan_dimen(),token.scan_dimen()))}%
168 }
169
170 \def\puttext{\ea\ea\ea\puttextA\scantwodimens}
171 \def\puttextA#1#2#3{\setbox0=\hbox{#3}\_dimen1=#1sp \_dimen2=#2sp \puttextB}
172 \def\puttextB{%
173 \ifvmode
174 \ifdim\_prevdepth>0pt \vskip-\_prevdepth \relax \fi
175 \nointerlineskip
176 \fi
177 \wd0=0pt \ht0=0pt \dp0=0pt
178 \vbox to0pt{\_kern-\_dimen2 \hbox to0pt{\_kern\_dimen1 \box0\_hss}\_vss}}
179
180 \def\putpic{\ea\ea\ea\putpicA\scantwodimens}
181 \def\putpicA#1#2{\_dimen1=#1sp \_dimen2=#2sp \ea\ea\ea\putpicB\scantwodimens}
182 \def\putpicB#1#2#3{\setbox0=\hbox{\picwidth=#1sp \picheight=#2sp \inspic{#3}}\puttextB}
183
184 \newbox\_bgbox
185 \def\backgroundpic#1{%
186 \setbox\_bgbox=\hbox{\picwidth=\pdfpagewidth \picheight=\pdfpageheight \inspic{#1}}%
187 \pgbackground={\_copy\_bgbox}
188 }
189 \def\nospec{0pt}
190 \public \puttext \putpic \backgroundpic ;

```

**\circle** $\{\langle x \rangle\}\{\langle y \rangle\}$  creates an ellipse with  $\langle x \rangle$  axis and  $\langle y \rangle$  axis. The origin is in the center.

**\oval** $\{\langle x \rangle\}\{\langle y \rangle\}\{\langle \text{roundness} \rangle\}$  creates an oval with  $\langle x \rangle$ ,  $\langle y \rangle$  size and with given  $\langle \text{roundness} \rangle$ . The real size is bigger by  $2\langle \text{roundness} \rangle$ . The origin is at the left bottom corner.

**\mv** $\{\langle x \rangle\}\{\langle y \rangle\}\{\langle \text{curve} \rangle\}$  moves current point to  $\langle x \rangle$ ,  $\langle y \rangle$ , creates the  $\langle \text{curve} \rangle$  and returns back the current point. All these macros are fully expandable and they can be used in the **\pdfliteral** argument.

graphics.opm

```

206 \def\circle#1#2{\_expr{.5*(#1)} 0 m
207 \_expr{.5*(#1)} \_expr{.276*(#2)} \_expr{.276*(#1)} \_expr{.5*(#2)} 0 \_expr{.5*(#2)} c
208 \_expr{-.276*(#1)} \_expr{.5*(#2)} \_expr{-.5*(#1)} \_expr{.276*(#2)} \_expr{-.5*(#1)} 0 c
209 \_expr{-.5*(#1)} \_expr{-.276*(#2)} \_expr{-.276*(#1)} \_expr{-.5*(#2)} 0 \_expr{-.5*(#2)} c
210 \_expr{.276*(#1)} \_expr{-.5*(#2)} \_expr{.5*(#1)} \_expr{-.276*(#2)} \_expr{.5*(#1)} 0 c h}
211
212 \def\oval#1#2#3{0 \_expr{-(#3)} m \_expr{#1} \_expr{-(#3)} 1
213 \_expr{(#1)+.552*(#3)} \_expr{-(#3)} \_expr{(#1)+(#3)} \_expr{-.552*(#3)}
214 \_expr{(#1)+(#3)} 0 c
215 \_expr{(#1)+(#3)} \_expr{#2} 1
216 \_expr{(#1)+(#3)} \_expr{(#2)+.552*(#3)} \_expr{(#1)+.552*(#3)} \_expr{(#2)+(#3)}
217 \_expr{#1} \_expr{(#2)+(#3)} c
218 0 \_expr{(#2)+(#3)} 1
219 \_expr{-.552*(#3)} \_expr{(#2)+(#3)} \_expr{-(#3)} \_expr{(#2)+.552*(#3)}
220 \_expr{-(#3)} \_expr{#2} c
221 \_expr{-(#3)} 0 1

```

```

222 \_expr{-(#3)} \_expr{-.552*(#3)} \_expr{-.552*(#3)} \_expr{-(#3)} 0 \_expr{-(#3)} c h}
223
224 \def\_mv#1#2#3{1 0 0 1 \_expr{#1} \_expr{#2} cm #3 1 0 0 1 \_expr{-(#1)} \_expr{-(#2)} cm}

```

The `\inoval{<text>}` is an example of `\oval` usage.

The `\incircle{<text>}` is an example of `\circle` usage.

The `\ratio`, `\width`, `\fcolor`, `\lcolor`, `\shadow` and `\overlapmargins` are parameters, they can be set by user in optional brackets [...]. For example `\fcolor=\Red` does `\let\_fcolorvalue=\Red` and it means filling color.

The `\setflcolor` uses the `\fillstroke` macro to separate filling color and drawing color.

graphics.opm

```

237 \_newdimen \_lwidth
238 \_def\_fcolor{\_let\_fcolorvalue}
239 \_def\_lcolor{\_let\_lcolorvalue}
240 \_def\_shadow{\_let\_shadowvalue}
241 \_def\_overlapmargins{\_let\_overlapmarginsvalue}
242 \_def\_ratio{\_isnextchar ={\_ratioA}{\_ratioA=}}
243 \_def\_ratioA =#1 {\_def\_ratiovalue{#1}}
244 \_def\_toupvalue#1{\_ifx#1n\_let#1=N\_fi}
245
246 \_def\_setflcolors#1{% use only in a group
247   \_def\_setcolor##1{##1}%
248   \_def\_fillstroke##1##2{##1}%
249   \_edef#1{\_fcolorvalue}%
250   \_def\_fillstroke##1##2{##2}%
251   \_edef#1{#1\_space\_lcolorvalue\_space}%
252 }
253 \_optdef\_inoval[]{\_vbox\_bgroup
254   \_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
255   \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt
256   \_the\_ovalparams \_relax \_the\_opt \_relax
257   \_toupvalue\_overlapmarginsvalue \_toupvalue\_shadowvalue
258   \_ifx\_overlapmarginsvalue N%
259     \_advance\_hsize by-2\_hhkern \_advance\_hsize by-2\_roundness \_fi
260   \_setbox0=\hbox\_bgroup\_bgroup \_aftergroup\_inovalA \_kern\_hhkern \_let\_next=%
261 }
262 \_def\_inovalA{\_isnextchar\_colorstackpop\_inovalB\_inovalC}
263 \_def\_inovalB#1{#1\_isnextchar\_colorstackpop\_inovalB\_inovalC}
264 \_def\_inovalC{\_egroup % of \setbox0=\hbox\bgroup
265   \_ifdim\_vvkern=0pt \_else \_ht0=\_dimexpr\_ht0+\_vvkern \_relax
266     \_dp0=\_dimexpr\_dp0+\_vvkern \_relax \_fi
267   \_ifdim\_hhkern=0pt \_else \_wd0=\_dimexpr\_wd0+\_hhkern \_relax \_fi
268   \_ifx\_overlapmarginsvalue N\_dimen0=\_roundness \_dimen1=\_roundness
269   \_else \_dimen0=-\_hhkern \_dimen1=-\_vvkern \_fi
270   \_setflcolors\_tmp
271   \_hbox{\_kern\_dimen0
272     \_vbox to0pt{\_kern\_dp0
273       \_ifx\_shadowvalue N\_else
274         \_edef\_tmpb{{\_bp{\_wd0+\_lwidth}}}{\_bp{\_ht0+\_dp0+\_lwidth}}{\_bp{\_roundness}}}%
275         \_doshadow\_oval
276       \_fi
277       \_pdfliteral{q \_bp{\_lwidth} w \_tmp
278         \_oval{\_bp{\_wd0}}{\_bp{\_ht0+\_dp0}}{\_bp{\_roundness}} B Q}\_vss}%
279       \_ht0=\_dimexpr\_ht0+\_dimen1 \_relax \_dp0=\_dimexpr\_dp0+\_dimen1 \_relax
280       \_box0
281       \_kern\_dimen0}%
282   \_egroup % of \vbox\bgroup
283 }
284 \_optdef\_incircle[]{\_vbox\_bgroup
285   \_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
286   \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt
287   \_ea\_the \_ea\_circleparams \_space \_relax
288   \_ea\_the \_ea\_opt \_space \_relax
289   \_toupvalue\_overlapmarginsvalue \_toupvalue\_shadowvalue
290   \_setbox0=\hbox\_bgroup\_bgroup \_aftergroup\_incircleA \_kern\_hhkern \_let\_next=%
291 }
292 \_def\_incircleA {\_isnextchar\_colorstackpop\_incircleB\_incircleC}
293 \_def\_incircleB #1{#1\_isnextchar\_colorstackpop\_incircleB\_incircleC}

```

```

294 \def\incircleC {\_egroup % of \setbox0=\hbox\bgroup
295 \_wd0=\dimexpr \_wd0+\_hhkern \_relax
296 \_ht0=\dimexpr \_ht0+\_vvkern \_relax \_dp0=\dimexpr \_dp0+\_vvkern \_relax
297 \_ifdim \_ratiovalue\dimexpr \_ht0+\_dp0 > \_wd0
298 \_dimen3=\dimexpr \_ht0+\_dp0 \_relax \_dimen2=\_ratiovalue\dimen3
299 \_else \_dimen2=\_wd0 \_dimen3=\_expr{1/\_ratiovalue}\_dimen2 \_fi
300 \_setflcolors\_tmp
301 \_ifx\_overlapmarginvalue N\_dimen0=0pt \_dimen1=0pt
302 \_else \_dimen0=-\_hhkern \_dimen1=-\_vvkern \_fi
303 \_hbox{\_kern\_dimen0
304 \_ifx\_shadowvalue N\_else
305 \_edef\_tmpb{\_bp{\_dimen2+\_lwidth}}{\_bp{\_dimen3+\_lwidth}}{}}%
306 \_doshadow\_circlet
307 \_fi
308 \_pdfliteral{q \_bp{\_lwidth} w \_tmp \_mv{\_bp{.5\_wd0}}{\_bp{(\_ht0-\_dp0)/2}}
309 {\_circlef{\_bp{\_dimen2}}{\_bp{\_dimen3}} B} Q}%
310 \_ifdim\_dimen1=0pt \_else
311 \_ht0=\dimexpr \_ht0+\_dimen1 \_relax \_dp0=\dimexpr \_dp0+\_dimen1 \_relax \_fi
312 \_box0
313 \_kern\_dimen0}
314 \_egroup % of \vbox\bgroup
315 }
316 \def\_circlet#1#2#3{\_circle{#1}{#2}}
317
318 \_public \inoval \incircle \ratio \lwidth \fcolor \lcolor \shadow \overlapmargin ;

```

A shadow effect is implemented here. The shadow is equal to the silhouette of the given path in gray-transparent color shifted by `\_shadowmoveto` vector and with blurred boundary. A waistline with the width  $2*\_shadowb$  around the boundary is blurred. The `\shadowlevels` levels of transparent shapes is used for creating this effect. The  $\_shadowlevels+1/2$  level is equal to the shifted given path.

graphics.opm

```

329 \def\_shadowlevels{9} % number of layers for blurr effect
330 \def\_shadowdarknessA{0.025} % transparency of first shadowlevels/2 layers
331 \def\_shadowdarknessB{0.07} % transparency of second half of layers
332 \def\_shadowmoveto{1.8 -2.5} % vector defines shifting layer (in bp)
333 \def\_shadowb{1} % 2*shadowb = blurring area thickness

```

The `\_pdfpageresources` primitive is used to define transparency. It does not work when used in a box. So, we use it at the beginning of the output routine. The modification of the output routine is done using `\_insertshadowresources` only once when the shadow effect is used first.

graphics.opm

```

342 \def\_insertshadowresources{%
343 \_global\_addto\_begoutput{\_setshadowresources}%
344 \_xdef\_setshadowresources{%
345 \_pdfpageresources{/ExtGState
346 <<
347 /op1 <</Type /ExtGState /ca \_shadowdarknessA>>
348 /op2 <</Type /ExtGState /ca \_shadowdarknessB>>
349 \_morepgresources
350 >>
351 }%
352 }%
353 \_global\_let\_insertshadowresources=\_relax
354 }
355 \def\_morepgresources{

```

The `\_doshadow{<curve>}` does the shadow effect.

graphics.opm

```

361 \def\_doshadow#1{\_vbox{%
362 \_insertshadowresources
363 \_tmpnum=\_numexpr (\_shadowlevels-1)/2 \_relax
364 \_edef\_tmpfin{\_the\_tmpnum}%
365 \_ifnum\_tmpfin=0 \_def\_shadowb{0}\_def\_shadowstep{0}%
366 \_else \_edef\_shadowstep{\_expr{\_shadowb/\_tmpfin}}\_fi
367 \_def\_tmpa##1##2##3{\_def\_tmpb
368 {#1##1+2*\_the\_tmpnum*\_shadowstep}{##2+2*\_the\_tmpnum*\_shadowstep}{##3}}%
369 \_ea \_tmpa \_tmpb
370 \_def\_shadowlayer{%
371 \_ifnum\_tmpnum=0 /op2 gs \_fi

```

```

372     \tmpb\space f
373     \immediateassignment\advance\tmpnum by-1
374     \ifnum-\tmpfin<\tmpnum
375         \ifx#1\oval 1 0 0 1 \shadowstep\space \shadowstep\space cm \fi
376         \ea \shadowlayer \fi
377     }%
378     \pdfliteral{q /op1 gs 0 g 1 0 0 1 \shadowmoveto\space cm
379         \ifx#1\circlet 1 0 0 1 \expr{\bp{.5\wd0}} \expr{\bp{(\ht0-\dp0)/2}} cm
380         \else 1 0 0 1 -\shadowb\space -\shadowb\space cm \fi
381         \shadowlayer Q}
382 }

```

A generic macro `\clipinpath<x> <y> <curve> <text>` declares a clipping path by the `<curve>` shifted by the `<x>`, `<y>`. The `<text>` is typeset when such clipping path is active. Dimensions are given by bp without the unit here. The macros `\clipinoval <x> <y> <width> <height> {\text}` and `\clipincircle <x> <y> <width> <height> {\text}` are defined here. These macros read normal T<sub>E</sub>X dimensions in their parameters.

graphics.opm

```

393 \def\clipinpath#1#2#3#4{% #1=x-pos[bp], #2=y-pos[bp], #3=curve, #4=text
394     \hbox{\setbox0=\hbox{#4}}%
395     \tmpdim=\wd0 \wd0=0pt
396     \pdfliteral{q \mv{#1}{#2}{#3 W n}}%
397     \box0\pdfliteral{Q}\kern\tmpdim
398 }%
399 }
400
401 \def\clipinoval {\ea\ea\ea\clipinovalA\scantwodimens}
402 \def\clipinovalA #1#2{%
403     \def\tmp{#1/65781.76}{#2/65781.76}}%
404     \ea\ea\ea\clipinovalB\scantwodimens
405 }
406 \def\clipinovalB{\ea\clipinovalC\tmp}
407 \def\clipinovalC#1#2#3#4{%
408     \ea\clipinpath{#1-(#3/131563.52)+(\bp{\_roundness})}{#2-(#4/131563.52)+(\bp{\_roundness})}%
409     {\oval{#3/65781.76-(\bp{2\_roundness})}{#4/65781.76-(\bp{2\_roundness})}{\bp{\_roundness}}}%
410 }
411 \def\clipincircle {\ea\ea\ea\clipincircleA\scantwodimens}
412 \def\clipincircleA #1#2{%
413     \def\tmp{#1/65781.76}{#2/65781.76}}%
414     \ea\ea\ea\clipincircleB\scantwodimens
415 }
416 \def\clipincircleB#1#2{%
417     \ea\clipinpath\tmp{\circle{#1/65781.76}{#2/65781.76}}%
418 }
419 \_public \clipinoval \clipincircle ;

```

## 2.29 The \table macro

### 2.29.1 The boundary declarator :

The `<declaration>` part of `\table{<declaration>}{<data>}` includes column declarators (letters) and other material: the `|` or `(<cmd>)`. The boundaries of columns are just before each column declarator (with exception of the first one) if the boundary declarator `:` is not used. For example, the declaration `{|c||c(xx)(yy)c}` should be written more exactly using the boundary declarator `:` by `{|c|:c(xx)(yy):c}`. But you can set these boundaries to another places using the boundary declarator `:` explicitly, for example `{|c:|c(xx):(yy)c}`. The boundary declarator `:` can be used only once between each two column declarators.

Each table item have its own group. The `(<cmd>)` are parts of the given table item (depending on the boundary declarator position). If you want to apply a special setting for given column, you can do this by `(<setting>)` followed by column declarator. But if such column is not first, you must use `:(<setting>)`. Example. We have three centered columns, the second one have to be in bold font and the third one have to be in red: `\table{c:(\bf)c:(\Red)c}{<data>}`

### 2.29.2 Usage of the \tabskip primitive

The value of `\tabskip` primitive is used between all columns of the table. It is glue-type, so it can be stretchable or shrinkable, see next section 2.29.3.



By default, `\tabskip` is 0pt. It means that only `\tabiteml`, `\tabitemr` and (`\langle cnds \rangle`) can generate visual spaces between columns. But they are not real spaces between columns because they are in fact the part of the total column width.

The `\tabskip` value declared before the `\table` macro (or in `\everytable` or in `\thistable`) is used between all columns in the table. This value is equal for all spaces between columns. But you can set each such space individually if you use (`\tabskip=\langle value \rangle`) in the (`\langle declaration \rangle`) immediately before boundary character. The boundary character represents the column pair for which the `\tabskip` have individual value. For example `c(\tabskip=5pt):r` gives `\tabskip` value between `c` and `r` columns. You need not to use boundary character explicitly, so `c(\tabskip=5pt)r` gives the same result.

The space before first column is given by the `\tabskip1` and the space after last column is equal to `\tabskipr`. Default values are 0pt.

Use nonzero `\tabskip` only in special applications. If `\tabskip` is nonzero then horizontal lines generated by `\crli`, `\crlli` and `\crlp` have another behavior than you probably expected: they are interrupted in each `\tabskip` space.

### 2.29.3 Tables to given width

There are two possibilities how to create tables to given width:

- `\table to\langle size \rangle{\langle declaration \rangle}{\langle data \rangle}` uses stretchability or shrinkability of all spaces between columns generated by `\tabskip` value and eventually by `\tabskip1`, `\tabskipr` values. See example below.
- `\table pxt\langle size \rangle{\langle declaration \rangle}{\langle data \rangle}` expands the columns declared by `p{\langle size \rangle}`, if the (`\langle size \rangle`) is given by a virtual `\tsize` unit. See example below.

Example of `\table to\langle size \rangle`:

```
\thistable{\tabskip=0pt plus1fil minus1fil}
\table to\hsize {lr}{\langle data \rangle}
```

This table has its width `\hsize`. First column starts at the left boundary of this table and it is justified left (to the boundary). Second column ends at the right boundary of the table and it is justified right (to the boundary). The space between them are stretchable and shrinkable in order to reach given width `\hsize`.

Example of `\table pxt\langle size \rangle` (means “paragraphs expanded to”):

```
\table pxt\hsize {lc|p{\tsize}|}{\crl
aaa          & Ddkas jd dsjds ds cgha sfgs dd fddzf dfhz xxz
              dras ffg hksd kds d sdjds h sd jd dsjds ds cgha
              sfgs dd fddzf dfhz xxz. \crl
bb ddd ggg   & Dsjds ds cgha sfgs dd fddzf dfhz xxz
              ddkas jd dsjds ds cgha sfgs dd fddzf. \crl }
```

aaa	Ddkas jd dsjds ds cgha sfgs dd fddzf dfhz xxz dras ffg hksd kds d sdjds h sd jd dsjds ds cgha sfgs dd fddzf dfhz xxz.
bb ddd ggg	Dsjds ds cgha sfgs dd fddzf dfhz xxz ddkas jd dsjds ds cgha sfgs dd fddzf.

The first `c` column is variable width (it gets the width of most wide item) and the resting space to given `\hsize` is filled by the `p` column.

You can declare more than one `p{\langle coefficient \rangle \tsize}` columns in the table when `pxto` keyword is used. The total sum of (`\langle coefficient \rangle`) must be exactly one. For example,

```
\table pxt\13cm {r p{.3\tsize} p{.5\tsize} p{.2\tsize} l}{\langle data \rangle}
```

This gives the ratio of widths of individual paragraphs in the table.

### 2.29.4 Implementation of the `\table` macro

```
3 \_codedecl \table {Basic macros for OpTeX <2020-05-20>} % preloaded in format
```

table.opm

The result of the `\table{\langle declaration \rangle}{\langle data \rangle}` macro is inserted into `\_tablebox`. You can change default value if you want by `\let\_tablebox=\vtop` or `\let\_tablebox=\relax`.

```
11 \let\tablebox=\vbox
```

We save the `to<size>` or `pcto<size>` to #1 and `\tableW` sets the `to<size>` to the `\tablew` macro. If `pcto<size>` is used then `\tablew` is empty and `\tmpdim` includes given `<size>`. The `\ifpcto` returns true in this case.

The `\table` continues by reading `{<declaration>}` in the `\tableA` macro. Catcodes (for example the `|` character) have to be normal when reading `\table` parameters. This is the reason why we use `\catcodetable` here.

table.opm

```
24 \newif\ifpcto
25 \def\table#1{\tablebox\bgroup \tableW#1\empty\end
26   \bgroup \catcodetable\optexcatcodes \tableA}
27 \def\tableW#1#2\end{\pctofalse
28   \ifx#1\empty \def\tablew{}\else
29   \ifx#1p \def\tablew{\tableWx#2\end \else \def\tablew{#1#2}\fi\fi}
30 \def\tableWx xto#1\end{\tmpdim=#1\relax \pctotrue}
31 \public \table ;
```

The `\tablinespace` is implemented by enlarging given `\tabstrut` by desired dimension (height and depth too) and by setting `\lineskip=-2\tablinespace`. Normal table rows (where no `\hrule` is between them) have normal baseline distance.

The `\tableA{<declaration>}` macro scans the `<declaration>` by `\scantabdata#1\relax` and continues by reading `{<data>}` by the `\tableB` macro.

table.opm

```
43 \def\tableA#1{\egroup
44   \the\thistable \global\thistable={}%
45   \ea\ifx\ea\the\tabstrut\setbox\tstrutbox=\null
46   \else \setbox\tstrutbox=\hbox{\the\tabstrut}%
47     \setbox\tstrutbox=\hbox{\vrule widthOpt
48       height\dimexpr\ht\tstrutbox+\tablinespace
49       depth\dimexpr\dp\tstrutbox+\tablinespace}%
50     \offinterlineskip
51     \lineskip=-2\tablinespace
52   \fi
53   \colnum=0 \let\addtabitem=\addtabitemx
54   \def\tmpa{\tabdata={\colnum1\relax}\scantabdata#1\relax
55   \the\everytable \tableB
56 }
```

The `\tableB{<data>}` saves `<data>` to `\tmpb` and does four `\replstrings` to prefix each macro `\crli` (etc.) by `\crrc`. The reason is: we want to use macros which scan its parameter to the delimiter written in right part of table item declaration. See `\fs` for example. The `\crrc` cannot be hidden in other macro in such case.

The `\tabskip` value is saved for places between columns into the `\tabskipmid` macro. Then it runs

```
\tabskip=\tabskipl \halign{<converted declaration>\tabskip=\tabskipr \cr <data>\crrc}
```

This sets the desired boundary values of `\tabskip`. The “between-columns” values are set as `\tabskip=\tabskipmid` in the `<converted declaration>` immediately after each column declarator.

If `pcto` keyword was used, then we set the virtual unit `\tsize` to `\hsize` first. Then the first attempt of the table is created in box 0. Then the `\tsize` is re-calculated using `\wd0` and the real table is printed by `\halign` in the second pass.

If no `pcto` keyword was used, then we print the table using `\halign` directly. The `\tablew` macro is nonempty if the `to` keyword was used.

Because the color selector with `\aftergroup` can be used inside the table item, we must to create second real group for each table item. This is reason why we start `<converted declaration>` by `\bgroup` and we end it by `\egroup` in the `\tableC` macro. Each `&` character is stored as `\egroup&\bgroup` in `<converted declaration>`. The `\halign\tablew \tableC` really does:

```
\halign\tablew{\bgroup<converted declaration>\egroup\tabskip=\tabskipr \cr<data>\crrc}
```

table.opm

```
94 \def\tableB#1{\def\tmpb{#1}%
95   \replstring\tmpb{\crli}{\crrc\crli}\replstring\tmpb{\crll}{\crrc\crll}%
96   \replstring\tmpb{\crlli}{\crrc\crlli}\replstring\tmpb{\crllp}{\crrc\crllp}%
```

```

97 \edef\tabskipmid{\the\tabskip}\tabskip=\tabskipl
98 \ifpcto
99 \tsize=\hsize \setbox0 = \vbox{\halign \tableC}%
100 \tsize=\dimexpr\hsize-(\wd0-\tmpdim)\relax
101 \setbox0=\null \halign \tableC
102 \else
103 \halign\tablew \tableC
104 \fi \egroup
105 }
106 \def\tableC{\ea{\ea\bgroup\the\tabdata\egroup\tabskip=\tabskipr\cr\tmpb\crcr}}
107
108 \newbox\tstrutbox % strut used in table rows
109 \newtoks\tabdata % the \halign declaration line

```

The `\scantabdata` macro converts `\table`'s *<declaration>* to `\halign` *<converted declaration>*. The result is stored into `\tabdata` tokens list. For example, the following result is generated when *<declaration>*=`|cr||c1|`.

```

tabdata: \vrule\the\tabiteml\hfil#\unsskip\hfil\the\tabitemr\tabstrutA
&\the\tabiteml\hfil#\unsskip\the\tabitemr
\vrule\kern\vvkern\vrule\tabstrutA
&\the\tabiteml\hfil#\unsskip\hfil\the\tabitemr\tabstrutA
&\the\tabiteml#\unsskip\hfil\the\tabitemr\vrule\tabstrutA
ddlinedata: &\dditem &\dditem\vvittem &\dditem &\dditem

```

The second result in the `\ddlinedata` macro is a template of one row of the table used by `\crli` macro.

```

129 \def\scantabdata#1{\let\next=\scantabdata
130 \ifx\relax#1\let\next=\relax
131 \else\ifx|#1\addtabvrule
132 \else\ifx|#1\def\next{\scantabdataE}%
133 \else\ifx|#1\def\next{\scantabdataF}%
134 \else\isinlist{123456789}#1\iftrue \def\next{\scantabdataC#1}%
135 \else \ea\ifx\csname _tabdeclare#1\endcsname \relax
136 \ea\ifx\csname _paramtabdeclare#1\endcsname \relax
137 \opwarning{tab-declarator "#1" unknown, ignored}%
138 \else
139 \def\next{\ea\scantabdataB\csname _paramtabdeclare#1\endcsname}\fi
140 \else \def\next{\ea\scantabdataA\csname _tabdeclare#1\endcsname}%
141 \fi\fi\fi\fi\fi\fi \next
142 }
143 \def\scantabdataA#1{\addtabitem
144 \ea\addtabdata\ea{#1\tabstrutA \tabskip\tabskipmid}\scantabdata}
145 \def\scantabdataB#1#2{\addtabitem
146 \ea\addtabdata\ea{#1#2\tabstrutA \tabskip\tabskipmid}\scantabdata}
147 \def\scantabdataC {\def\tmpb{}\afterassignment\scantabdataD \tmpnum=}
148 \def\scantabdataD#1{\loop \ifnum\tmpnum>0 \advance\tmpnum by-1 \addto\tmpb{#1}\repeat
149 \ea\scantabdata\tmpb}
150 \def\scantabdataE#1{\addtabdata{#1}\scantabdata}
151 \def\scantabdataF {\addtabitem\def\addtabitem{\let\addtabitem=\addtabitemx}\scantabdata}

```

The `\addtabitemx` adds the boundary code (used between columns) to the *<converted declaration>*. This code is `\egroup &\bgroup \colnum=<value>\relax`. You can get the current number of column from the `\colnum` register, but you cannot write `\the\colnum` as the first object in a *<data>* item because `\halign` first expands the front of the item and the left part of the declaration is processed after this. Use `\relax\the\colnum` instead. Or you can write:

```

\def\showcolnum{\ea\def\ea\totcolnum\ea{\the\colnum}\the\colnum/\totcolnum}
\table{ccc}{\showcolnum & \showcolnum & \showcolnum}

```

This example prints  $1/3$   $2/3$   $3/3$ , because the value of the `\colnum` is equal to the total number of columns before left part of the column declaration is processed.

```

171 \newcount\colnum % number of current column in the table
172 \public \colnum ;
173
174 \def\addtabitemx{\ifnum\colnum>0
175 \addtabdata{\egroup &\bgroup}\addto\ddlinedata{&\dditem}\fi

```

```

176 \advance\colnum by1 \let\tmpa=\relax
177 \ifnum\colnum>1 \ea\addtabdata\ea{\ea\colnum\the\colnum\relax}\fi}
178 \def\addtabdata#1{\tabdata\ea{\the\tabdata#1}}

```

This code converts `||` or `|` from `\table` *<declaration>* to the *<converted declaration>*.

table.opm

```

184 \def\addtabvrule{%
185 \ifx\tmpa\vrule \addtabdata{\_kern\_vvkern}%
186 \ifnum\colnum=0 \addto\_vvleft{\_vvitem}\else\addto\_ddlinedata{\_vvitem}\fi
187 \else \ifnum\colnum=0 \addto\_vvleft{\_vvitemA}\else\addto\_ddlinedata{\_vvitemA}\fi\fi
188 \let\tmpa=\vrule \addtabdata{\_vrule}%
189 }
190 \def\tabstrutA{\_copy\_tstrutbox}
191 \def\_vvleft{}
192 \def\_ddlinedata{}

```

The default “declaration letters” `c`, `l`, `r` and `p` are declared by setting `\tabdeclarec`, `\tabdeclarel`, `\tabdeclarer` and `\paramtabdeclarep` macros. In general, define `\def\_tabdeclare<letter>{...}` for a non-parametric letter and `\def\_paramtabdeclare<letter>{...}` for a letter with a parameter. The double hash `##` must be in the definition, it is replaced by a real table item data. You can declare more such “declaration letters” if you want.

table.opm

```

204 \def\_tabdeclarec{\_the\_tabiteml\_hfil##\_unsskip\_hfil\_the\_tabitemr}
205 \def\_tabdeclarel{\_the\_tabiteml\_relax##\_unsskip\_hfil\_the\_tabitemr}
206 \def\_tabdeclarer{\_the\_tabiteml\_hfil##\_unsskip\_the\_tabitemr}
207 \def\_paramtabdeclarep#1{\_the\_tabiteml
208 \vtop{\_hsize=#1\_relax \_baselineskip=\_normalbaselineskip
209 \_lineskiplimit=0pt \_noindent##\_unsskip
210 \_ifvmode\_vskip\_dp\_tstrutbox \_else\_lower\_dp\_tstrutbox\_hbox{}\_fi}\_the\_tabitemr}

```

User puts optional spaces around the table item typically, i.e. he/she writes `& text &` instead of `&text&`. The left space is ignored by internal  $\TeX$  algorithm but the right space must be removed by macros. This is a reason why we recommend to use `\_unsskip` after each `##` in your definition of “declaration letters”. This macro isn’t only the primitive `\unskip` because we allow usage of plain  $\TeX$  `\hideskip` macro: `&\hideskip text\hideskip&`.

table.opm

```

221 \def\_unsskip{\_ifmode\_else\_ifdim\_lastskip>0pt \unskip\_fi\_fi}

```

The `\fL`, `\fR`, `\fC` and `\fX` macros only does a special parameters settings for paragraph building algorithm. The `\fS` prints the paragraph into box 0 first, measures the number of lines by the `\prevgraf` primitive and use (or don’t use) `\hfil` (for centering) before the first line.

table.opm

```

230 \let\_fL=\_raggedright
231 \def\_fR{\_leftskip=0pt plus 1fill \_relax}
232 \def\_fC{\_leftskip=0pt plus 1fill \_rightskip=0pt plus 1fill \_relax}
233 \def\_fX{\_leftskip=0pt plus 1fil \_rightskip=0pt plus 1fil \_parfillskip=0pt plus 2fil \_relax}
234 \long\_def\_fS #1\_unsskip{\_noindent \_setbox0 =\_vbox{\_noindent #1\_endgraf \_ea}%
235 \_ifnum\_prevgraf=1 \_hfil \_fi #1\_unsskip
236 }
237 \_public \fL \fR \fC \fX \fS ;

```

The family of `\_cr*` macros `\crl`, `\crl1`, `\crli`, `\crl1i`, `\crlp` and `\tskip` *<dimen>* is implemented here. The `\_zerotabrule` is used in order to suppress the negative `\lineskip` declared by `\tablinespace`.

table.opm

```

247 \def\_crl{\_crcr\_noalign{\_hrule}}
248 \def\_crl1{\_crcr\_noalign{\_hrule\_kern\_hhkern\_hrule}}
249 \def\_zerotabrule {\_noalign{\_hrule height0pt width0pt depth0pt}}
250
251 \def\_crli{\_crcr \_zerotabrule \_omit
252 \_gdef\_dditem{\_omit\_tablinefil}\_gdef\_vvitem{\_kern\_vvkern\_vrule}\_gdef\_vvitemA{\_vrule}%
253 \_vvleft\_tablinefil\_ddlinedata\_crcr \_zerotabrule}
254 \def\_crl1i{\_crli\_noalign{\_kern\_hhkern}\_crli}
255 \def\_tablinefil{\_leaders\_hrule\_hfil}
256
257 \def\_crlp#1{\_crcr \_zerotabrule \_noalign{\_kern-\_drulewidth}%
258 \_omit \_xdef\_crlplist{#1}\_xdef\_crlplist{\_expandafter}\_expandafter\_crlpA\_crlplist,\_end,%
259 \_global\_tmpnum=0 \_gdef\_dditem{\_omit\_crlpD}%
260 \_gdef\_vvitem{\_kern\_vvkern\_kern\_drulewidth}\_gdef\_vvitemA{\_kern\_drulewidth}%

```

```

261 \vvleft\crlpD\ddlinedata \global\tmpnum=0 \crrc \zerotabrule}
262 \def\crlpA#1,{\ifx\end#1\else \crlpB#1-\end,\expandafter\crlpA\fi}
263 \def\crlpB#1#2-#3,{\ifx\end#3\edef\crlplist{\crlplist#1#2,}\else\crlpC#1#2-#3,\fi}
264 \def\crlpC#1-#2-#3,{\tmpnum=#1\relax
265 \loop \xdef\crlplist{\crlplist\the\tmpnum,}\ifnum\tmpnum<#2\advance\tmpnum by1 \repeat}
266 \def\crlpD{\global\advance\tmpnum by1
267 \edef\tmpa{\noexpand\isinlist\noexpand\crlplist{,\the\tmpnum,}}%
268 \tmpa\iftrue \kern-\drulewidth \tablinefil \kern-\drulewidth\else\hfil \fi}
269
270 \def\tskip{\afterassignment\tskipA \tmpdim}
271 \def\tskipA[\gdef\dditem{\gdef\vvitem{\gdef\tabstrutA{}%
272 \vbox to\tmpdim{\ddlinedata \crrc
273 \zerotabrule \noalign{\gdef\tabstrutA{\copy\tstrutbox}}}}
274
275 \public \crl \crl1 \crli \crl1i \crlp \tskip ;

```

The `\mspan{<declaration>}{<text>}` macro generates similar `\omit\span\omit\span` sequence as plain  $\TeX$  macro `\multispan`. Moreover, it uses `\scantabdata` to convert `<declaration>` from `\table` syntax to `\halign` syntax.

```

283 \def\mspan{\omit \tabdata={\tabstrutA}\let\tmpa=\relax \afterassignment\mspanA \mscount=}
284 \def\mspanA[#1]#2{\loop \ifnum\mscount>1 \cs{span}\omit \advance\mscount-1 \repeat
285 \count1=\colnum \colnum=0 \def\tmpa{\tabdata={}\scantabdata#1\relax
286 \colnum=\count1 \setbox0=\vbox{\halign\ea{\ea\bgroupthe\tabdata\egroup\cr#2\cr}%
287 \global\setbox8=\lastbox}%
288 \setbox0=\hbox{\unhbox8 \unskip \global\setbox8=\lastbox}%
289 \unhbox8 \ignorespaces}
290 \public \mspan ;

```

The `\vspan{<number>}{<text>}` implementation is here. We need to lower the box by

$$(\langle number \rangle - 1) * (\text{ht} + \text{dp of } \text{tabstrut}) / 2.$$

The `#1` parameter must be one-digit number. If you want to set more digits then use braces.

```

302 \def\vspan#1#2{\vtop to Opt{\hbox{\lower \dimexpr
303 #1\dimexpr(\ht\tstrutbox+\dp\tstrutbox)/2\relax
304 -\dimexpr(\ht\tstrutbox+\dp\tstrutbox)/2\relax \hbox{#2}}\vss}}
305 \public \vspan ;

```

The parameters of primitive `\vrule` and `\hrule` keeps the rule “last wins”. If we re-define `\hrule` to `\orihrule height1pt` then each usage of redefined `\hrule` uses 1pt height if this parameter isn’t overwritten by another following `height` parameter. This principle is used for settings another default rule thickness than 0.4pt by the macro `\rulewidth`.

```

316 \newdimen\drulewidth \drulewidth=0.4pt
317 \let\orihrule=\hrule \let\orivrule=\vrule
318 \def\rulewidth{\afterassignment\rulewidthA \drulewidth}
319 \def\rulewidthA{\edef\hrule{\orihrule height\drulewidth}%
320 \edef\vrule{\orivrule width\drulewidth}%
321 \let\rulewidth=\drulewidth
322 \public \vrule \hrule \rulewidth;}
323 \public \rulewidth ;

```

The `\frame{<text>}` uses “`\vbox in \vtop`” trick in order to keep the baseline of the internal text at the same level as outer baseline. User can write `\frame{abcxyz}` in normal paragraph line, for example and gets the expected result: abcxyz. The internal margins are set by `\vbkern` and `\hhkern` parameters.

```

333 \long\def\frame#1{%
334 \hbox{\vrule\vtop{\vbox{\hrule\kern\vbkern
335 \hbox{\kern\hhkern\relax#1\kern\hhkern}%
336 }\kern\vbkern\hrule}\vrule}}
337 \public \frame ;

```

## 2.30 Balanced multi-columns

multicolumns.opm

```
3 \_codedecl \begmulti {Balanced columns <2020-03-26>} % preloaded in format
```

This code is documented in detail in the “TeXbook naruby”, pages 244–246, free available, <http://petr.olsak.net/tbn.html>, but in Czech. Roughly speaking, macros complete all material between `\begmulti{num-columns}` and `\endmulti` into one `\vbox 6`. Then the macro measures the amount of free space at the current page using `\pagegoal` and `\pagetotal` and does `\vsplit` of `\vbox 6` to columns with height of such free space. This is done only if we have enough amount of material in `\vbox 6` to fill full page by columns. This is repeated in loop until we have less amount of material in `\vbox 6`. Then we run `\_balancecolumns` which balances the last part of columns. Each part of printed material is distributed to main vertical list as `\hbox{<columns>}` and we need not do any change in the output routine.

If you have paragraphs in `\begmulti... \endmulti` environment then you may say `\raggedright` inside this environment and you can re-assign `\widowpenalty` and `\clubppenalty` (they are set to 10000 in OpTeX).

multicolumns.opm

```
24 \_def\_multiskip{\_medskip} % space above and below \begmulti...\endmulti
25
26 \_newcount\_mullines
27
28 \_def\_begmulti #1 {\_par\_bgroup\_wipeepar\_multiskip\_penalty0 \_def\_Ncols{#1}
29 \_setbox6=\vbox\_bgroup\_penalty0
30 %% \hspace := column width = (\hspace+\colsep) / n - \colsep
31 \_advance\_hspace by\_colsep
32 \_divide\_hspace by\_Ncols \_advance\_hspace by\_colsep
33 \_mullines=0
34 \_def\par{\_ifhmode\_endgraf\_global\_advance\_mullines by\_prevgraf\_fi}%
35 }
36 \_def\_endmulti{\_vskip-\_prevdepth\_vfil
37 \_ea\_egroup\_ea\_baselineskip\_the\_baselineskip\_relax
38 \_dimen0=.8\_maxdimen \_tmpnum=\_dimen0 \_divide\_tmpnum by\_baselineskip
39 \_splittopskip=\_baselineskip
40 \_setbox1=\_vsplit6 to0pt
41 %% \dimen1 := the free space on the page
42 \_ifdim\_pagegoal=\_maxdimen \_dimen1=\_vsize \_corrsize{\_dimen1}
43 \_else \_dimen1=\_pagegoal \_advance\_dimen1 by-\_pagetotal \_fi
44 \_ifdim \_dimen1<2\_baselineskip
45 \_vfil\_break \_dimen1=\_vsize \_corrsize{\_dimen1} \_fi
46 \_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
47 \_divide\_dimen0 by\_Ncols \_relax
48 %% split the material to more pages?
49 \_ifdim \_dimen0>\_dimen1 \_splitpart
50 \_else \_balancecolumns \_fi % only balancing
51 \_multiskip\_egroup
52 }
```

Splitting columns...

multicolumns.opm

```
58 \_def\_makecolumns{\_bgroup % full page, destination height: \dimen1
59 \_vbadness=20000 \_setbox1=\_hbox{\_tmpnum=0
60 \_loop \_ifnum\_Ncols>\_tmpnum
61 \_advance\_tmpnum by1
62 \_setbox1=\_hbox{\_unhbox1 \_vsplit6 to\_dimen1 \_hss}
63 \_repeat
64 \_hbox{\_nobreak\_vskip-\_splittopskip \_nointerlineskip
65 \_line{\_unhbox1\_unskip}
66 \_dimen0=\_dimen1 \_divide\_dimen0 by\_baselineskip \_multiply\_dimen0 by\_Ncols
67 \_global\_advance\_mullines by-\_dimen0
68 \_egroup
69 }
70 \_def\_splitpart{%
71 \_makecolumns % full page
72 \_vskip 0pt plus 1fil minus\_baselineskip \_break
73 \_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
74 \_divide\_dimen0 by\_Ncols \_relax
75 \_ifx\_balancecolumns\_flushcolumns \_advance\_dimen0 by-.5\_vsize \_fi
```



```

76 \_dimen1=\_vsize \_corrsize{\_dimen1}\_dimen2=\_dimen1
77 \_advance\_dimen2 by-\_baselineskip
78 %% split the material to more pages?
79 \_ifvoid6 \_else
80 \_ifdim \_dimen0>\_dimen2 \_ea\_ea\_ea \_splitpart
81 \_else \_balancecolumns % last balancing
82 \_fi \_fi
83 }

```

Final balancing of the columns.

multicolumns.opm

```

89 \_def\_balancecolumns{\_bgroup \_setbox7=\_copy6 % destination height: \_dimen0
90 \_ifdim\_dimen0>\_baselineskip \_else \_dimen0=\_baselineskip \_fi
91 \_vbadness=20000
92 \_def\_tmp{%
93 \_setbox1=\_hbox{\_tmpnum=0
94 \_loop \_ifnum\_Ncols>\_tmpnum
95 \_advance\_tmpnum by1
96 \_setbox1=\_hbox{\_unhbox1
97 \_ifvoid6 \_hbox to\_wd6{\_hss}\_else \_vsplit6 to\_dimen0 \_fi\_hss}
98 \_repeat
99 \_ifvoid6 \_else
100 \_advance \_dimen0 by.2\_baselineskip
101 \_setbox6=\_copy7
102 \_ea \_tmp \_fi}\_tmp
103 \_hbox{\_nobreak\_vskip-\_splittopskip \_nointerlineskip
104 \_hbox to\_hsize{\_unhbox1\_unskip}%
105 \_egroup
106 }
107 \_def\_corrsize #1{%% #1 := #1 + \_splittopskip - \_topskip
108 \_advance #1 by \_splittopskip \_advance #1 by-\_topskip
109 }
110 \_public \_begmulti \_endmulti ;

```

## 2.31 Citations, bibliography

### 2.31.1 Macros for citations and bibliography preloaded in the format

cite-bib.opm

```

3 \_codedecl \_cite {Cite, Bibliography <2020-03-09>} % loaded in format

```

Registers used by `\cite`, `\bib` macros are declared here. The `\bibnum` counts the bibliography items from one. The `\bibmark` is used when `\nonumcitations` is set.

cite-bib.opm

```

11 \_newcount\_bibnum % the bibitem counter
12 \_newtoks\_bibmark % the bibmark used if \nonumcitations
13 \_newcount\_lastcitenum \_lastcitenum=0 % for \shortcitations
14 \_public \_bibnum \_bibmark ;

```

`\cite` [*<label>*,*<label>*,...,*<label>*] manages *<labes>* using `\_citeA` and prints [*<bib-marks>*] using `\_printsavedcites`.

`\nocite` [*<label>*,*<label>*,...,*<label>*] only manages *<labels>* but prints nothing.

`\rcite` [*<label>*,*<label>*,...,*<label>*] behaves like `\cite` but prints *<bib-marks>* without brackets.

`\ecite` [*<label>*]{*<text>*} behaves like `\rcite` [*<label>*] but prints *<text>* instead *<bib-mark>*. The *<text>* is hyperlinked like *<bib-marks>* when `\cite` or `\rcite` is used. The empty internal macro `\_savedcites` will include the *<bib-marks>* list to be printed. This list is set by `\_citeA` inside group and it is used by `\_printsavedcites` in the same group. Each `\cite`/`\rcite`/`\ecite` macro starts from empty list of *<bib-marks>* because new group is opened.

cite-bib.opm

```

34 \_def\_cite[#1]{\_citeA#1,,,\_printsavedcites}}
35 \_def\_nocite[#1]{\_citeA#1,,,\_printsavedcites}}
36 \_def\_rcite[#1]{\_citeA#1,,,\_printsavedcites}}
37 \_def\_ecite[#1]{\_bgroup\_citeA#1,,,\_ea\_eciteB\_savedcites;}
38 \_def\_eciteB#1,#2;#3{\_if?#1\_relax #3\_else \_ilink[cite:#1]{#3}\_fi\_egroup}
39 \_def\_savedcites{}
40
41 \_public \_cite \_nocite \_rcite \_ecite ;

```

$\langle bib\text{-}marks \rangle$  may be numbers or a special text related to cited bib-entry. It depends on `\nonumcitations` and on used bib-style. The mapping from  $\langle label \rangle$  to  $\langle bib\text{-}mark \rangle$  is done when `\bib` or `\usebib` is processed. These macros store the information to `\_Xbib{\langle label \rangle}{\langle number \rangle}{\langle nonumber \rangle}` where  $\langle number \rangle$  and  $\langle nonumber \rangle$  are two variants of  $\langle bib\text{-}mark \rangle$  (numbered or text-like). This information is read from `.ref` file and it is saved to macros `\_bib:\langle label \rangle` and `\_bibm:\langle number \rangle`. First one includes number and second one includes  $\langle nonumber \rangle$ . The `\_lastbibnum` macro includes last number of bib-entry used in the document. A designer can use it to set appropriate indentation when printing the list of all bib-entries.

cite-bib.opm

```
57 \_def\_Xbib#1#2#3{\_sdef\_bib:#1}{\_bibnn{#2}&}%
58 \_if^#3\_else\_sdef\_bim:#2}{#3}\_fi\_def\_lastbibnum{#2}
```

`\_citeA`  $\langle label \rangle$ , processes one label from list of labels given in the parameter of `\cite`, `\nocite`, `\rcite` or `\ecite` macros. It adds the  $\langle label \rangle$  to global list `\_citelist` which will be used by `\usebib` (it must to know what  $\langle labels \rangle$  are used in the document in order to pick-up only relevant bib-entries from the database. Because we want to save space and not to save the same  $\langle label \rangle$  to `\_citelist` twice, we distinguish four cases:

- $\langle label \rangle$  was not declared by `\_Xbib` and it is first such  $\langle label \rangle$  in the document: Then `\_bib:\langle label \rangle` is undefined and we save label using `\_addcitlist`, write warning on the terminal and define `\_bib:\langle label \rangle` as empty.
- $\langle label \rangle$  was not declared by `\_Xbib` but it was used previously in the document: Then `\_bib:\langle label \rangle` is empty and we do nothing (only data to `\_savedcites` are saved).
- $\langle label \rangle$  was declared by `\_Xbib` and it is first such  $\langle label \rangle$  in the document: Then `\_bin:\langle label \rangle` includes `\_bibnn{\langle number \rangle}&` and we test this case by `\_if &\_bibnn{\langle number \rangle}&`. This is true when `\_bibnn{\langle number \rangle}` expands to empty. The  $\langle label \rangle$  is saved by `\_addcitelist` and `\_bib:\langle label \rangle` is re-defined directly as  $\langle number \rangle$ .
- $\langle label \rangle$  was declared by `\_Xbib` and it was used previously in the document. Then we do nothing (only data to `\_savedcites` are saved).

The `\_citeA` macro runs repeatedly over whole list of  $\langle labels \rangle$ .

cite-bib.opm

```
87 \_def\_citeA #1#2,{\_if#1,\_else
88 \_if *#1\_addcitelist{*}\_ea\_skiptorelax \_fi
89 \_ifcsname\_bib:#1#2\_endcsname \_else
90 \_addcitelist{#1#2}%
91 \_opwarning{The cite [#1#2] unknown. Try to TeX me again}\_openref
92 \_addto\_savedcites{?,}\_def\_sortcitesA{\\_lastcitenum=0
93 \_ea\_gdef \_csname\_bib:#1#2\_endcsname {}%
94 \_ea\_skiptorelax \_fi
95 \_ea\_ifx \_csname\_bib:#1#2\_endcsname \_empty
96 \_addto\_savedcites{?,}\_def\_sortcitesA{\\_lastcitenum=0
97 \_ea\_skiptorelax \_fi
98 \_def\_bibnn#1{}}%
99 \_if &\_csname\_bib:#1#2\_endcsname
100 \_def\_bibnn#1#2{##1}%
101 \_addcitelist{#1#2}%
102 \_sdef\_bib:#1#2}{\_csname\_bib:#1#2\_endcsname}%
103 \_fi
104 \_edef\_savedcites{\_savedcites \_csname\_bib:#1#2\_endcsname,}%
105 \_relax
106 \_ea\_citeA\_fi
107 }
108 \_def\_addcitelist#1{\_global\_addto\_citelist{\_citeI[#1]}}
109 \_def\_citelist{}
```

The  $\langle bib\text{-}marks \rangle$  (in numeric or text form) are saved in `\_savedcites` macro separated by commas. The `\_printsavedcites` prints them by normal order or sorted if `\sortcitations` is specified or condensed if `\shordcitations` is specified.

The `\sortcitations` appends the dummy number 300000 and we suppose that normal numbers of bib-entries are less than this constant. This constant is removed after sorting algorithm. The `\shordcitations` sets simply `\_lastcitenum=1`. The macros for  $\langle bib\text{-}marks \rangle$  printing follows (sorry, without detail documentation). They are documented in `opmac-d.pdf` (but only in Czech).

cite-bib.opm

```
125 \_def\_printsavedcites{\_sortcitesA
126 \_chardef\_tmpb=0 \_ea\_citeB\_savedcites,%
```

```

127 \_ifnum\_tmpb>0 \_printdashcite{\_the\_tmpb}\_fi
128 }
129 \_def\_sortcitesA{}
130 \_def\_sortcitations{%
131 \_def\_sortcitesA{\_edef\_savedcites{300000,\_ea}\_ea\_sortcitesB\_savedcites,%
132 \_def\_tmpa###1300000,{\_def\_savedcites{###1}}\_ea\_tmpa\_savedcites}%
133 }
134 \_def\_sortcitesB #1,{\_if $1$%
135 \_else
136 \_mathchardef\_tmpa=#1
137 \_edef\_savedcites{\_ea}\_ea\_sortcitesC \_savedcites\_end
138 \_ea\_sortcitesB
139 \_fi
140 }
141 \_def\_sortcitesC#1,{\_ifnum\_tmpa<#1\_edef\_tmpa{\_the\_tmpa,#1}\_ea\_sortcitesD
142 \_else\_edef\_savedcites{\_savedcites#1,}\_ea\_sortcitesC\_fi}
143 \_def\_sortcitesD#1\_end{\_edef\_savedcites{\_savedcites\_tmpa,#1}}
144
145 \_def\_citeB#1,{\_if$1$\_else
146 \_if?#1\_relax??%
147 \_else
148 \_ifnum\_lastcitenum=0 % only comma separated list
149 \_printcite{#1}%
150 \_else
151 \_ifx\_citesep\_empty % first cite item
152 \_lastcitenum=#1\_relax
153 \_printcite{#1}%
154 \_else % next cite item
155 \_advance\_lastcitenum by1
156 \_ifnum\_lastcitenum=#1\_relax % cosecutive cite item
157 \_mathchardef\_tmpb=\_lastcitenum
158 \_else % there is a gap between cite items
159 \_lastcitenum=#1\_relax
160 \_ifnum\_tmpb=0 % previous items were printed
161 \_printcite{#1}%
162 \_else
163 \_printdashcite{\_the\_tmpb}\_printcite{#1}\_chardef\_tmpb=0
164 \_fi\_fi\_fi\_fi\_fi
165 \_ea\_citeB\_fi
166 }
167 \_def\_shortcitations{\_lastcitenum=1 }
168
169 \_def\_printcite#1{\_citesep\_ilink[cite:#1]{\_citelinkA{#1}}\_def\_citesep{,\_hskip.2em\_relax}}
170 \_def\_printdashcite#1{\_ifmode-\_else\_hbox{--}\_fi\_ilink[cite:#1]{\_citelinkA{#1}}}%
171 \_def\_citesep{}
172
173 \_def\_nonumcitations{\_lastcitenum=0\_def\_sortcitesA{}\_def\_etalchar##1{$^{##1}$}%
174 \_def\_citelinkA##1{\_isdefined{bim:##1}\_iftrue \_csname \_bim:##1\_endcsname
175 \_else ##1\_opwarning{\_noexpand\_nonumcitations + empty bibmark. Maybe bad BibTeX style}\_fi}
176 }
177 \_def\_citelinkA{}
178
179 \_public \nonumcitations \sortcitations \shortcitations ;

```

The `\bib [⟨label⟩] {⟨optional bib-mark⟩}` prints one bib-entry without reading any database. The bib-entry follows after this command. This command counts the used `\bibs` from one by `\bibnum` counter and saves `\Xbib{⟨label⟩}{\_the\_bibnum}{\_the\_bibmark}` into `.ref` file immediately using `\wbib`. This is the core of creation of mapping from `⟨labels⟩` to `⟨bib-marks⟩`.

cite-bib.opm

```

190 \_def\_bib[#1]{\_def\_tmp{\_isnextchar={\_bibA[#1]}\_bibmark={}\_bibB[#1]}}%
191 \_ea\_tmp\_romannumeral-`.} % ignore optional space
192 \_def\_bibA[#1]=#2{\_bibmark={#2}\_bibB[#1]}
193 \_def\_bibB[#1]{\_par \_bibskeep
194 \_advance\_bibnum by1
195 \_noindent \_def\_tmpb{#1}\_wbib{#1}{\_the\_bibnum}{\_the\_bibmark}%
196 \_printlabel{#1}%
197 \_printbib \_ignorespaces
198 }
199 \_def\_wbib#1#2#3{\_dest[cite:\_the\_bibnum]%

```

```

200 \_ifx\_wref\_wrefrelax\_else \_immediate\_wref\_Xbib{{#1}{{#2}{{#3}}}\_fi}
201
202 \_public \bib ;

```

The `\_printbib` prints the bib-entry itself. You can re-define it if you want different design. The `\_printbib` starts in horizontal mode after `\noindent` and after the eventual hyperlink destination is inserted. By default, the `\_printbib` sets the indentation by `\hangindent` and prints numeric *<bib-marks>* by `\llap{[\_the\bibnum]}`. If `\nonumcitations` then the `\_citelinkA` is not empty and *<bib-marks>* (`\_the\bibnum` nor `\_the\bibmark`) are not printed. The text of bib-entry follows. User can create this text manually using `\bib` command or it is generated automatically from a `.bib` database by `\usebib` command.

The vertical space between bib-entries is controlled by `\_bibskip` macro.

```

219 \_def \_printbib {\_hangindent=\_iindent
220 \_ifx\_citelinkA\_empty \_hskip\_iindent \_llap{[\_the\bibnum] }\_fi
221 }
222 \_def \_bibskip {\_ifnum\_bibnum>0 \_smallskip \_fi}

```

`cite-bib.opm`

The `\usebib` command is implemented in `usebib.opm` file which is loaded when the `\usebib` command is firstly used. The `usebib.opm` file loads the `librarian.tex` for scanning the `.bib` files. See the section 2.31.2, where the file `usebib.opm` is documented.

```

232 \_def\_usebib{\_par \_opinput {usebib.opm} \_usebib}
233 \_def{usebib}{\_usebib}

```

`cite-bib.opm`

The macros above works if all `\cite` (or similar) commands are used before the `\usebib` command is used because `\usebib` prints only such bib-entries their *<labels>* are saved in the `\_citelist`. But if some `\cite` is used after `\usebib`, then `\usebib` sets `\_addcitelist` to `\_writeXcite`, so such `\cite` saves the information to the `.ref` file in the format `\_Xcite{<label>}`. Such information are copied to `\_citelistB` during reading `.ref` file and `\usebib` concatenates two lists of *<labels>* from `\_citelist` and `\_citelistB` and uses this concatenated list.

```

247 \_def\_Xcite#1{\_addto\_citelistB{\_citeI[#1]}}
248 \_def\_writeXcite#1{\_openref\_immediate\_wref\_Xcite{{#1}}}
249 \_def\_citelistB{}

```

`cite-bib.opm`

## 2.31.2 The `\usebib` command

The file `usebib.opm` implements the command `\usebib/<sorttype> (<style>) <bibfiles>` where *<sorttype>* is one letter `c` (references ordered by citation order in the text) or `s` (references ordered by key in the style file), *<style>* is the part of the name `bib-<style>.opm` of the style file and *<bibfiles>* are one or more `.bib` file names without suffix separated by comma without space. Example:

```
\usebib/s (simple) mybase,yourbase
```

This command reads the *<bibfiles>* directly and creates the list of bibliographics references (only those declared by `\cite[]` or `\nocite[]` in the text). The formatting of such references is defined in the style file. The usage is mentioned in user documentation too.

The principle “first entry wins” is used. Suppose `\usebib/s (simple) local,global`. If an entry with the same label is declared in `local.bib` and in `global.bib` too then the first wins. So, you can set an exceptions in your `local.bib` file for your document.

### Notes for style writers

The `bib-<style>.opm` file must define the commands:

- `\_authorname` ... formatting of one name in the authors list. The macro can use the following data: `\_NameCount` (the number of the currently processed author name in the list), `0\_namecount` (the total number of the authors in the list), `\_Lastname`, `\_Firstname`, `\_Von`, `\_Junior` (the parts of the name). See the documentation of the `librarian` package for more info.
- `\_editorname` ... the same as `\_authorname`, but for editors list.
- `\_print:<entrytype>` (defined by `\_sdef`) for formatting the entry of *<entrytype>*. The *<entrytype>* have to be lowercase. This command can use the command:
- `\_bprinta [<fieldname>] {\_if defined} {\_if not defined}`. The part *<if defined>* is executed if *<fieldname>* is declared in `.bib` file for the entry which is currently processed. Else the part

$\langle if not defined \rangle$  is processed. The part  $\langle if defined \rangle$  can include the `*` parameter which is replaced by the value of the  $\langle fieldname \rangle$ . The part  $\langle if not defined \rangle$  can include the `\_bibwarning` command if the  $\langle fieldname \rangle$  is mandatory.

- `\_bprintb` [ $\langle fieldname \rangle$ ]  $\{ \langle if defined \rangle \} \{ \langle if not defined \rangle \}$ . The same as `\_bprinta`, but the `##1` parameter is used instead `*`. Differences: `##1` parameter can be used more than once and can be enclosed in nested braces. The parameter can be used at most once and cannot be enclosed in braces. Warning: if the `\_bprintb` commands are nested (`\_bprintb` in `\_bprintb`), then you need to write `####1` parameter for internal `\_bprintb`. But if `\_bprinta` commands are nested then the parameter is not duplicated.
- `\_pbprintc` `\macro`  $\{ \langle if non-empty \rangle \}$ . The  $\langle if non-empty \rangle$  part is executed if `\macro` is non-empty. The parameter can be used, it is replaced by the `\macro`.
- `\_bprintv` [ $\langle field1 \rangle, \langle field2 \rangle, \dots$ ]  $\{ \langle if defined \rangle \} \{ \langle if not defined \rangle \}$ . The part  $\langle if defined \rangle$  is executed if  $\langle field1 \rangle$  or  $\langle field2 \rangle$  or ... is defined, else the second part  $\langle if not defined \rangle$  is executed. There is one field name or the list field names separated by commas. The parts cannot include any parameter.

There are two special fieldnames: `!author` and `!editor`. The processed list of authors or editors (by repeatedly calling `\_authorname` or `\_editorname`) are used here instead of raw data.

You can define `_print:BEGIN` and/or `_print:END` which is executed at the begin or end of each  $\langle entrytype \rangle$ . The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by `\_printbib` macro used in `OpTeX` (and may be redefined by the author or document designer).

You can declare `\_bimark={something}` in the `_print:END` macro. This bibmark is saved to the `.ref` file (created by `OpTeX`) and used in the next `TeX` run as `\cite` marks when `\nonumcitations` is set.

The whole style file is read in the group during `\usebib` command is executed before typesetting the reference list. Each definition or setting is local here.

If you are using non-standard fieldnames in `.bib` database and `bib.` style, you has to declare them by `\_CreateField`  $\{ \langle fieldname \rangle \}$ .

You can declare `\_SortingOrder` in the manner documented by `librarian` package.

If your style adds some words or abbreviations you can make them multilingual by saying `\_mtext{<label>}` instead such word and `\_mtdef{<label>}`  $\{ \langle English \rangle \} \{ \langle Czech \rangle \} \{ \langle Slovak \rangle \}$  declaration. The right part is printed by current value of the `\language` register. You can add more languages by re-defining the `\_mtdef` command. See the section 2.36.3 for more information.

If you are using `\nonumcitations`, then the `\_bibmark` tokens register have to be prepared in the style file (in `_print:BEGIN`, `_print:END`, in `\_authorname` etc.) This value will be used in the `\cite[]` places in the document.

The example of the style file is in `bib-simple.opm`.

User or author of the `bib.` style can create the hidden field which has a precedence while sorting names. Example:

```
\CreateField {sortedby}
\SpecialSort {sortedby}
```

Suppose that the `.bib` file includes:

```
...
author    = "Jan Chadima",
sortedby  = "Hzzadima Jan",
...
```

Now, this author is sorted between H and I, because the Ch digraph in this name has to be sorted by this rule.

If you need (for example) to place the autocitations before other citations, then you can mark your entries in `.bib` file by `sortedby = "@"`, because this character is sorted before A.

### 2.31.3 The `usebib.opm` macro file loaded when `\usebib` is used

```
3 \_codedecl \MakeReference {Reading bib databases <2020-03-13>} % loaded on demand by \usebib
```

Loading the `librarian.tex` macro package. See `texdoc librarian` for more information about it.

```

10 \def\tmp{}
11 \let\errmessageori=\errmessage % we needn't \errmessage during \input librarian
12 \def\errmessage#1{\def\tmp{error}}
13 \let\newwriteori=\newwrite % we need not to create \jobname.lbr:
14 \def\newwrite#1{\csname lb@restoreat\endcsname \endinput}
15 \def\tmpb{\catcode\_=12 \input librarian \catcode\_=11 }\tmpb
16 \let\errmessage=\errmessageori
17 \let\newwrite=\newwriteori
18 \ifx\tmp\_empty\_else
19   \def\usebib/#1 (#2) #3 {%
20     \opwarning{eTeX and (pdfTeX or XeTeX or LuaTeX) not detected}%
21     \immediate\write16{\space\space
22       But librarian package needs it. \noexpand\usebib ignored.}%
23   }
24   \endinput \fi
25 \private \BibFile \ReadList \SortList \SortingOrder \NameCount \AbbreviateFirstname
26 \CreateField \RetrieveFieldInFor \RetrieveFieldIn ;

```

The \usebib command.

```

32 \def\usebib/#1 (#2) #3 {%
33   \ifx\_citelist\_empty
34     \opwarning{No cited items. \noexpand\usebib ignored}%
35   \else
36     \bgroup \par
37     \emergencystretch=.3\hsize
38     \ifx\_bibpart\_undefined \def\_bibpart{none}\fi
39     \def\optexbibstyle{#2}%
40     \setctable\optexcatcodes
41     \input bib-#2.opm
42     \the \bibtexhook
43     \let\_citeI=\relax \xdef\_citelist{\_citelist\_citelistB}%
44     \global\let\_addcitelist=\writeXcite
45     \def\tmp##1[*]##2\relax{\_def\tmp{##2}}\expandafter\tmp\_citelist[*]\relax
46     \ifx\tmp\_empty\_else % there was \nocite[*] used.
47       \setbox0=\vbox{\hsize=\maxdimen \def\_citelist{}\_adef@\_readbibentry}%
48       \input #3.bib
49       \expandafter\_expandafter\_def\_expandafter\_citelist\_expandafter{\_citelist}%
50     \fi
51     \def\_citeI[##1]{\_csname lb@cite\endcsname{##1}{\_bibpart}{}}\_citelist
52     \BibFile{#3}%
53     \if s#1\SortList{\_bibpart}\fi
54     \ReadList{\_bibpart}%
55     \restorectable
56   \egroup
57   \fi
58 }
59 \def\_readbibentry#1#{\_readbibentryA}
60 \def\_readbibentryA#1{\_readbibentryB#1,,\_relax!.}
61 \def\_readbibentryB#1#2,#3\_relax!.{\_addto\_citelist{\_citeI[#1#2]}}

```

Corrections in librarian macros.

```

67 \tmpnum=\catcode\@ \catcode\@=11
68 \def\lb@checkmissingentries#1,{% we needn't \errmessage here, only \opmacwarning
69   \def\lb@temp{#1}%
70   \unless\_ifx\lb@temp\lb@eoe
71     \lb@ifcs{#1}{fields}%
72     {}%
73     {\_opwarning{\_string\_usebib: entry [#1] isn't found in .bib file(s)}}%
74   \ea\lb@checkmissingentries
75   \fi
76 }
77 \def\lb@readentry#1#2#3,{% space before key have to be ingnored
78   \def\lb@temp{#2#3}% we need case sensitive keys
79   \def\lb@next{\_ea\lb@gotoat\lb@gobbletoeoe}%
80   \lb@ifcs\lb@temp{requested}%
81     {\_let\lb@entrykey\lb@temp
82       \lb@ifcs\lb@entrykey{fields}{}%
83       {\lb@defcs\lb@entrykey{fields}{}%

```



```

84         \_lowercase{\lb@addfield{entrytype}{#1}}%
85         \_let\lb@next\lb@analyzeentry}}{}%
86     \lb@next
87 }
88 \_let\lb@compareA=\lb@compare
89 \_let\lb@preparesortA=\lb@preparesort
90 \_def\lb@compare#1\lb@eoe#2\lb@eoe{% SpecialSort:
91     \_ifx\lb@sorttype\lb@namestring
92         \_ifx\lb@sortfield\_undefined \lb@compareA#1\lb@eoe#2\lb@eoe
93         \_else
94             \_ea\_RetrieveFieldInFor\_ea{\_sortfield}\lb@entrykey\lb@temp
95             \_ifx\lb@temp\_empty \_toks1={#1\lb@eoe}\_else \_toks1=\_ea{\lb@temp\lb@eoe}\_fi
96             \_ea\_RetrieveFieldInFor\_ea{\_sortfield}\lb@currententry\lb@temp
97             \_ifx\lb@temp\_empty \_toks2={#2\lb@eoe}\_else \_toks2=\_ea{\lb@temp\lb@eoe}\_fi
98             \_edef\lb@temp{\_noexpand\lb@compareA\_space\_the\_toks1 \_space\_the\_toks2}\lb@temp
99         \_fi
100     \_else \lb@compareA#1\lb@eoe#2\lb@eoe \_fi
101 }
102 \_def\lb@preparesort#1#2\lb@eoe{%
103     \_if#1-%
104         \_def\lb@sorttype{#2}%
105         \_else
106             \_def\lb@sorttype{#1#2}%
107         \_fi
108     \lb@preparesortA#1#2\lb@eoe
109 }
110 \_def\_SpecialSort#1{\_def\_sortfield{#1}}
111 \_def\WriteImmediateInfo#1{} % the existence of .lbr file blocks new reading of .bib
112 \_catcode\@=\_tmpnum

```

Main action per every entry.

usebib.opm

```

118 \_def\MakeReference{\_par \_bibskip
119     \_advance\_bibnum by1
120     \_isdefined{\_bim:\_the\_bibnum}\_iftrue
121         \_edef\_tmpb{\_csname \_bim:\_the\_bibnum\_endcsname}%
122         \_bibmark=\_ea{\_tmpb}%
123     \_else \_bibmark={}\_fi
124     \_edef\_tmpb{\_EntryKey}%
125     \_noindent \_dest[cite:\_the\_bibnum]\_printlabel\_EntryKey
126     \_printbib
127     {%
128         \_RetrieveFieldIn{entrytype}\_entrytype
129         \_csname \_print:BEGIN\_endcsname
130         \_isdefined{\_print:\_entrytype}\_iftrue
131             \_csname \_print:\_entrytype\_endcsname
132         \_else
133             \_ifx\_entrytype\_empty \_else
134                 \_opwarning{Entrytype @\_entrytype\_space from [\_EntryKey] undefined}%
135                 \_csname \_print:misc\_endcsname
136             \_fi\_fi
137             \_csname \_print:END\_endcsname
138             \_ifx\_wref\_wrefrelax\_else
139                 \_immediate\_wref\_Xbib{{\_EntryKey}{\_the\_bibnum}{\_the\_bibmark}}\_fi
140         }\_par
141     }

```

The `\_bprinta`, `\_bprintb`, `\_bprintc`, `\_bprintv` commands used in the style files:

usebib.opm

```

148 \_def\_bprinta {\_bprintb*}
149 \_def\_bprintb #1[#2#3]{%
150     \_def\_bibfieldname{#2#3}%
151     \_if!#2\_relax
152         \_def\_bibfieldname{#3}%
153         \_RetrieveFieldIn{#3}\_bibfield
154         \_ifx\_bibfield\_empty\_else
155             \_RetrieveFieldIn{#3number}\_namecount
156             \_def\_bibfield{\_csname \_Read#3\_ea\_endcsname \_csname \_pp:#3\_endcsname}%
157         \_fi
158     \_else

```

```

159 \RetrieveFieldIn{#2#3}\_bibfield
160 \_fi
161 \_if^#1^%
162 \_ifx\_bibfield\_empty\_ea\_ea\_ea\_doemptyfield
163 \_else\_ea\_ea\_ea\_dofullfield\_fi
164 \_else\_ea\_bprintaA
165 \_fi
166 }
167 \_def\_dofullfield#1#2{\_def\_dofield##1{#1}\_ea\_dofield\_ea{\_bibfield}}
168 \_def\_doemptyfield#1#2{\_def\_dofield##1{#2}\_ea\_dofield\_ea{\_bibfield}}
169 \_let\_Readauthor=\ReadAuthor \_let\_Readeditor=\ReadEditor
170 \_def\_bprintaA #1#2{\_ifx\_bibfield\_empty #2\_else\_bprintaB #1*\_eee\_fi}
171 \_def\_bprintaB #1*#2*#3\_eee{\_if^#3^#1\_else\_ea\_bprintaC\_ea{\_bibfield}{#1}{#2}\_fi}
172 \_def\_bprintaC #1#2#3{#2#1#3}
173 \_def\_bprintc#1#2{\_bprintcA#1#2*\_relax}
174 \_def\_bprintcA#1#2*#3*#4\_relax{\_ifx#1\_empty\_else\_if^#4^#2\_else#2#1#3\_fi\_fi}
175 \_def\_bprintv [#1]#2#3{\_def\_tmpa{#2}\_def\_tmpb{#3}\_bprintvA #1,,}
176 \_def\_bprintvA #1,{%
177 \_if^#1^\_tmpb\_else
178 \_RetrieveFieldIn{#1}\_tmp
179 \_ifx\_tmp\_empty
180 \_else\_tmpa\_def\_tmpb{}\_def\_tmpa}%
181 \_fi
182 \_ea\_bprintvA
183 \_fi
184 }
185 \_sdef{\_pp:author}{\_letNames\_authorname}
186 \_sdef{\_pp:editor}{\_letNames\_editorname}
187 \_def\_letNames{\_let\_Firstname=\Firstname \_let\_Lastname=\Lastname
188 \_let\_Von=\Von \_let\_Junior=\Junior
189 }

```

Various macros + multilinguas.

usebib.opm

```

195 \_def\_bibwarning{\_opwarning{Missing field "\_bibfieldname" in [\EntryKey]}}
196
197 \_def\_mtdef#1#2#3#4{\_sdef{\_mt:#1:en}{#2} \_sdef{\_mt:#1:cs}{#3}
198 \_if$#4$\_slet{\_mt:#1:sk}{\_mt:#1:cs}
199 \_else \_sdef{\_mt:#1:sk}{#4}
200 \_fi
201 }

```

## 2.31.4 Usage of the bib-iso690 style

This is the iso690 bibliographic style used by OpTeX.

See op-example.bib for an example of the .bib input. You can try it by:

```

\fontfam[LMfonts]
\nocite[*]
\usebib/s (iso690) op-example
\end

```

### Common rules in .bib files

There are entries of type @F00{...} in the .bib file. Each entry consists of fields in the form name<sub>□</sub>=<sub>□</sub>"value", or name<sub>□</sub>=<sub>□</sub>{value}. No matter which form is used. If the value is pure numeric then you can say simply name<sub>□</sub>=<sub>□</sub>value. Warning: the comma after each field value is mandatory! If it is missing then the next field is ignored or bad interpreted.

The entry names and field names are case insensitive. If there exist a data field no mentioned here then it is simply ignored. You can use it to store more information (abstract, for example).

There are "standard fields" used in ancient bibTeX (author, title, editor, edition, etc., see <http://en.wikipedia.org/wiki/BibTeX>). The iso690 style introduces several "non-standard" fields: ednote, numbering, isbn, issn, doi, url, citedate, key, bibmark. They are documented here.

Moreover, there are two optional special fields:

- lang = language of the entry. The hyphenation plus autogenerated phrases and abbreviations will be typeset by this language.
- option = options by which you can control special printing of various fields.

There can be only one option field per each entry with (may be) more options separated by spaces. You can declare the global option(s) in your document applied for each entry by `\biboptions={...}`.

### The author field

All names in the author list have to be separated by “ and ”. Each author can be written by various formats (the von part is typically missing):

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

Only the Lastname part is mandatory. Examples:

```
Petr Olšák
or
Olšák, Petr

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

The separator “ and ” between authors will be converted to comma during printing, but between semi-final and final author the word “and” (or something different depending on current language) is printed.

The first author is printed in reverse order: “LASTNAME, Firstname(s) von, After” and the others author are printed in normal order: “Firstname(s) von LASTNAME, After”. This feature follows the ISO 690 norm. The Lastname is capitalized using uppercase letters. But if the `\caps` font modifier is defined, then it is used and printed `{\caps\_rm\_Lastname}`.

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

There is an `aumin:⟨number⟩` option which denotes the definitive number of printed authors if the author list is not fully printed due to `aumax`. If `aumin` is unused then `aumax` authors is printed in such case.

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

```
\biboptions={aumax:7 aumin:1}
% if there is 8 or more authors then only first author is printed.
\entdd
```

Examples:

```
\begtt
author = "John Green and Bob Brown and Alice Black",
```

output: GREEN, John, Bob BROWN, and Alice BLACK.

```
author = "John Green and Bob Brown and Alice Black",
option = "aumax:1",
```

output: GREEN, John et al.

```
author = "John Green and Bob Brown and Alice Black",
option = "aumax:2",
```

output: GREEN, John, Bob BROWN et al.

```
author = "John Green and Bob Brown and Alice Black",
option = "aumax:3",
```

output: GREEN, John, Bob BROWN, and Alice BLACK.

```
author = "John Green and Bob Brown and Alice Black",
option = "aetal",
```

output: GREEN, John, Bob BROWN, Alice BLACK et al.

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

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

output: CALBRAITH Robert [pseudonym of J. K. Rowling].

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

```
author = "John Green and Bob Brown and Alice Black",
option = "aetal autrim:1",
```

output: GREEN, J., B. BROWN, A. BLACK et al.

If you need to write a team name or institution instead authors, replace all spaces by `\_` in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after comma. Example:

```
author = "Czech\ Technical\ University\ in\ Prague,
Faculty\ of\ Electrical\ Engeneering",
```

output: CZECH TECHNICAL UNIVERSITY IN PRAGUE, Faculty of Electrical Engeneering.

### The editor field

The editor field is used for list of the authors of the collection. The analogous rules as in author field are used here. It means that the authors are separated by “ and ”, the Firstnames, Lastnames etc. are interpreted and you can use the options `edmax:⟨number⟩`, `edmin:⟨number⟩`, `edetal`, `edtrim:⟨number⟩` and `edprint:{⟨value⟩}` (with `\ED` macro). Example:

```
editor = "Jan Tomek and Petr Karas",
option = "edprint:{\ED, editors.} edtrim:1",
```

Output: J. TOMEK and P. KARAS, editors.

If `edprint` option is not set then `{\ED, eds.}` or `{\ED, ed.}` is used depending on the entry language and on the singular or plural of the editor(s).

### The ednote field

The ednote field is used as the secondary authors and more editorial info. The value is read as raw data without any interpretation of Lastname, Firstname etc.

```
ednote = "Illustrations by Robert \upper{Agarwal}, edited by Tom \upper{Nowak}",
```

output: Illustrations by Robert AGARWAL, edited by Tom NOWAK.

The `\upper` command have to be used for Lastnames in ednote field.

### The title field

This is the title of the work. It will be printed (in common entry types) by italics. The ISO 690 norm declares, that the title plus optional subtitle are in italics and they are separated by colon. Next, the optional secondary title have to be printed in upright font. This can be added by `titlepost:{⟨value⟩}`. Example:

```
title = "The Simple Title of The Work",
or
title = "Main Title: Subtitle",
or
title = "Main Title: Subtitle",
option = "titlepost:{Secondary title}",
```

The output of the last example: *Main Title: Subtitle.* Secondary title.

### The edition field

This field is used only for second or more edition of cited work. Write only the number without the word "edition". The shortcut "ed." (or something else depending on current language) is added automatically. Examples:

```
edition = "Second",
edition = "2nd",
edition = "2${\rm nd}$",
edition = "2.",
```

Output of the last example: 2. ed.

```
edition = "2."
lang     = "cs",
```

Output: 2. vyd.

Note, that the example `edition="Second"` may cause problems. If you are using language "cs" then the output is bad: Second vyd. But you can use `editionprint:{<value>}` option. The the `<value>` is printed instead of edition field and shortcut. The edition field must be set. Example:

```
edition = "whatever",
option  = "editionprint:{Second full revised edition}",
```

Output: Second full revised edition.

You can use `\EDN` macro in `editionprint` value. This macro is expanded to the edition value. Example:

```
edition = "Second",
option  = "editionprint:{\EDN\space full revised edition}",
or
edition = "Second full revised edition",
option  = "editionprint:{\EDN}",
```

### The address, publisher, year fields

This is an anachronism from ancient Bib<sub>TeX</sub> (unfortunately no exclusive) that the address field includes only the city of the publisher residence. No more data are here. The publisher field includes the name of the publisher.

```
address = "Berlin",
publisher = "Springer Verlag",
year = 2012,
```

Output: Berlin: Springer Verlag, 2012.

Note, that the year needn't to be inserted into quotes because it is pure numeric.

The letter a, b etc. are appended to the year automatically, if two or more subsequent entries in the bibliography list are not distinct by the first author and year fields. If you needn't this feature, you can use the `noautoletters` option.

You can use `"yearprint:<value>"` option. If it is set then the `<value>` is used for printing year instead the real field value. The reason: year is sort sensitive, may be you need to print something else than only sorting key. Example:

```
year    = 2000,
option  = "yearprint:{© 2000}",
```

Output: © 2000, sorted by: 2000.

```
year    = "2012a",
option  = "yearprint:{2012}",
```

Output: 2012, sorted by: 2012a.

The address, publisher and year are typically mandatory fields. If they are missing then the warning occurs. But you can set `unpublished` option. Then this warning is suppressed. There is no difference in the printed output.

### The url field

Use it without `\url` macro, but with `http://` prefix. Example:

```
url = "http://petr.olsak.net/opmac.html",
```

The ISO 690 norm recommends to add the text “Available from” (or something else if different current language is used) before URL. It means, that the output of previous example is:

Available from <http://petr.olsak.net/opmac.html>.

If the `cs` language is the current one than the output is:

Dostupné z: <http://petr.olsak.net/opmac.html>.

If the `urlalso` option is used, then the added text has the form “Available also from” or “Dostupné také z:” (if `cs` language is current).

### The citedate field

This is the citation date. The field must be in the form year/month/day. It means, that the two slashes must be written here. The output depends on the current language. Example:

```
citedate = "2004/05/21",
```

Output when `en` is current: [cit. 2004-05-21].

Output when `cs` is current: [vid. 21. 5. 2004].

### The howpublished field

This declares the available medium for cited document if it is not in printed form. Alternatives: online, CD, DVD, etc. Example:

```
howpublished = "online",
```

Output: [online].

### The volume, number, pages and numbering fields

The volume is the “big mark” of the journal issue and the number is the “small mark” of the journal issue and pages includes the page range of the cited article in the journal. The volume is prefixed by Vol. , the number by No. and the pages by pp. . But these prefixes depends on the language of the entry.

Example:

```
volume = 31,  
number = 3,  
pages = "37--42",
```

Output: Vol. 31, No. 3, pp. 37–42.

```
volume = 31,  
number = 3,  
pages = "37--42",  
lang = "cs",
```

Output: ročník 31, č. 3, s. 37–42.

If you disagree with the default prefixes, you can use the numbering field. When it is set then it is used instead of volume, number, pages fields and instead of any mentioned prefixes. The numbering can include macros `\VOL`, `\NO`, `\PP`, which are expanded to the respective values of fields. Example:

```
volume = 31,  
number = 3,  
pages = "37--42"  
numbering = "Issue~\VOL/\NO, pages~\PP",
```

Output: Issue 31/3, pages 37–42

Note: The volume, numbers and pages fields are printed without numbering filed only in the `@ARTICLE` entry. It means, that if you need to visible them in the `@INBOOK`, `@INPROCEEDINGS` etc. entries, then you must to use numbering field.

### Common notes about entries

The order of the fields in the entry is irrelevant. We use the printed order in this manual. The exclamation mark (!) denotes the mandatory field. If such field is missing then the warning occurs during processing.



If the `unpublished` option is set then the fields `address`, `publisher`, `year`, `isbn` and `pages` are not mandatory. If the `nowarn` option is set then no warnings about missing mandatory fields occurs.

If the field is used but not mentioned in the entry documentation below then it is silently ignored.

- The `@BOOK` entry

This is used for book-like entries.

Fields: `author(!)`, `title(!)`, `howpublished`, `edition`, `ednote`, `address(!)`, `publisher(!)`, `year(!)`, `citedate`, `series`, `isbn(!)`, `doi`, `url`, `note`.

The `ednote` field here means the secondary authors (illustrator, cover design etc.).

- The `@ARTICLE` entry

This is used for articles published in a journal.

Fields: `author(!)`, `title(!)`, `journal(!)`, `howpublished`, `address`, `publisher`, `month`, `year`, [`numbering` or `volume`, `number`, `pages(!)`], `citedate`, `issn`, `doi`, `url`, `note`.

If the `numbering` is used then it is used instead `volume`, `number`, `pages`.

- The `@INBOOK` entry

This is used for the part of a book.

Fields: `author(!)`, `title(!)`, `booktitle(!)`, `howpublished`, `edition`, `ednote`, `address(!)`, `publisher(!)`, `year(!)`, `numbering`, `citedate`, `series`, `isbn` or `issn`, `doi`, `url`, `note`.

The `author` field is used for author(s) of the part, the `editor` field includes author(s) or editor(s) of whole document. The `pages` field specifies the page range of the part. The `series` field can include more information about the part (chapter numbers etc.).

The `@INPROCEEDINGS` and `@CONFERENCE` entries are equivalent to `@INBOOK` entry.

- The `@THESIS` entry

This is used for student's thesis.

Fields: `author(!)`, `title(!)`, `howpublished`, `address(!)`, `school(!)`, `month`, `year(!)`, `citedate`, `type(!)`, `ednote`, `doi`, `url`, `note`.

The `type` field must include the text "Master's Thesis" or something similar (depending on the language of the outer document).

There are nearly equivalent entries: `@BACHELORSTHESIS`, `@MASTERSTHESIS` and `@PHDTHESIS`. These entries set the `type` field to an appropriate value automatically. The `type` field is optional in such case. If it is used then it has a precedence before default setting.

- The `@MISC` entry

It is intended for various usage.

Fields: `author`, `title`, `howpublished`, `ednote`, `citedate`, `doi`, `url`, `note`.

You can use `\AU`, `\ED`, `\EDN`, `\VOL`, `\NO`, `\PP`, `\ADDR`, `\PUBL`, `\YEAR` macros in `ednote` field. These macros print authors list, editors list, edition, volume, number, pages, address, publisher and year field values respectively.

The reason of this entry is to give to you the possibility to set the format of entry by your own decision. The most of data are concentrated in `ednote` field.

- The `@BOOKLET`, `@INCOLLECTION`, `@MANUAL`, `@PROCEEDINGS`, `@TECHREPORT`, `@UNPUBLISHED` entries

These entries are equivalent to `@MISC` entry because we need to save the simplicity. They are implemented only for (almost) backward compatibility with the ancient BibTeX. But the `ednote` is mandatory field here, so you cannot use these entries from the old databases without warnings and without some additional work with the `.bib` file.

### The cite-marks (bibmark) used when `\nonumcitations` is set

When `\nonumcitations` is set then `\cite` prints text oriented bib-marks instead numbers. This style file autogenerates these marks in the form "Lastname of the first author, comma, space, the year" if `bibmark` field isn't declared. If you need to set an exception from this common format, then you can use `bibmark` field.

The OPmac trick <http://petr.olsak.net/opmac-tricks-e.html#bibmark> describes how to redefine the algorithm for `bibmark` auto-generating when you need the short form of the type [Au13].

### Sorting

If `\usebib/c` is used then entries are sorted by citation order in the text. If `\usebib/s` is used then entries are sorted by "Lastname, Firstname(s)" of the first author and if more entries have this value

equal, then the year is used (from older to newer). This feature follows the recommendation of the ISO 690 norm.

If you have the same authors and the same year, you can control the sorting by setting years as 2013, 2013a, 2013b, etc. You can print something different to the list using `yearprint{⟨value⟩}` option, see the section about address, publisher and year above. The real value of year field (ie. not yearprint value) is also used in the text oriented bib-marks when `\nonumcitations` is set.

If you have some problems with name sorting, you can use the hidden field `key`, which is used for sorting instead of the “Lastname Firstname(s)” of authors. If the `key` field is unset then the “Lastname Firstname(s)” is used for sorting normally. Example:

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

This entry is now sorted between C and D.

The norm recommends to place the autocitations to the top of the list of references. You can do this by setting `key_="@"`, to each entry with your name because the `@` character is sorted before A.

## Languages

There is the language of the outer document and the languages of each entry. The ISO 690 norm recommends that the technical notes (the prefix before URL, the media type, the “and” conjunction between semifinal and final author) may be printed in the language of the outer document. The data of the entry have to be printed in the entry language (edition ed./vyd., Vol./ročník, No./č. etc.). Finally there are the phrases independent on the language (for example In:). Unfortunately, the `bibTeX` supposes that the entry data are not fully included in value parts of the fields (see edition, volume etc. fields) so the automaton have to add some text during processing. But what language have to be chosen?

The current value of the `\language` register at the start of the `.bib` processing is decided as the language of the outer document. This language is used for technical notes regardless of the entry language. Each entry can have the `lang` field with the two-letter mark of the entry language. This language is used for ed./vyd., vol./ročník etc. and it is used for hyphenation too. If the entry language is not set then the outer document language is used.

If the outer document language is known before creating of the `.bib` file, you can store some language-dependent phrases into it. On the other hand, if the main document language is unknown, you can use the `\Mtext` macro to create the text multilingual. Example:

```
howpublished = "\Mtext{blue-ray}"
```

Now, you can set the variants of blue-ray into your macros:

```
\mtdef {blue-ray} {Blue-ray disc} {Blue-ray disk} {}
```

## Tips for using more languages

This style prefers English, Czech and Slovak languages. However, you can add more languages. Use the shortcuts of language names (`de` and `pl` in the example below). You can define all phrases for your language:

```
\def\mtdefx#1#2#3{\sdef{#1:de}{#2}\sdef{#1:pl}{#3}}

\mtdefx {bib.and}          % German          % Polish
                          { und }            { a }
\mtdefx {bib.phdthesis}    {Ph.D. Dissertation} {Praca doktorska}
...
```

See more about language phrases in the [2.36.3](#) section.

## Summary of non-standard fields

This style uses the following fields unknown by `bibTeX`:

```
option    ... options separated by spaces
lang      ... the language two-letter code of one entry
ednote    ... editional info (secondary authors etc.) or
           global data in @MISC-like entries
citedate  ... the date of the citation in year/month/day format
numbering ... format for volume, number, pages
```

```

isbn      ... ISBN
issn      ... ISSN
doi       ... DOI
url       ... URL

```

## Summary of options

```

aumax:<number>      ... maximum number of printed authors
aumin:<number>      ... number of printed authors if aumax exceeds
autrim:<number>      ... full Firstnames iff number of authors are less than this
auprint:<{<value>}  ... text instead authors list (\AU macro may be used)
edmax, edmin, edtrim ... similar as above for editors list
edprint:<{<value>}  ... text instead editors list (\ED macro may be used)
titlepost:<{<value>} ... text after title
yearprint:<{<value>} ... text instead real year (\YEAR macro may be used)
editionprint:<{<value>} . text instead real edition (\EDN macro may be used)
urlalso           ... the ``available also from'' is used instead ``available from''
unpublished      ... the publisher etc. fields are not mandatory
nowarn           ... no mandatory fields

```

Another options in the option field are silently ignored.

## 2.31.5 Implementation of the bib-iso690 style

```

3 % bibliography style (iso690), version <2020-03-10>, loaded on demand by \usebib
4
5 \ifx\optexbibstyle\undefined \errmessage
6 {This file can be read by: \string\usebib/? (iso690) bibfiles command only}
7 \endinput \fi

```

bib-iso690.opm

`\_maybetod` (alias `\.` in the style file group) does not put second dot.

```

13 \def\_maybetod{\ifnum\_spacefactor=\_sfcode\.\_relax\_else.\_fi}
14 \_tmpnum=\_sfcode\.\_advance\_tmpnum by-2 \_sfcode\.\_=\_tmpnum
15 \_sfcode\?=\_tmpnum \_sfcode\!=\_tmpnum
16 \let\.\_maybetod % prevents from double periods

```

bib-iso690.opm

Option field.

```

22 \_CreateField {option}
23 \_def\_isbiboption#1#2{\_edef\_tmp{\_noexpand\_isbiboptionA{#1}}\_tmp}
24 \_def\_isbiboptionA#1{\_def\_tmp##1 #1 ##2\_relax{%
25   \_if^##2^\_csname iffalse\_ea\_endcsname \_else\_csname iftrue\_ea\_endcsname \_fi}%
26   \_ea\_tmp\_biboptionsi #1 \_relax}
27 \_def\_bibopt[#1]#2#3{\_isbiboption{#1}\_iftrue\_def\_tmp{#2}\_else\_def\_tmp{#3}\_fi\_tmp}
28 \_def\_biboptionvalue#1#2{\_def\_tmp##1 #1:##2 ##3\_relax{\_def#2{##2}}}%
29   \_ea\_tmp\_biboptionsi #1: \_relax}
30
31 \_def\_readbiboptions{%
32   \_RetrieveFieldIn{option}\_biboptionsi
33   \_toks1=\_ea{\_biboptionsi}%
34   \_edef\_biboptionsif{\_space \_the\_toks1 \_space \_the\_biboptions \_space}%
35 }
36 \_newtoks\_biboptions
37 \_public \biboptions ;

```

bib-iso690.opm

Formating of Author/Editor lists.

```

43 \_def\_firstauthorformat{%
44   \_upper{\_Lastname}\_bprintc\_Firstname{, *}\_bprintc\_Von{ *}\_bprintc\_Junior{, *}%
45 }
46 \_def\_otherauthorformat{%
47   \_bprintc\_Firstname{* }\_bprintc\_Von{* }\_upper{\_Lastname}\_bprintc\_Junior{, *}%
48 }
49 \_def\_commonname{%
50   \_ifnum\_NameCount=1
51     \_firstauthorformat

```

bib-iso690.opm

```

52 \ifx\dobibmark\undefined \edef\dobibmark{\_Lastname}\_fi
53 \else
54 \ifnum0\namecount=\_NameCount
55 \ifx\maybeetal\empty \bibconjunctionand\else , \fi
56 \else , \fi
57 \otherauthorformat
58 \fi
59 }
60 \def\_authorname{%
61 \ifnum\_NameCount>0\namecount\_relax\_else \_commonname \_fi
62 \ifnum\_NameCount=0\namecount\_relax \_maybeetal \_fi
63 }
64 \let\_editorname=\_authorname
65
66 \def\_prepareauedoptions#1{%
67 \def\_mabyetal{}\_csname lb@abbreviatefalse\_endcsname
68 \biboptionvalue{#1max}\_authormax
69 \biboptionvalue{#1min}\_authormin
70 \biboptionvalue{#1pre}\_authorpre
71 \biboptionvalue{#1print}\_authorprint
72 \isbiboption{#1etal}\_iftrue \def\_maybeetal{\_Mtext{bib.etal}}\_fi
73 \biboptionvalue{#1trim}\_autrim
74 \let\_namecountraw=\_namecount
75 \ifx\_authormax\empty \else
76 \ifnum 0\_authormax<0\_namecount
77 \edef\_namecount{\_ifx\_authormin\empty\_authormax\_else\_authormin\_fi}%
78 \def\_maybeetal{\_Mtext{bib.etal}}%
79 \fi\_fi
80 \ifx\_autrim\empty \def\_autrim{10000}\_fi
81 \ifnum\_autrim=0 \def\_autrim{10000}\_fi
82 \ifnum 0\_namecount<\_autrim\_relax \else \_AbbreviateFirstname \_fi
83 }
84 \def\_maybeetal{}
85
86 \ifx\upper\undefined
87 \ifx\caps \undefined \def\upper{\_uppercase\_ea}\_else
88 \def\upper#1{{\caps\_rm #1}}\_fi
89 \fi
90 \let\_upper=\upper

```

Preparing bib-mark (used when \nonumcitations is set).

bib-iso690.opm

```

96 \def\_setbibmark{%
97 \ifx\dobibmark\undefined \def\_dobibmark{}\_fi
98 \RetrieveFieldIn{bibmark}\_tmp
99 \ifx\_tmp\empty \_RetrieveFieldIn{year}\_tmp \edef\_tmp{\_dobibmark, \_tmp}\_fi
100 \_bibmark=\_ea{\_tmp}%
101 }
102 % Multilinguals:
103
104 \_mtdef{bib.and} {, and } { a } {}
105 \_mtdef{bib.etal} { et al.} { a~kol.} {}
106 \_mtdef{bib.edition} { ed.} { vyd.} {}
107 \_mtdef{bib.bachthesis} {Bachelor's Thesis} {Bakalářská práce} {Bakalárska práca}
108 \_mtdef{bib.masthesis} {Master's Thesis} {Diplomová práce} {Diplomová práca}
109 \_mtdef{bib.phdthesis} {Ph.D. Thesis} {Disertační práce} {Dizertačná práca}
110 \_mtdef{bib.available} {Available from } {Dostupné na } {}
111 \_mtdef{bib.availablealso} {Available also from } {Dostupné tiež na } {Dotupné tiež na }
112 \_mtdef{bib.citedate} {cit.~} {vid.~} {}
113 \_mtdef{bib.volume} {Vol.~} {ročník~} {}
114 \_mtdef{bib.number} {No.~} {č.~} {}
115 \_mtdef{bib.prepages} {pp.~} {s.~} {}
116 \_mtdef{bib.postpages} {~p.} {~s.} {}
117 \_mtdef{bib.editor} {,~ed.} {,~editor} {}
118 \_mtdef{bib.editors} {,~eds.} {,~editoři} {,~editori}
119
120 \def\_bibconjunctionand{\_Mtext{bib.and}}
121 \def\_preurl{\_Mtext{bib.available}}
122 \let\_predoi=\_preurl

```

```

123 \_def\_postedition{\_mtext{bib.edition}}
124 \_def\_Inclause{In:-}
125 \_def\_prevolume{\_mtext{bib.volume}}
126 \_def\_prenumber{\_mtext{bib.number}}
127 \_def\_prepapes{\_mtext{bib.prepages}}
128 \_def\_posteditor{\_ifnum0\_namecountraw>1 \_Mtext{bib.editors}\_else\_Mtext{bib.editor}\_fi}
129
130 \_chardef\_documentlanguage=\_language
131 \_def\_Mtext#1{\_csname\_mt:#1:\_csname\_lan:\_the\_documentlanguage\_endcsname\_endcsname}
132
133 \_CreateField {lang}
134 \_def\_setlang#1{\_ifx#1\_empty \_else
135     \_ea \_ifx \_csname\_#1Patt\_endcsname \_relax
136     \_opwarning{The language "#1" used in .bib file is unknown}
137     \_else \_language=\_csname\_#1Patt\_endcsname
138     \_fi\_fi
139 }

```

Non-standard fieldnames.

bib-iso690.opm

```

145 \_CreateField {ednote}
146 \_CreateField {citedate}
147 \_CreateField {numbering}
148 \_CreateField {isbn}
149 \_CreateField {issn}
150 \_CreateField {doi}
151 \_CreateField {url}
152 \_CreateField {bibmark}

```

Sorting.

bib-iso690.opm

```

158 \_SortingOrder{name,year}{lfvj}
159 \_SpecialSort {key}

```

Supporting macros.

bib-iso690.opm

```

165 \_def\_bibwarninga{\_bibwarning}
166 \_def\_bibwarningb{\_bibwarning}
167
168 \_def\_docitedate #1/#2/#3/#4\_relax{[\_Mtext{bib.citedate}]%
169     \_if^#2^#1\_else
170     \_if^#3^#1/#2\_else \_docitedateA{#1}{#2}{#3}%
171     \_fi\_fi ]%
172 }
173 \_def\_docitedateA#1#2#3{%
174     \_ifnum\_documentlanguage=\_csPatt \_docitedateCS{#1}{#2}{#3}%
175     \_else \_ifnum\_documentlanguage=\_skPatt \_docitedateSK{#1}{#2}{#3}%
176     \_else \_docitedateEN{#1}{#2}{#3}%
177     \_fi\_fi
178 }
179 \_def\_docitedateEN#1#2#3{#1-#2-#3}
180 \_def\_docitedateCS#1#2#3{\_hbox{\_tmpnum=#3 \_the\_tmpnum. \_tmpnum=#2 \_the\_tmpnum. #1}}
181 \_let\_docitedateSK=\_docitedateCS
182
183 \_def\_doyear#1{
184     \_biboptionvalue{yearprint}\_yearprint
185     \_ifx\_yearprint\_empty#1\_else\_def\YEAR{#1}\_yearprint\_fi
186 }
187 \_def\_preparenumbering{%
188     \_def\VOL{\_RetrieveField{volume}}%
189     \_def\NO{\_RetrieveField{number}}%
190     \_def\PP{\_RetrieveField{pages}}%
191 }
192 \_def\_prepareednote{%
193     \_def\EDN{\_RetrieveField{edition}}%
194     \_def\ADDR{\_RetrieveField{address}}%
195     \_def\PUBL{\_RetrieveField{publisher}}%
196     \_def\YEAR{\_RetrieveField{year}}%
197     \_def\AU{\_bprintb[!author]{\_doauthor0{####1}}{}}%
198     \_def\ED{\_bprintb[!editor]{\_doeditor0{####1}}{}}%

```

```

199 \_preparenumbering
200 }
201 \_def\_doedition#1{%
202 \_biboptionvalue{editionprint}\_editionprint
203 \_ifx\_editionprint\_empty#1\_postedition\_else\_def\ED{#1}\_editionprint\_fi
204 }
205 \_def\_doauthor#1#2{\_prepareauedoptions{au}\_let\_iseditorlist=\_undefined
206 \_if1#1\_def\AU{#2}\_else\_let\_authorprint=\_empty\_fi
207 \_ifx\_authorprint\_empty #2\_else \_authorprint\_fi
208 }
209 \_def\_doeditor#1#2{\_prepareauedoptions{ed}\_let\_firstauthorformat=\_otherauthorformat
210 \_if1#1\_def\ED{#2}\_else\_let\_authorprint=\_empty\_fi
211 \_ifx\_authorprint\_empty #2\_posteditor\_else \_authorprint\_fi
212 }

```

Entry types.

bib-iso690.opm

```

218 \_sdef\_print:BEGIN){%
219 \_readbiboptions
220 \_biboptionvalue{titlepost}\_titlepost
221 \_isbiboption{unpublished}\_iftrue \_let\_bibwarninga=\_relax \_let\_bibwarningb=\_relax \_fi
222 \_isbiboption{nowarn}\_iftrue \_let\_bibwarning=\_relax \_fi
223 \_isbiboption{urlalso}\_iftrue \_def\_preurl{\_Mtext{bib.availablealso}}\_fi
224 \_RetrieveFieldIn{lang}\_langentry \_setlang\_langentry
225 }
226 \_sdef\_print:END){%
227 \_bprinta [note] {*.}\_}%
228 \_setbibmark
229 }
230 \_def\_bookgeneric#1{%
231 \_bprinta [howpublished] {[*].\ }\_}%
232 \_bprintb [edition] {\_doedition{##1}\.\ }\_}%
233 \_bprinta [ednote] {*.}\_}%
234 \_bprinta [address] {*\_bprintv[publisher]{:}{,}\ }{\_bibwarninga}%
235 \_bprinta [publisher] {*\_bprintv[year]{,}{.}\ }{\_bibwarninga}%
236 \_bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}{.}\ }\_}%
237 {\_bibwarninga}%
238 \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{.}{.}\ }\_}%
239 \_bprinta [citedate] {\_docitedate*//\_relax.\ }\_}%
240 #1%
241 \_bprinta [series] {*.}\_}%
242 \_bprinta [isbn] {ISBN~*.\ }{\_bibwarningb}%
243 \_bprinta [issn] {ISSN~*.\ }\_}%
244 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }\_}%
245 \_bprintb [url] {\_preurl\_url{##1}. }\_}%
246 }
247 \_sdef\_print:book){%
248 \_bprintb [!author] {\_doauthor1{##1}\.\ }{\_bibwarning}%
249 \_bprintb [title] {\_em##1}\_bprintc\_titlepost{.\ }*\_bprintv[howpublished]{.}{.}\ }\_}%
250 {\_bibwarninga}%
251 \_bookgeneric{\_}%
252 }
253 \_sdef\_print:article){%
254 \_biboptionvalue{journalpost}\_journalpost
255 \_bprintb [!author] {\_doauthor1{##1}\.\ }{\_bibwarning}%
256 \_bprinta [title] {*. \_bprintc\_titlepost{*. \ }{\_bibwarning}%
257 \_bprintb [journal] {\_em##1}\_bprintc\_journalpost{.\ }*\_bprintv[howpublished]{.}{.}\ }\_}%
258 {\_bibwarninga}%
259 \_bprinta [howpublished] {[*].\ }\_}%
260 \_bprinta [address] {*\_bprintb[publisher]{:}{,}\ }\_}%
261 \_bprinta [publisher] {*, }\_}%
262 \_bprinta [month] {*, }\_}%
263 \_bprintb [year] {\_doyear{##1}\_bprintv[volume,number,pages]{,}{.}\ }\_}%
264 \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{.}{.}\ }
265 {\_bprinta [volume] {\_prevolume*\_bprintv[number,pages]{,}{.}\ }\_}%
266 {\_bprinta [number] {\_prenumber*\_bprintv[pages]{,}{.}\ }\_}%
267 {\_bprintb [pages] {\_prepages\_hbox{##1}\_bprintv[citedate]{.}{.}\ }\_}%
268 {\_bibwarninga}%
269 \_bprinta [citedate] {\_docitedate*//\_relax.\ }\_}%

```

```

270 \_bprinta [issn] {ISSN~*. \ }{}%
271 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}. \ }{}%
272 \_bprintb [url] {\_preurl\_url{##1}. \ }{}%
273 }
274 \_sdef{_print:inbook}{%
275 \_let\_bibwarningb=\_relax
276 \_bprintb [!author] {\_doauthor1{##1}\. \ }{\_bibwarning}%
277 \_bprinta [title] {*. \ }{\_bibwarning}%
278 \_Inclause
279 \_bprintb [!editor] {\_doeditor1{##1}\. \ }{}%
280 \_bprintb [booktitle] {\_em##1}\_bprintc\_titlepost{\. \ *}\_bprintv[howpublished]{\.\. \ }%
281 {\_bibwarning}%
282 \_bookgeneric{\_bprintb [pages] {\_prepages\_hbox{##1}. \ }{}%
283 }
284 \_slet{_print:inproceedings}{\_print:inbook}
285 \_slet{_print:conference}{\_print:inbook}
286
287 \_sdef{_print:thesis}{%
288 \_bprintb [!author] {\_doauthor1{##1}\. \ }{\_bibwarning}%
289 \_bprintb [title] {\_em##1}\_bprintc\_titlepost{\. \ *}\_bprintv[howpublished]{\.\. \ }%
290 {\_bibwarning}%
291 \_bprinta [howpublished] {[*]. \ }{}%
292 \_bprinta [address] {*\_bprintv[school]{\.\. \ }{\_bprintv[year]{\.\. \ }{\_bibwarning}%
293 \_bprinta [school] {*\_bprintv[year]{\.\. \ }{\_bibwarning}%
294 \_bprinta [month] {*, \ }{}%
295 \_bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\.\. \ }{\_bibwarning}%
296 \_bprinta [citedate] {\_docitedate*///\_relax. \ }{}%
297 \_bprinta [type] {*\_bprintv[ednote]{\.\. \ }%
298 {\_ifx\_thesistype\_undefined\_bibwarning
299 \_else\_thesistype\_bprintv[ednote]{\.\. \ } \_fi}%
300 \_bprinta [ednote] {*. \ }{}%
301 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}. \ }{}%
302 \_bprintb [url] {\_preurl\_url{##1}. \ }{}%
303 }
304 \_sdef{_print:phdthesis}{\_def\_thesistype{\_Mtext{bib.phdthesis}}\_cs{_print:thesis}}
305 \_sdef{_print:mastersthesis}{\_def\_thesistype{\_Mtext{bib.mastthesis}}\_cs{_print:thesis}}
306 \_sdef{_print:bachelorstthesis}{\_def\_thesistype{\_Mtext{bib.bachthesis}}\_cs{_print:thesis}}
307
308 \_sdef{_print:generic}{%
309 \_bprintb [!author] {\_doauthor1{##1}\. \ }{\_bibwarning}%
310 \_bprintb [title] {\_em##1}\_bprintc\_titlepost{\. \ *}\_bprintv[howpublished]{\.\. \ }%
311 {\_bibwarning}%
312 \_bprinta [howpublished] {[*]. \ }{}%
313 \_bprinta [ednote] {\_prepareednote*\_bprintv[citedate]{\.\. \ }{\_bibwarning}%
314 \_bprinta [year] {\.\. \ }{\_bibwarning}%
315 \_bprinta [citedate] {\_docitedate*///\_relax. \ }{}%
316 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}. \ }{}%
317 \_bprintb [url] {\_preurl\_url{##1}. \ }{}%
318 }
319 \_slet{_print:booklet}{\_print:generic}
320 \_slet{_print:incollecion}{\_print:generic}
321 \_slet{_print>manual}{\_print:generic}
322 \_slet{_print:proceedings}{\_print:generic}
323 \_slet{_print:techreport}{\_print:generic}
324 \_slet{_print:unpublished}{\_print:generic}
325
326 \_sdef{_print:misc}{\_let\_bibwarning=\_relax \_cs{_print:generic}}

```

## 2.32 Sorting and making Index

makeindex.opm

```
3 \_codedecl \_makeindex {Makeindex and sorting <2020-04-26>} % loaded in format
```

`\_makeindex` implements sorting algorithm at  $\text{\TeX}$  macro-language level. You need not any external program.

There are two passes in sorting algorithm. Primary pass does not distinguish between a group of letters (typically non-accented and accented). If the result of comparing two string is equal in primary pass then secondary pass is started. It distinguish between variously accented letters. Czech rules, for



example says: not accented before dieresis before acute before circumflex before ring. At less priority: lowercase letters must be before uppercase letters.

The `\_sortingdata<iso-code>` implements these rules for the language `<iso-code>`. The groups between commas are not distinguished in the first pass. The second pass distinguishes all characters mentioned in the `\_sortingdata<iso-code>` (commas are ignored). The order of letters in the `\_sortingdata<iso-code>` macro is significant for sorting algorithm. The Czech rules (cs) are implemented here:

makeindex.opm

```

25 \_def \_sortingdatacs {%
26 /,{ },-,&,@,%
27 aAäÄáÁ,%
28 bB,%
29 cC,%
30 ěĚ,%
31 dDďĎ,%
32 eEéÊëĚ,%
33 fF,%
34 gG,%
35 hH,%
36 ^T^U^V,% ch Ch CH
37 iIíÎ,%
38 jJ,%
39 kK,%
40 lLĺĹ,%
41 mM,%
42 nNňÑ,%
43 oOöŮóôõ,%
44 pP,%
45 qQ,%
46 rRřŘ,%
47 řŘ,%
48 sS,%
49 šŠ,%
50 tTťĚ,%
51 uUůŮuůú,%
52 vV,%
53 wW,%
54 xX,%
55 yYýÝ,%
56 zZ,%
57 žŽ,%
58 0,1,2,3,4,5,6,7,8,9,'%
59 }

```

Characters ignored by sorting algorithm are declared in `\_ignoredchars<iso-code>`. The compound characters (two or more characters interpreted as one character in sorting algorithm) is mapped to single invisible characters in `\_compoundchars<iso-code>`. Czech rules declares ch or Ch or CH as a single letter sorted between H and I. See `\_sortingdatacs` above where these declared characters are used.

The characters declared in `\_ignoredchars` are ignored in first pass without additional condition. All characters are taken into account in second pass: ASCII characters with code '65 are sorted first if they are not mentioned in the `\_sortingdata<iso-code>` macro. Others not mentioned characters have undefined behavior during sorting.

makeindex.opm

```

76 \_def \_ignoredcharscs {.,;?!:'"()[]<>=+}
77 \_def \_compoundcharscs {ch:~T Ch:~U CH:~V} % DZ etc. are sorted normally

```

Slovak sorting rules are the same as Czech. The macro `\_sortingdatacs` includes Slovak letters too. Compound characters are the same. English sorting rules can be defined by `\_sortingdatacs` too because English alphabet is subset of Czech and Slovak alphabets. Only difference: `\_compoundcharsen` is empty in English rules.

You can declare these macros for more languages, if you wish to use `\makeindex` with sorting rules in respect to your language. Note: if you need to map compound characters to a character, don't use `~I` or `~M` because these characters have very specific category code. And use space to separate more mappings, like in `\_compoundcharscs` above.

makeindex.opm

```

93 \_let \_sortingdatacs = \_sortingdatacs

```

```

94 \let \compoundcharssk = \compoundcharscs
95 \let \ignoredcharssk = \ignoredcharscs
96 \let \sortingdataen = \sortingdatacs
97 \def \compoundcharsen {}
98 \let \ignoredcharsen = \ignoredcharscs

```

Preparing to primary pass is implemented by the `\setprimarysorting` macro. It is called from `\makeindex` macro and all processing of sorting is in a group.

makeindex.opm

```

105 \def\setprimarysorting {%
106   \ea\let \ea\sortingdata \csname _sortingdata\sortinglang\endcsname
107   \ea\let \ea\compoundchars \csname _compoundchars\sortinglang\endcsname
108   \ea\let \ea\ignoredchars \csname _ignoredchars\sortinglang\endcsname
109   \ifx \sortingdata\relax \addto\_nold{ sortingdata}%
110     \let \sortingdata = \sortingdataen \fi
111   \ifx \compoundchars\relax \addto\_nold{ compoundchars}%
112     \let \compoundchars = \compoundcharsen \fi
113   \ifx \ignoredchars\relax \addto\_nold{ ignoredchars}%
114     \let \ignoredchars = \ignoredcharsen \fi
115   \ifx \compoundchars\_empty \else
116     \edef \compoundchars {\detokenize\ea{\compoundchars}} \fi % all must be catcode 12
117   \def \act ##1{\ifx##1\relax \else
118     \ifx##1,\advance\tmpnum by1
119     \else \lccode`##1=\tmpnum \fi
120     \ea\act \fi}%
121   \tmpnum=65 \ea\act \sortingdata \relax
122   \def \act ##1{\ifx##1\relax \else
123     \lccode`##1=\^^I
124     \ea\act \fi}%
125   \ea\act \ignoredchars \relax
126 }

```

Preparing to secondary pass is implemented by the `\setsecondarysorting` macro.

makeindex.opm

```

132 \def\setsecondarysorting {%
133   \def \act ##1{\ifx##1\relax \else
134     \ifx##1,\else \advance\tmpnum by1 \lccode`##1=\tmpnum \fi
135     \ea\act \fi}%
136   \tmpnum=65 \ea\act \sortingdata \relax
137 }

```

Strings to be sorted are prepared in `\,<string>` control sequences (in order to save `\TeX` memory). The `\preparesorting` `\,<string>` converts `<string>` to `\tmpb` with respect to the data initialized in `\setprimarysorting` or `\setsecondarysorting`.

The compound characters are converted to single characters by the `\docompound` macro.

makeindex.opm

```

149 \def \preparesorting #1{%
150   \edef \tmpb {\ea\ignorefirst\csstring #1}% \,<string> -> <string>
151   \ea \docompound \compoundchars \relax:{} % replace compound characters
152   \lowercase \ea{\ea\def \ea\tmpb \ea{\tmpb}}% convert in respect to \sortingdata
153   \ea\replstring \ea\tmpb \ea{\csstring\^^I}{}% remove ignored characters
154 }
155 \def \docompound #1:#2 {%
156   \ifx\relax#1\else \replstring\tmpb {#1}{#2}\ea\docompound \fi
157 }
158 \def \ignorefirst#1{}

```

Macro `\isAleB` `\,<string1>` `\,<string2>` returns the result of comparison of given two strings to `\ifAleB` control sequence. Usage: `\isAleB` `\,<string1>` `\,<string2>` `\ifAleB` ... `\else` ... `\fi` The converted strings (in respect of the data prepared for first pass) must be saved as values of `\,<string1>` and `\,<string2>` macros. The reason is speed: we don't want to convert them repeatedly in each comparison. The macro `\testAleB` `<converted string1>``&\relax``<converted-string2>``\relax` `\,<string1>``\,<string2>` does the real work. It reads first character from both converted strings, compares them and if it is equal then calls itself recursively else gives result.

makeindex.opm

```

175 \newif \ifAleB
176
177 \def\isAleB #1#2{%

```

```

178 \_edef\_tmpb {#1&\_relax#2&\_relax}%
179 \_ea \_testAleB \_tmpb #1#2%
180 }
181 \_def\_testAleB #1#2\_relax #3#4\_relax #5#6{%
182 \_if #1#3\_if #1&\_testAleBsecondary #5#6% goto to the second pass::
183 \_else \_testAleB #2\_relax #4\_relax #5#6%
184 \_fi
185 \_else \_ifnum `#1<`#3 \_AleBtrue \_else \_AleBfalse \_fi
186 \_fi
187 }
188 \_def\_testAleBsecondary#1#2{%
189 \_bgroup
190 \_setsecondarysorting
191 \_preparesorting#1\_let\_tmpa=\_tmpb \_preparesorting#2%
192 \_edef\_tmpb{\_tmpa0\_relax\_tmpb1\_relax}%
193 \_ea \_testAleBsecondaryX \_tmpb
194 \_egroup
195 }
196 \_def\_testAleBsecondaryX #1#2\_relax #3#4\_relax {%
197 \_if #1#3\_testAleBsecondaryX #2\_relax #4\_relax
198 \_else \_ifnum `#1<`#3 \_global\_AleBtrue \_else \_global \_AleBfalse \_fi
199 \_fi
200 }

```

Merge sort is very effectively implemented by T<sub>E</sub>X macros. The following code is created by my son Miroslav. The `\_mergesort` macro expects that all items in `\_iilist` are separated by comma when it starts. It ends with sorted items in `\_iilist` without commas. So `\_dosorting` macro must prepare commas between items.

```

210 \_def\_mergesort #1#2,#3{% by Miroslav Olsak
211 \_ifx,#1% % prazdna-skupina,neco, (#2=neco #3=pokracovani)
212 \_addto\_iilist{#2,}% % dvojice skupin vyresena
213 \_sortreturn{\_fif\_mergesort#3}% % \mergesort pokracovani
214 \_fi
215 \_ifx,#3% % neco,prazna-skupina, (#1#2=neco #3=,)
216 \_addto\_iilist{#1#2,}% % dvojice skupin vyresena
217 \_sortreturn{\_fif\_mergesort}% % \mergesort dalsi
218 \_fi
219 \_ifx\_end#3% % neco,konec (#1#2=neco)
220 \_ifx\_empty\_iilist % neco=kompletni setrideny seznam
221 \_def\_iilist{#1#2}%
222 \_sortreturn{\_fif\_fif\_gobbletoend}% % koncim
223 \_else % neco=posledni skupina nebo \end
224 \_sortreturn{\_fif\_fif % spojim \indexbuffer+necoa cele znova
225 \_edef\_iilist{\_ea\_mergesort\_iilist#1#2,#3}%
226 \_fi\_fi % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
227 \_isAleB #1#3\_ifAleB % p1<p2
228 \_addto\_iilist{#1}% % p1 do bufferu
229 \_sortreturn{\_fif\_mergesort#2,#3}% % \mergesort neco1,p2+neco2,
230 \_else % p1>p2
231 \_addto\_iilist{#3}% % p2 do bufferu
232 \_sortreturn{\_fif\_mergesort#1#2,}% % \mergesort p1+neco1,neco2,
233 \_fi
234 \_relax % zarazka, na ktere se zastavi \sortreturn
235 }
236 \_def\_sortreturn#1#2\_fi\_relax{#1} \_def\_fif{\_fi}
237 \_def\_gobbletoend #1\_end{}

```

The `\_dosorting` `\list` macro redefines `\list` as sorted `\list`. The `\list` have to include control sequences in the form `\<c>\<string>`. These control sequences will be sorted in respect to `\<strings>` without change of meanings of these control sequences. Their meanings are irrelevant when sorting. The first character `\<c>` in `\<c>\<string>` should be whatever. It does not influence the sorting. OpT<sub>E</sub>X uses comma at this place for sorting indexes: `\,<word1> \,<word2> \,<word3> ...`

The actual language (chosen for hyphenation patterns) is used for sorting data. If the `\_sortinglang` macro is defined as `\<iso-code>`(for example `\_def\_sortinglang{de}`) then this has precedence and actual language is not used. Moreover, if you specify `\_asciisortingtrue` then ASCII sorting will be processed and all language sorting data will be ignored.

```

256 \_newifi \_ifasciisorting \_asciisortingfalse
257 \_def\_dosorting #1{%
258   \_begingroup
259   \_def\_nold{%}
260   \_ifx\_sotringlang\_undefined \_edef\_sortinglang{\_cs{\_lan:\_the\_language}}\_fi
261   \_ifasciisorting
262     \_edef\_sortinglang{ASCII}%
263     \_def \_preparesorting##1{\_edef\_tmpb{\_ea\_ignorefirst\_csstring##1}}%
264     \_let \_setsecondarysorting=\_relax
265   \_else
266     \_setprimarysorting
267   \_fi
268   \_message{OpTeX: Sorting \_string#1 (\_sortinglang) ...^^J}%
269   \_ifx\_nold\_empty\_else \_opwarning{Missing\_nold\_space for language (\_sortinglang)}\_fi
270   \_def \_act##1{\_preparesorting ##1\_edef##1{\_tmpb}}%
271   \_ea\_xargs \_ea\_act #1;%
272   \_def \_act##1{\_addto #1{##1,}}%
273   \_edef #1{\_ea}\_ea\_xargs \_ea\_act #1;%
274   \_edef \_iilist{\_ea}\_ea\_mergesort #1\_end,\_end
275   \_ea\_endgroup
276   \_ea\_def\_ea#1\_ea{\_iilist}%
277 }

```

The `\makeindex` prints the index. First, it sorts the `\_iilist` second, it prints the sorted `\_iilist`, each item is printed using `\_printindexitem`.

```

285 \_def\_makeindex{\_par
286   \_ifx\_iilist\_empty \_opwarning{index data-buffer is empty. TeX me again}
287   \_else
288     \_dosorting \_iilist % sorting \_iilist
289     \_bgroup
290     \_rightskip=0pt plus1fil \_exhyphenpenalty=10000 \_leftskip=\_iindent
291     \_ea\_xargs \_ea\_printindexitem \_iilist ;\_par
292     \_egroup
293   \_fi
294 }
295 \_public \_makeindex ;

```

The `\_printindexitem` `\,<word>` prints one item to the index. If `\,<word>` is defined then this is used instead real `<word>` (this exception is declared by `\iis` macro). Else `<word>` is printed by `\_printii`. Finally, `\_printiipages` prints the value of `\,<word>`, i.e. the list of pages.

```

305 \_def\_printindexitem #1{%
306   \_ifcsname \_csstring #1\_endcsname
307     \_ea\_ea\_ea \_printii \_csname \_csstring #1\_endcsname &%
308   \_else
309     \_ea\_ea\_ea \_printii \_ea\_ignorefirst \_csstring #1&%
310   \_fi
311   \_ea\_printiipages #1&
312 }

```

`\_printii` `<word>&` does more intelligent work because we are working with words in the form `<main-word>/<sub-word>/<sub-sub-word>`. The `\everyii` tokens register is applied before `\noindent`. User can declare something special here.

The `\_newiiletter` `{<letter>}` macro is empty by default. It is invoked if first letter of index entries is changed. You can declare a design between index entries here. You can try, for example:

```

\def\_newiiletter#1#2{%
  \bigskip \hbox{\setfontsize{at15pt}\bf\uppercase{#1}}\medskip}

```

```

329 \_def\_printii #1#2{%
330   \_ismacro\_lastii{#1}\_iffalse \_newiiletter{#1}{#2}\_def\_lastii{#1}\_fi
331   \_gdef\_currii{#1#2}\_the\_everyii\_noindent
332   \_hskip-\_iindent \_ignorespaces\_printiiA#1#2//}
333 \_def\_printiiA #1/{\_if^#1\_let\_previi=\_currii \_else
334   \_ea\_scanprevii\_previi/&\_edef\_tmpb{\_detokenize{#1}}%
335   \_ifx\_tmpa\_tmpb \_iiemdash \_else#1 \_gdef\_previi{}\_fi
336   \_expandafter\_printiiA\_fi

```

```

337 }
338 \def\iiemdash{\_kern.1em---\_space}
339 \def\lastii{}
340 \def\newiiletter#1#2{}
341
342 \def\scanprevii#1/#2&{\_def\_previi{#2}\_edef\_tmpa{\_detokenize{#1}}}
343 \def\_previi{} % previous index item

```

**\\_printii**pages  $\langle pglis \rangle$  & gets  $\langle pglis \rangle$  in the form  $\langle pg \rangle : \langle type \rangle, \langle pg \rangle : \langle type \rangle, \dots \langle pg \rangle : \langle type \rangle$  and it converts them to  $\langle pg \rangle$ ,  $\langle pg \rangle$ ,  $\langle from \rangle -- \langle to \rangle$ ,  $\langle pg \rangle$  etc. The same pages must be printed only once and continuous consequences of pages must be compressed to the form  $\langle from \rangle - \langle to \rangle$ . Moreover, the consequence is continuous only if all pages have the same  $\langle type \rangle$ . Empty  $\langle type \rangle$  is most common, pages with **b**  $\langle type \rangle$  must be printed as bold and with *i*  $\langle type \rangle$  as italics. Moreover, the  $\langle pg \rangle$  mentioned here are  $\langle gpageno \rangle$ , but we have to print  $\langle pageno \rangle$ . The following macros solves these tasks.

makeindex.opm

```

357 \def\_printii#1&{\_let\_pgtype=\_undefined\_tmpnum=0\_printpages #1,.,\_par}
358 \def\_printpages#1:#2,{% state automaton for comprimg pages
359 \_ifx,#1,\_uselastpgnum
360 \_else \_def\_tmpa{#2}%
361 \_ifx\_pgtype\_tmpa \_else
362 \_let\_pgtype=\_tmpa
363 \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
364 \_tmpnum=#1 \_returnfi \_fi
365 \_ifnum\_tmpnum=#1 \_returnfi \_fi
366 \_advance\_tmpnum by1
367 \_ifnum\_tmpnum=#1 \_ifx\_lastpgnum\_undefined \_usepgdash\_fi
368 \_edef\_lastpgnum{\_the\_tmpnum:{\_pgtype}}%
369 \_returnfi \_fi
370 \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
371 \_tmpnum=#1
372 \_relax
373 \_ea\_printpages \_fi
374 }
375 \def\_returnfi #1\_relax{\_fi}
376 \def\_uselastpgnum{\_ifx\_lastpgnum\_undefined
377 \_else \_ea\_pgprint\_lastpgnum \_let\_lastpgnum=\_undefined \_fi
378 }
379 \def\_usepgcomma{\_ifnum\_tmpnum>0, \_fi} % comma+space between page numbers
380 \def\_usepgdash{\_hbox{--}} % dash in the <from>--<to> form

```

You can re-define **\\_pgprint**  $\langle gpageno \rangle : \{ \langle iitype \rangle \}$  if you need to implement more  $\langle iitype \rangle$ .

makeindex.opm

```

387 \def\_pgprint #1:#2{%
388 \_ifx ,#2,\_pgprintA{#1}\_returnfi \_fi
389 \_ifx b#2{\_bf \_pgprintA{#1}}\_returnfi \_fi
390 \_ifx i#2{\_it \_pgprintA{#1}}\_returnfi \_fi
391 \_ifx u#2{\_pgu{\_pgprintA{#1}}}\_returnfi \_fi
392 \_pgprintA{#1}\_relax
393 }
394 \def\_pgprintA #1{\_ilink[pg:#1]{\_cs{\_pgi:#1}}} % \ilink[pg:<gpageno>]{<pageno>}
395 \def\_pgu#1{\_leavevmode\_vtop{\_hbox{#1}\_kern.3ex\_hrule}}

```

The **\\_iindex**{ $\langle word \rangle$ } puts one  $\langle word \rangle$  to the index. It writes **\\_Xindex**{ $\langle word \rangle$ }{ $\langle iitype \rangle$ } to the .ref file. All othes variants of indexing macros expands internally to **\\_iindex**.

makeindex.opm

```

403 \def\_iindex#1{\_isempty{#1}\_iffalse\_openref{def~{ }%
404 \edef\_act{\_noexpand\_wref\_noexpand\_Xindex{#1}{\_iitiesaved}}}\_act}\_fi}
405 \_public \_iindex ;

```

The **\\_Xindex**{ $\langle word \rangle$ }{ $\langle iitype \rangle$ } stores  $\backslash, \langle word \rangle$  to the **\\_iilist** if there is first occurrence of the  $\langle word \rangle$ . The list of pages where  $\langle word \rangle$  occurs, is the value of the macro  $\backslash, \langle word \rangle$ , so the  $\langle gpageno \rangle : \langle iitype \rangle$  is appended to this list. Moreower, we need a mapping from  $\langle gpageno \rangle$  to  $\langle pageno \rangle$ , because we print  $\langle pageno \rangle$  in the index, but hyperlinks are implemented by  $\langle gpageno \rangle$ . So, the macro **\\_pgi**: $\langle gpageno \rangle$  is defined as  $\langle pageno \rangle$ .

makeindex.opm

```

417 \def \_iilist {}
418 \def \_Xindex #1#2{\_ea\_XindexA \_csname ,#1\_ea\_endcsname \_currpage {#2}}
419 \def \_XindexA #1#2#3#4{% #1=\,<word> #2=<gpageno> #3=<pageno> #4=<iitype>

```

```

420 \_ifx#1\_relax\_global\_addto\_iilist{#1}%
421 \_gdef #1{#2:#4}%
422 \_else\_global\_addto #1{, #2:#4}%
423 \_fi
424 \_sxddef{\_pgi:#2}{#3}%
425 }

```

The implementation of macros `\ii`, `\iid`, `\iis` follows. Note that `\ii` works in horizontal mode on order to the `\write` whatsit is not broken from the following word. If you need to keep vertical mode, use `\iindex{<word>}` directly.

The `\iitype{<type>}` saves the `<type>` to the `\_iitypesaved` macro. It is used in the `\iindex` macro.

makeindex.opm

```

437 \_def\_ii #1 {\_leavevmode\_def\_tmp{#1}\_iiA #1,,\_def\_iitypesaved{}}
438
439 \_def\_iiA #1,{\_if$#1$\_else\_def\_tmpa{#1}%
440 \_ifx\_tmpa\_iiatsign \_ea\_iiB\_tmp,,\_else\_iindex{#1}\_fi
441 \_ea\_iiA\_fi}
442 \_def\_iiatsign{@}
443
444 \_def\_iiB #1,{\_if$#1$\_else \_iiC#1/\_relax \_ea\_iiB\_fi}
445 \_def\_iiC #1/#2\_relax{\_if$#2$\_else\_iindex{#2#1}\_fi}
446
447 \_def\_iid #1 {\_leavevmode\_iindex{#1}#1\_futurelet\_tmp\_iid\_def\_iitypesaved{}}
448 \_def\_iid{\_ifx\_tmp,\_else\_ifx\_tmp.\_else\_space\_fi\_fi}
449
450 \_def\_iis #1 #2{{\_def~{ }\_global\_sdef{_,#1}{#2}}\_ignorespaces}
451
452 \_def\_iitypesaved{}
453 \_def\_iitype #1{\_def\_iitypesaved{#1}\_ignorespaces}
454
455 \_public \ii \iid \iis \iitype ;

```

## 2.33 Footnotes and marginal notes

fnotes.opm

```

3 \_codedecl \fnote {Footnotes, marginal notes OpTeX <2020-03-20>} % loaded in format

```

`\_gfnotenum` is counter which counts footnotes globally in the document. whole document, chapters, pages.

`\_lfnotenum` is counter which counts footnotes at each chapter from one. It is used for local page footnote counters too.

`\_ifpgfnote` says that footnote numbers are counted on each page from one. We need to run `\openref` in such case.

`\fnotenum` is a macro which expands to footnote number counted in declared part.

`\fnotenumchapters` declares footnotes numbered in each chapter from one (default), `\fnotenumglobal` declares footnotes numbered in whole document from one and `\fnotenumpages` declares footnotes numbered at each page from one.

fnotes.opm

```

19 \_newcount\_gfnotenum \_gfnotenum=0
20 \_newcount\_lfnotenum
21
22 \_newifi \_ifpgfnote
23 \_def \_fnotenumglobal {\_def\_fnotenum{\_the\_gfnotenum}\_pgfnotefalse}
24 \_def \_fnotenumchapters {\_def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse}
25 \_def \_fnotenumpages {\_def\_fnotenum{\_trycs\_fn:\_the\_gfnotenum}{?}}\_pgfnotetrue}
26 \_fnotenumchapters % default are footnotes counted from one in each chapter
27 \_def \_fnotenum{\_fnotenum}
28 \_public \fnotenumglobal \fnotenumchapters \fnotenumpages ;
29 \_let \runningfnotes = \fnotenumglobal % for backward compatibility

```

The `\_printfnotemark` prints the footnote mark. You can re-define this macro if you want another design of footnotes. For example

```

\_fnotenumpages
\_def \_printfnotemark {\ifcase 0\fnotenum\or
  *\or**\or***\or$\mathbox{†}$\or$\mathbox{‡}$\or$\mathbox{††}$\fi}

```



This code gives footnotes\* and \*\* and\*\*\* and† etc. and it supposes that there are no more than 6 footnotes at one page.

If you want to distinguish between footnote marks in the text and in the front of footnote itself, then you can define `\_printfnoteA` and `\_printfnoteB`.

The `\fnotelinks<colorA><colorB>` implements the hyperlinked footnotes (from text to footnote and backward).

fnotes.opm

```

49 \def \_printfnoteA {$^{\_fnoteA}$} % default footnote mark
50 \def \_printfnoteB {\_printfnoteA} % footnote marks used in text
51 \def \_printfnoteC {\_printfnoteA} % footnote marks used in front of footnotes
52
53 \def \fnotelinks#1#2{% <inText color> <inFootnote color>
54 \def\_printfnoteA{\_link[fnt:\_the\_gfnoteA]{\_localcolor#1}{\_printfnoteA}}%
55 \_dest[fnt:\_the\_gfnoteA]}%
56 \def\_printfnoteB{\_link[fnt:\_the\_gfnoteB]{\_localcolor#2}{\_printfnoteB}}%
57 \_dest[fnt:\_the\_gfnoteB]}%
58 }
59 \public \fnotelinks ;

```

Each footnote saves the `\_Xfnote` (without parameter) to the `.ref` file (if `\openref`). We can create the mapping from `<gfnoteA>` to `<pgfnoteA>` in the macro `\_fn:<gfnoteA>`. Each `\_Xpage` macro sets the `\_lfnoteA` to zero.

fnotes.opm

```

68 \def \_Xfnote {\_incr\_lfnoteA \_incr\_gfnoteA
69 \_sxdef{\_fn:\_the\_gfnoteA}{\_the\_lfnoteA}}

```

The `\fnote{<text>}` macro is simple, `\fnoteA` and `\fnoteB` does the real work.

fnotes.opm

```

76 \def \fnote{\_fnoteA \_fnoteB}
77 \def \fnoteA#1{\_advance\_gfnoteA by#1\_advance\_lfnoteA by#1\_relax \_printfnoteA}

```

The `\fnoteB` calls `\opfootnote` which is equivalent to plain TeX `\footnote`. It creates new data to Insert `\footins`. The only difference is that we are able to propagate a macro parameter into Insert group before the text is printed (see section 2.17). This propagated macro is `\_fnset` which sets smaller fonts.

Note that `\footnote` and `\opfootnote` does't read the text as a parameter but during normal horizontal mode. This is reason why catcode changes (for example in-line verbatim) can be used here.

fnotes.opm

```

91 \def \fnoteB{\_incr\_gfnoteA \_incr\_lfnoteA % global increment
92 \_ifpgfnote \openref \_fi
93 \_wref \_Xfnote}%
94 \_ifpgfnote \_ifcname \_fn:\_the\_gfnoteA \_endcsname \_else
95 \_opwarning{unknown \_noexpand\fnote mark. TeX me again}
96 \_fi\_fi
97 \opfootnote\_fnset\_printfnoteB
98 }
99 \def \_fnset{\_everypar={}\_scalemain \_typoscale[800/800]}
100
101 \_public \fnote \fnoteA \fnoteB ;

```

By default `\mnote{<text>}` are in right margin at odd pages and they are in left margin at even pages. The `\mnote` macro saves its position to `.ref` file as `\_Xmnote` without parameter. We define `\_mn:<mnoteA>` as `\right` or `\left` when the `.ref` file is read. The `\ifnum 0<#2` trick returns true if `<pageno>` has numeric type and false if it is non-numeric type (Roman numeral, for example). We prefer to use `<pageno>`, but only if it has numeric type. We use `<pgpageno>` in other cases.

fnotes.opm

```

113 \_newcount \_mnoteA \_mnoteA=0 % global counter of mnotes
114 \def \_Xmnote {\_incr\_mnoteA \_ea \_XmnoteA \_currpage}
115 \def \_XmnoteA #1#2{% #1=<pgpageno> #2=<pageno>
116 \_sxdef{\_mn:\_the\_mnoteA}{\_ifodd\_numtype{#2}{#1} \_right \_else \_left \_fi}}
117 \def \_numtype #1#2{\_ifnum 0<#1 #1\_else #2\_fi}

```

User can declare `\fixmnotes\left` or `\fixmnotes\right`. It defines `\_mnotesfixed` as `\_left` or `\_right` which declares the placement of all marginal notes and such declaration has a precedence.

fnotes.opm

```

125 \def \fixmnotes #1{\_edef\_mnotesfixed{\_cs{\_csstring #1}}
126 \_public \fixmnotes ;

```



The outer box of marginal note has zero width and zero depth and it is appended after current line using `\vadjust` primitive or it is inverted to vertical mode as a box with `\vskip-\baselineskip` followed.

fnotes.opm

```
134 \long\def\mnote#1{\ifvmode {\mnoteA{#1}}\nobreak\vskip-\baselineskip \else
135 \lower\dp\strutbox\hbox{\vadjust{\kern-\dp\strutbox \mnoteA{#1}\kern\dp\strutbox}}%
136 \fi
137 }
138 \public \mnote ;
```

The `\mnoteA` macro does the real work. The `\lrnote{<left>}{<right>}` uses only first or only second parameter depending on the left or right marginal note.

fnotes.opm

```
146 \long\def\mnoteA #1{\incr\mnotenum
147 \ifx\mnnotesfixed\undefined
148 \ifcsname _mn:\the\mnotenum \endcsname
149 \edef\mnnotesfixed{\cs{mn:\the\mnotenum}}%
150 \else
151 \opwarning{unknown \noexpand\mnote side. TeX me again}\openref
152 \def\mnnotesfixed{\right}%
153 \fi\fi
154 \hbox to0pt{\wref\Xmnote{\everypar={}}%
155 \lrnote{\kern-\mnnotesize \kern-\mnnoteindent}{\kern\hsize \kern\mnnoteindent}}%
156 \vbox to0pt{\vss \setbox0=\vtop{\hsize=\mnnotesize
157 \lrnote{\leftskip=0pt plus 1fill \rightskip=0pt}
158 {\rightskip=0pt plus 1fil \leftskip=0pt}}%
159 {\the\everymnote\noindent#1\endgraf}}%
160 \dp0=0pt \box0 \kern\mnnoteskip \global\mnnoteskip=0pt\hss}%
161 }
162 \def \lrnote#1#2{\ea\ifx\mnnotesfixed\left #1\else #2\fi}
```

We don't want to process `\fnote`, `\fnotemark`, `\mnote` in TOC, headlines nor outlines.

fnotes.opm

```
169 \regmacro {\def\fnote#1{}} {\def\fnote#1{}} {\def\fnote#1{}}
170 \regmacro {\def\fnotemark#1{}} {\def\fnotemark#1{}} {\def\fnotemark#1{}}
171 \regmacro {\def\mnote#1{}} {\def\mnote#1{}} {\def\mnote#1{}}
```

## 2.34 Styles

OpTeX provides three styles: `\report`, `\letter` and `\slides`. Their behavior is documented in user part of the manual in the section 1.7.2 and `\slides` style (for presentations) is documented in `op-slides.pdf` which is an example of the presentation.

### 2.34.1 \report and \letter styles

styles.opm

```
3 \codedcl \report {Basic styles of OpTeX <2020-03-28>} % preloaded in format
```

We define auxiliary macro first (used by the `\address` macro)

The `{\boxlines <line-1>\eol<line-2>\eol...<line-n>\eol}` returns to the outer vertical mode a box with `<line-1>`, next box with `<line-2>` etc. Each box has its natural width. This is reason why we cannot use paragraph mode where each resulting box has the width `\hsize`. The `\eol` is set active and `\everypar` starts `\hbox{` and active `\eol` closes this `\hbox` by `}`.

styles.opm

```
16 \def\boxlines{%
17 \def\boxlinesE{\ifhmode\egroup\empty\fi}%
18 \def\nl{\boxlinesE}%
19 \bgroup \lccode\~=\^M\lowercase{\egroup\let~}\boxlinesE
20 \everypar{\setbox0=\lastbox\endgraf
21 \hbox\bgroup \catcode\^M=13 \let\par=\nl \aftergroup\boxlinesC}%
22 }
23 \def\boxlinesC{\futurelet\next\boxlinesD}
24 \def\boxlinesD{\ifx\next\empty\else\ea\egroup\fi}
25
26 \public \boxlines ;
```

The `\report` and `\letter` style initialization macros are defined here.

The `\letter` defines `\address` and `\subject` macros.

```

34 \_def\_report{
35   \_typo[11/13.2]
36   \_vsize=\_dimexpr \_topskip + 52\_baselineskip \_relax % added 2020-03-28
37   \_let\_titfont=\_chapfont
38   \_titskip=3ex
39   \_eoldef\_author##1{\_removelastskip\_bigskip
40     {\_leftskip=0pt plus1fill \_rightskip=\_leftskip \_it \_noindent ##1\_par}\_nobreak\_bigskip
41   }
42   \_public \author ;
43   \_parindent=1.2em \_iindent=\_parindent \_ttindent=\_parindent
44   \_footline={\_global\_footline={\_hss\_rmfixed\_folio\_hss}}
45 }
46 \_def\_letter{
47   \_def\_address{\_vtop\_bgroup\_boxlines \_parskip=0pt \_let\_par=\_egroup}
48   \_def\_subject{{\_bf \_mtext{subj}: }}
49   \_public \address \subject ;
50   \_typo[11/14]
51   \_vsize=\_dimexpr \_topskip + 49\_baselineskip \_relax % added 2020-03-28
52   \_parindent=0pt
53   \_parskip=\_medskipamount
54   \_nopagenumbers
55 }
56 \_public \letter \report ;

```

The `\slides` macro reads macro file `slides.opm`, see the section 2.34.2.

```

62 \_def\_slides{\_par
63   \_input slides.opm
64 }
65 \_public \slides ;

```

## 2.34.2 \slides style for presentations

```

3 \_codedecl \slideshow {Slides style for OpTeX <2020-03-19>} % loaded on demand by \slides

```

Default margins and design is declared here. The `\ttfont` is scaled by `mag1.15` in order to balance the ex height of Helvetica (Heros) and LM fonts Typewriter. The `\begtt...\endtt` verbatim is printed by smaller text.

```

12 \_margins/1 a5l (14,14,10,3)mm % landscape A5 format
13 \_def\_wideformat{\_margins/1 (263,148) (16,16,10,3)mm } % 16:9 format
14
15 \_fontfam[Heros]
16 \_typo[16/19]
17 \_famvardef\_ttfont{\_setfontsize{mag1.15}\_tt}
18 \_def\_urlfont{}
19 \_everytt={\_typo[13/16] \advance\hsize by10mm}
20 \_fontdef\_fixbf{\_bf}
21
22 \_nopagenumbers
23 \_parindent=0pt
24 \_ttindent=5mm
25 \_parskip=5pt plus 4pt minus2pt
26 \_rightskip=0pt plus 1fil
27 \_ttindent=10pt
28 \_def\_ttskip{\_smallskip}
29
30 \_onlyrgb % RGB color space is better for presentations

```

The bottom margin is set to 3 mm. If we use 1 mm, then baseline of `\footline` is 2 mm from the bottom page. This is depth of the `\Grey` rectangle used for page numbers. It is r-lapped to `\hoffset` width because left margin = `\hoffset` = right margin. It is 14 mm for narrow pages or 16 mm for wide pages.

```

40 \_footlinedist=1mm
41 \_footline={\_hss \_rlap{%
42   \_rlap{\Grey\_kern.2\_hoffset\_vrule height6mm depth2mm width.8\_hoffset}%
43   \_hbox to\_hoffset{\White\_hss\_folio\_kern3mm}}}

```

The `\subtit` is defined analogically like `\tit`.

```

49 \_eoldef\_subtit#1{\_vskip20pt {\_leftskip=0pt plus1fill \_rightskip=\_leftskip
50 \_subtitfont #1\_nbp}}

```

The `\pshow<num>` prints the text in invisible (transparent) font when `\layernum<num>`. The transparency is set by `\pdfpagemresources` primitive.

```

58 \pdfpagemresources{/ExtGState << /Invisible << /Type /ExtGState /ca 0 /CA 0 >>
59 /Visible << /Type /ExtGState /ca 1 /CA 1 >> >>}
60 \addto\_morepgrresources{/Invisible << /Type /ExtGState /ca 0 /CA 0 >>
61 /Visible << /Type /ExtGState /ca 1 /CA 1 >>}
62 \def\Invisible {\_pdfliteral{/Invisible gs}}
63 \def\Visible {\_pdfliteral{/Visible gs}}
64 \def\Transparent {\Invisible \_aftergroup \Visible}
65
66 \_def\_use#1#2{\_ifnum\_layernum#1\_relax#2\_fi}
67 \_def\_pshow#1{\_use{#1}\Red \_use{<#1>\Transparent \_ignorespaces}

```

The main level list of items is activated here. The `\_item:X` and `\_item:x` are used and are re-defined here. If we are in nested level of items and `\pg+` is used then `\egroups` macro expands to the right number of `\egroups` in order to close page correctly. The level of nested item lists is saved to the `\_ilevel` register and used when we start again the next text after `\pg+`.

```

79 \_newcount\_gilevel
80 \_def\*{*}
81 \_adef*\_startitem}
82 \_sdef\_item:X{\Blue\_raise.2ex\_fullrectangle{.8ex}\_kern.5em}
83 \_sdef\_item:x{\Blue\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
84 \_style X
85 \_def\_egroups{\_par\_global\_gilevel=\_ilevel \_egroup}
86 \_everylist={\_novspaces \_ifcase\_ilevel \_or \_style x \_else \_style - \_fi
87 \_addto\_egroups{\_egroup}}

```

The default values of `\pg`, i.e. `\pg;`, `\pg+` and `\pg.` are very simple. They are used when `\showslides` is not specified.

```

94 \_def\_pg#1{\_cs{\_spg:#1}}
95 \_sdef{\_spg:;}{\_vfil\_break \_lfnotenumreset}
96 \_sdef{\_spg:.\_end}
97 \_sdef{\_spg:+}\_par}

```

We need no numbers and no table of contents when using slides. The `\_printsec` macro is redefined in order the title is centered and typeset in `\Blue`.

```

105 \_def\_titfont{\_typosize[42/60]\_bf \Blue}
106 \_def\_subtitfont{\_typosize[20/30]\_bf}
107 \_def\_secfont{\_typosize[25/30]\_bf \Blue}
108
109 \_nonum \_notoc \_let\_resetnonumnotoc=\_relax
110 \_def\_printsec#1{\_par
111 \_abovetitle{\_penalty-400}\_bigskip
112 {\_secfont \_noindent \_leftskip=0pt plus1fill \_rightskip=\_leftskip
113 \_printrefnum[@\_quad]#1\_nbp}\_insertmark{#1}%
114 \_nobreak \_belowtitle{\_medskip}%
115 }

```

When `\slideshow` is active then each page is opened by `\setbox\_slidepage=\vbox\bgroup` (roughly speaking) and closed by `\egroup`. The material is `\unvboxed` and saved for the usage in the next usage if `\pg+` is in process. The `\_slidelayer` is incremented instead `\pageno` if `\pg+`. This counter is equal to `\count1`, so it is printed to the terminal and log file next to `\pageno`.

The code is somewhat more complicated when `\layers` is used. Then `<layered-text>` is saved to the `\_layertext` macro, the material before it is in `\_slidepage` box and the material after it is in `\_slidepageB` box. The pages are completed in the `\loop` which increments the `\layernum` register.

```

133 \_newbox\_slidepage \_newbox\_slidepageB
134 \_countdef\_slidelayer=1
135 \_def\_decr#1{\_global\_advance#1 by-1 }
136
137 \_def\_slideshow{\_slidelayer=1 \_slideshowactive \_setbox\_slidepage=\vbox\bgroup}

```

```

138
139 \def\slideshowactive{%
140   \sdef\spg:;}{\closepage \global\slidelayer=1 \resetpage \openslide}
141   \sdef\spg:;}{\closepage \end}
142   \sdef\spg:;}{\closepage \incr\slidelayer \decr\pageno \openslide}
143   \def\bye {\closepage \end}
144   \let\layers=\layersactive
145   \def\destbox[##1:##2]{\_isequal{##1}{ref}\_iffalse \destboxori[##1:##2]\_fi}%
146 }
147 \def\openslide{\setbox\slidepage=\vbox\bgroup \setilevel
148   \ifvoid\slidepage \else \unvbox\slidepage \nointerlineskip\lastbox \fi}
149 \def\setilevel{\_loop \decr\gilevel \ifnum\gilevel<0 \else \begitem \repeat}
150
151 \def\closepage{\_egroups
152   \ifnum \maxlayers=0 \unvcopy\slidepage \vfil\break
153   \else \begingroup \setwarnslides \layernum=0
154     \_loop
155       \ifnum\layernum<\maxlayers \advance\layernum by1
156       \printlayers \vfil\break
157       \ifnum\layernum<\maxlayers \incr\slidelayer \decr\pageno \fi
158     \repeat
159     \global\maxlayers=0
160     \incr\layernum \global\setbox\slidepage=\vbox{\printlayers}%
161   \endgroup
162   \fi}
163 \def\resetpage{%
164   \global\setbox\slidepage=\_box\_voidbox \global\setbox\slidepageB=\_box\_voidbox
165   \lfnotenumberreset
166 }
167 \def\setwarnslides{%
168   \def\pg##1{\_opwarning{\_string\pg##1 \layersenv}\_def\pg####1{}}%
169   \def\layers##1 {\_opwarning{\_string\layers\_space \layersenv}\_def\layers####1{}}%
170 }
171 \def\layersenv{cannot be inside \_string\layers...\_string\endlayers, ignored}
172
173 \def\printlayers{\_unvcopy\slidepage \nointerlineskip\lastbox
174   \_layertext \endgraf
175   \_ifdim\_prevdepth>-1000pt \_kern-\_prevdepth \_kern\_dp\_strutbox \_fi
176   \vskip\_parskip
177   \unvcopy\slidepageB
178 }
179 \let\destboxori=\destbox
180
181 \_newcount\_layernum \_newcount\_maxlayers
182 \_maxlayers=0
183
184 \_long\_def\_layersactive #1 #2\endlayers{%
185   \_par\_egroup
186   \_gdef\_layertext{#2}%
187   \_global\_maxlayers=#1
188   \setbox\slidepageB=\vbox\bgroup
189 }
190 \_public \subtit \slideshow \pg \wideformat \use \pshow ;

```

Default `\layers`  $\langle num \rangle$  macro (when `\slideshow` is not activated) is simple. It prints the  $\langle layered-text \rangle$  with `\layernum`= $\langle num \rangle+1$  because we need the result after last layer is processed.

slides.opm

```

198 \def\layers #1 {\_par\_layernum=\_numexpr#1+1\_relax}
199 \let\endlayers=\_relax
200
201 \def\layers{\_layers}

```

We must to redefine `\fnotenumpages` because the data from `.ref` file are less usable for implementing such feature: the footnote should be in more layers repeatedly. But we can suppose that each page starts by `\pg`; macro, so we can reset the footnote counter by this macro.

slides.opm

```

211 \def \fnotenumpages {\_def\_fnotenum{\_the\_lfnnotenum}\_pgfnotefalse
212   \_def\_lfnnotenumreset{\_global\_lfnnotenum=0 }}
213 \let \lfnnotenumreset=\_relax

```

```
214 \_public \fnotenumpages ;
```

## 2.35 Logos

logos.opm

```
3 \_codedecl \TeX {Logos TeX, LuaTeX, etc. <2020-02-28>} % preloaded in format
```

Despite plain TeX each macro for logos ends by `\ignoreslash`. This macro ignores next slash if it is present. You can use `\TeX/` like this for protecting the space following the logo. This is visually more comfortable. The macros `\TeX`, `\OpTeX`, `\LuaTeX`, `\XeTeX` are defined.

logos.opm

```
13 \_protected\def \TeX {T\_kern-.1667em\_lower.5ex\_hbox{E}\_kern-.125emX\_ignoreslash}
14 \_protected\def \OpTeX {Op\_kern-.1em\_TeX}
15 \_protected\def \LuaTeX {Lua\_TeX}
16 \_protected\def \XeTeX {X\_kern-.125em\_phantom E%
17 \_pdfsave\_rlap{\pdfscale{-1}{1}\_lower.5ex\_hbox{E}}\_pdfrestore \_kern-.1667em \TeX}
18
19 \def\ignoreslash {\_futurelet\_next \ignoreslashA}
20 \def\ignoreslashA {\_ifx\_next/\_ea\_ignoreit\_fi}
21
22 \_public \TeX \OpTeX \LuaTeX \XeTeX \ignoreslash ;
```

The `\slantcorr` macro expands to slant-correction of current font. It is used to shifting A if the `\LaTeX` logo is in italic.

logos.opm

```
29 \_protected\def \LaTeX{\_tmpdim=.42ex L\_kern-.36em \_kern \slantcorr % slant correction
30 \_raise \_tmpdim \_hbox{\_thefontscale[710]A}%
31 \_kern-.15em \_kern-\slantcorr \TeX}
32 \def\slantcorr{\_ea\_ignorept \_the\_fontdimen1\_font\_tmpdim}
33
34 \_public \LaTeX ;
```

`\OPmac`, `\CS` and `\csplain` logos.

logos.opm

```
40 \_def\_OPmac{\_leavevmode
41 \_lower.2ex\_hbox{\_thefontscale[1400]O}\_kern-.86em P{\_em mac}\_ignoreslash}
42 \_def\_CS{\$\_cal C$\_kern-.1667em\_lower.5ex\_hbox{\$\_cal S}\_ignoreslash}
43 \_def\_csplain{\_CS plain\_ignoreslash}
44
45 \_public \OPmac \CS \csplain ;
```

The expandable versions of logos used in Outlines needs the expandable `\ingnslash` (instead of the `\ignoreslash`).

logos.opm

```
52 \_def\ingnslash#1{\_ifx/#1\_else #1\_fi}
53 \_regmacro {}{}{% conversion for PDF outlines
54 \_def\TeX\TeX\_ignslash}\_def\OpTeX\OpTeX\_ignslash}%
55 \_def\LuaTeX\LuaTeX\_ignslash}\_def\XeTeX\XeTeX\_ignslash}%
56 \_def\LaTeX\LaTeX\_ignslash}\_def\OPmac\OPmac\_ignslash}%
57 \_def\CS\CS}\_def\csplain\csplain\_ignslash}%
58 }
59 \_public \ingnslash ;
```

## 2.36 Multilingual support

### 2.36.1 Lowercase, uppercase codes

All codes in unicode table keep information about pairs lowercase-uppercase letters or single letter. We need to read such information and set appropriate `\lccode` and `\uccode`. The `\catcode` above the code 127 is not set, i. e. the `\catcode=12` for all codes above 127.

The file `uni-lcuc.opm` does this work. It is not much interesting file, only first few lines from 15928 lines in total is shown here.

uni-lcuc.opm

```
3 % Preloaded in format. A copy o uni-lcuc.tex fom csplain is here:
4
5 % uni-lcuc.tex -- sets \lccodes and \uccodes for Unicode chars, nothing more
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

7 % Petr Olsak, Jul. 2014
8
9 \_wterm{Setting lccodes and uccodes for Unicode characters}
10
11 \_def\_tmp #1 #2 {\_ifx~#1~\_else
12   \_lccode"#1="#1
13   \_ifx.#2%
14     \_uccode"#1="#1
15   \_else
16     \_uccode"#2="#2
17     \_lccode"#2="#1
18     \_uccode"#1="#2
19   \_fi
20   \_ea \_tmp \_fi
21 }
22 \_tmp
23 00AA .
24 00B5 039C
25 00BA .
26 00E0 00C0
27 00E1 00C1
28 00E2 00C2
29 00E3 00C3
30 00E4 00C4

```

...etc. (see `uni-lcuc.opm`)

## 2.36.2 Hyphenations

```

3 \_codedecl \_langlist {Initialization of hyphenation patterns <2020-03-10>} % preloaded in format

```

The `<iso-code>` means a shortcut of language name (mostly by ISO 639-1). The following control sequences are used for language switching:

- `\_lan:<number>` expands to `<iso-code>` of the language. The number is internal number of languages used as a value of `\_language` register.
- `\_ulan:<long-lang>` expands to `<iso-code>` too. This is transformation from long name of language (lowercase letters) to `<iso-code>`.
- `\_<iso-code>Patt` (for example `\_csPatt`) is the language `<number>` declared by `\_chardef`.
- `\_<iso-code>lang` (for example `\_enlang`, `\_cslang`, `\_sklang`, `\_delang`, `\_pllang`) is language selector. It exists in two states

- Initialization state: when `\_<iso-code>lang` is used first then it must load the patterns into memory using Lua code. If it is done then the `\_<iso-code>lang` re-defines itself to processing state.
- Processing state: it only sets `\_language=\_<iso-code>Patt`, i.e it selects the hyphenation patterns. It does a little more language-dependent work, as mentioned below.
- `\_langspecific:<isocode>` is processed by `\_<iso-code>lang` and it should include language-specific macros declared by user or macro designer.

The USenglish patterns are preloaded first:

```

32 \_chardef\_enPatt=0
33 \_def\_pattlist{\_enPatt=0}
34 \_def\_langlist{en(USenglish)}
35 \_sdef{\_lan:0}{en}
36 \_sdef{\_ulan:usenglish}{en}
37 \_def\_enlang{\_uselang{en}\_enPatt23} % \lefthyph=2 \righthyph=3
38 \_def\_enlang{\_enlang}
39 \_sdef{\_langspecific:en}{\_nonfrenchspacing}
40
41 \lefthyphenmin=2 \righthyphenmin=3 % disallow x- or -xx breaks
42 \input hyphen % en(USenglish) patterns from TeX82

```

`\_prelang <iso-code> <long-lang> <number-cs> <number> <pre-hyph> <post-hyph>` prepares the `\_<iso-code>lang` to its initialization state. Roughly speaking, it does:

```

\chardef\_{iso-code}Patt = \number
\def\_lan:\number {iso-code}
\def\_ulan:\long-lang {iso-code}
\def\_iso-code\lang {%
  \loadpatttrs \long-lang \number % loads patterns using Lua code
  \gdef\_iso-code\lang {\_uselang{iso-code}\_iso-codePatt \pre-hyph\post-hyph}}
  \_iso-code\lang % runs itself in processing state
}
\def\_iso-code\lang {\_iso-code\lang} % public version \_iso-code\lang

```

You can see that `\_iso-code\lang` runs `\_loadpatttrs \long-lang \iso-code` in initialization state and `\_uselang` in processing state.

hyphen-lan.opm

```

63 \_def\_preplang #1 #2 #3#4 #5 {%
64   \_chardef#3=#4
65   \_sdef\_lan:#4#{#1}\_lowercase{\_sdef\_ulan:#2}#{#1}%
66   \_def\_next{\_ea\_noexpand\_csname\_#1lang\_endcsname}
67   \_ea\_edef\_csname\_#1lang\_endcsname {%
68     \_lowercase{\_noexpand\_loadpatttrs #2} #4 % loads patterns
69     \_gdef\_next{\_noexpand\_uselang{#1}#3#5}% re-defines itself
70     \_next % runs itself in processing state
71   }
72   \_addto\_langlist{ #1(#2)}%
73   \_sdef{#1lang}{\_csname\_#1lang\_endcsname}% unprefixes \_isocode>lang
74 }
75 \_def\_loadpatttrs#1 #2 {%
76   \_directlua{
77     require("luatex-hyphen")
78     luatexhyphen.loadlanguage("#1",#2)
79   }%
80 }

```

`\_uselang{iso-code}\_iso-codePatt \pre-hyph\post-hyph}`

sets `\language`, `\lefthyphenmin`, `\righthyphenmin` and runs `\frenchspacing`. This default language-dependent settings should be re-declared by `\_langspecific:iso-code` which is run finally (it is `\relax` by default, only `\_langspecific:en` runs `\nonfrenchspacing`).

hyphen-lan.opm

```

91 \_def\_uselang#1#2#3#4{\_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\_relax
92   \_frenchspacing % \nonfrenchspacing can be set in \cs{langspecific:lan}
93   \_cs{langspecific:#1}%
94 }

```

The `\uselanguage {long-lang}` is defined here (for compatibility with e-plain users).

hyphen-lan.opm

```

100 \_def\_uselanguage#1{\_lowercase{\_cs{\_cs{ulan:#1}lang}}}
101 \_public \uselanguage ;

```

The numbers for languages are declared as fixed constants (no auto-generated). This concept is inspired from CSplain. There are typical numbers of languages in CSplain: 5=Czech in IL2, 15=Czech in T1 and 115=Czech in Unicode. We keep these constants but we load only Unicode patterns (greater than 100), of course.

hyphen-lan.opm

```

111 \_preplang enus USenglishmax \_enusPatt 100 23
112 \_preplang engb UKenglish \_engbPatt 101 23
113 \_preplang it Italian \_itPatt 102 22
114 \_preplang ia Interlingua \_iaPatt 103 22
115 \_preplang id Indonesian \_idPatt 104 22
116
117 \_preplang cs Czech \_csPatt 115 23
118 \_preplang sk Slovak \_skPatt 116 23
119 \_preplang de nGerman \_dePatt 121 22
120 \_preplang fr French \_frPatt 122 22
121 \_preplang pl Polish \_plPatt 123 22
122 \_preplang cy Welsh \_cyPatt 124 23
123 \_preplang da Danish \_daPatt 125 22
124 \_preplang es Spanish \_esPatt 126 22
125 \_preplang sl Slovenian \_slPatt 128 22

```



```

126 \_preplang fi Finnish \_fiPatt 129 22
127 \_preplang hy Hungarian \_huPatt 130 22
128 \_preplang tr Turkish \_trPatt 131 22
129 \_preplang et Estonian \_etPatt 132 23
130 \_preplang eu Basque \_euPatt 133 22
131 \_preplang ga Irish \_gaPatt 134 23
132 \_preplang nb Bokmal \_nbPatt 135 22
133 \_preplang nn Nynorsk \_nnPatt 136 22
134 \_preplang nl Dutch \_nlPatt 137 22
135 \_preplang pt Portuguese \_ptPatt 138 23
136 \_preplang ro Romanian \_roPatt 139 22
137 \_preplang hr Croatian \_hrPatt 140 22
138 \_preplang zh Pinyin \_zhPatt 141 11
139 \_preplang is Icelandic \_isPatt 142 22
140 \_preplang hsb Uppersorbian \_hsbPatt 143 22
141 \_preplang af Afrikaans \_afPatt 144 12
142 \_preplang gl Galician \_glPatt 145 22
143 \_preplang kmr Kurmanji \_kmrPatt 146 22
144 \_preplang tk Turkmen \_tkPatt 147 22
145 \_preplang la Latin \_laPatt 148 22
146 \_preplang lac classicLatin \_lacPatt 149 22
147 \_preplang lal liturgicalLatin \_lalPatt 150 22
148 \_preplang elm monoGreek \_elmPatt 201 11
149 \_preplang elp Greek \_elpPatt 202 11
150 \_preplang grc ancientGreek \_grcPatt 203 11
151 \_preplang ca Catalan \_caPatt 204 22
152 \_preplang cop Coptic \_copPatt 205 11
153 \_preplang mn Mongolian \_mnPatt 206 22
154 \_preplang sa Sanskrit \_saPatt 207 13
155 \_preplang ru Russian \_ruPatt 208 22
156 \_preplang uk Ukrainian \_ukPatt 209 22
157 \_preplang hy Armenian \_hyPatt 210 12
158 \_preplang as Assamese \_asPatt 211 11
159 \_preplang hi Hindi \_hiPatt 212 11
160 \_preplang kn Kannada \_knPatt 213 11
161 \_preplang lv Latvian \_lvPatt 215 22
162 \_preplang lt Lithuanian \_ltPatt 216 22
163 \_preplang ml Malayalam \_mlPatt 217 11
164 \_preplang mr Marathi \_mrPatt 218 11
165 \_preplang or Oriya \_orPatt 219 11
166 \_preplang pa Panjabi \_paPatt 220 11
167 \_preplang ta Tamil \_taPatt 221 11
168 \_preplang te Telugu \_tePatt 222 11

```

The `\langlist` includes names of all languages which are ready to load and use their hyphenation patterns. This list is printed to terminal and to log at iniT<sub>E</sub>X state here. It can be used when processing document too.

[hyphen-lan.opm](#)

```

176 \message{Language hyph.patterns ready to load: \_langlist.
177 Use \_string<shortname>lang to initialize language,
178 \_string<cslang>\_space for example}
179
180 \_public \langlist ;

```

Maybe, you need to do more language specific actions than just switching hyphenation patterns. For example you need to load a specific font with a specific script used in selected language, you can define a macros for quotation marks depending on the language etc.

The example shows how to declare such language specific things.

```

\def\langset #1 #2{\sdef{_\langspecific:#1}{#2}}

\langset fr {... declare French quotation marks}
\langset de {... declare German quotation marks}
\langset gr {... switch to Greek fonts family}
... etc.

```

Note that you need not to set language specific phrases (like `\today`) by this code. Another concept is used for such tasks. See the section [2.36.3](#) for more details.

## 2.36.3 Multilingual phrases and quotation marks

languages.opm

```
3 \_codedecl \_mtext {Languages <2020-04-29>} % preloaded in format
```

Only four words are generated by OpTeX macros: “Chapter”, “Table”, “Figure” and “Subject”. These phrases can be generated depending on the current value of `\language` register, if you use `\_mtext{⟨phrase-id⟩}`, specially `\_mtext{chap}`, `\_mtext{t}`, `\_mtext{f}` or `\_mtext{subj}`. If your macros generate more words then you can define such words by `\sdef{\_mt:⟨phrase-id⟩:⟨lang⟩}` where `⟨phrase-id⟩` is a label for declared word and `⟨lang⟩` is language shortcut (iso code).

languages.opm

```
16 \_def \_mtext#1{\_trycs{\_mt:#1:\_trycs{\_lan:\_the\_language}{en}}
17   {\_csname \_mt:#1:en\_endcsname}}
18
19 \_sdef{\_mt:chap:en}{Chapter} \_sdef{\_mt:chap:cs}{Kapitola} \_sdef{\_mt:chap:sk}{Kapitola}
20 \_sdef{\_mt:t:en}{Table} \_sdef{\_mt:t:cs}{Tabulka} \_sdef{\_mt:t:sk}{Tabuľka}
21 \_sdef{\_mt:f:en}{Figure} \_sdef{\_mt:f:cs}{Obrázek} \_sdef{\_mt:f:sk}{Obrázok}
22 \_sdef{\_mt:subj:en}{Subject} \_sdef{\_mt:subj:cs}{Věc} \_sdef{\_mt:subj:sk}{Vec}
```

Using `\_langw ⟨lang⟩ ⟨chapter⟩ ⟨table⟩ ⟨figure⟩ ⟨subject⟩` you can declare these words more effectively:

languages.opm

```
30 \_def \_langw #1 #2 #3 #4 #5 {%
31   \_sdef{\_mt:chap:#1}{#2}\_sdef{\_mt:t:#1}{#3}\_sdef{\_mt:f:#1}{#4}%
32   \_sdef{\_mt:subj:#1}{#5}%
33 }
34
35 \_langw en Chapter Table Figure Subject
36 %-----
37 \_langw cs Kapitola Tabulka Obrázek Věc
38 \_langw de Kapitel Tabelle Abbildung Betreff
39 \_langw es Capítulo Tabla Figura Sujeto
40 \_langw fr Chapitre Tableau Figure Matière
41 \_langw it Capitolo Tabella Fig. Soggetto
42 \_langw pl Rozdział Tabela Ilustracja Temat
```

...etc. (see `languages.opm`)

You can add more words as you wish. For example `\today` macro:

languages.opm

```
51 \_def \_monthw #1 #2 #3 #4 #5 #6 #7 {%
52   \_sdef{\_mt:m1:#1}{#2}\_sdef{\_mt:m2:#1}{#3}\_sdef{\_mt:m3:#1}{#4}%
53   \_sdef{\_mt:m4:#1}{#5}\_sdef{\_mt:m5:#1}{#6}\_sdef{\_mt:m6:#1}{#7}%
54   \_monthwB #1
55 }
56 \_def \_monthwB #1 #2 #3 #4 #5 #6 #7 {%
57   \_sdef{\_mt:m7:#1}{#2}\_sdef{\_mt:m8:#1}{#3}\_sdef{\_mt:m9:#1}{#4}%
58   \_sdef{\_mt:m10:#1}{#5}\_sdef{\_mt:m11:#1}{#6}\_sdef{\_mt:m12:#1}{#7}%
59 }
60
61 \_monthw en January February March April May June
62           July August September October November December
63 \_monthw cs ledna února března dubna května června
64           července srpna září října listopadu prosince
65 \_monthw sk januára februára marca apríla mája júna
66           júla augusta septembra októbra novembra decembra
67
68 \_sdef{\_mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
69 \_sdef{\_mt:today:cs}{\_the\_day.~\_mtext{m\_the\_month} \_the\_year}
70 \_slet{\_mt:today:sk}{\_mt:today:cs}
71
72 \_def \_today{\_mtext{today}}
73 \_public \today ;
```

Quotes should be tagged by `\"⟨text⟩"` and `\'⟨text⟩'` if `\⟨iso-code⟩quotes` is declared at beginning of the document (for example `\enquotes`). If not, then the control sequences `\"` and `\'` are undefined. Remember, that they are used in another meaning when `\oldaccents` command is used. The macros `\"` and `\'` are not defined as `\protected` because we need their expansion when `\outlines` are created. User can declare quotes by `\quoteschars⟨clqq⟩⟨crqq⟩⟨clq⟩⟨crq⟩`, where `⟨clqq⟩...⟨crqq⟩` are normal quotes and `⟨clq⟩...⟨crq⟩` are alternative quotes. or use `\altquotes` to swap between meaning of these two types

of quotes.

`\enquotes`, `\csquotes`, `\dequotes`, `\frquotes` etc. are defined here.

languages.opm

```

90 \def \enquotes {\quoteschars ""'}
91 \def \csquotes {\quoteschars "","'}
92 \def \frquotes {\quoteschars "«»}
93 \let \plquotes = \frquotes
94 \let \esquotes = \frquotes
95 \let \grquotes = \frquotes
96 \let \ruquotes = \frquotes
97 \let \itquotes = \frquotes
98 \let \skquotes = \csquotes
99 \let \dequotes = \csquotes
100
101 \def \quoteschars #1#2#3#4{\def\altquotes{\quoteschars#3#4#1#2}\_public\altquotes;%
102 \def \###1'#{#1##1#2}\def \###1'#{#3##1#4}}

```

Sometimes should be usable to leave the markup "such" or 'such' i.e. without the first backslash. Then you can make the characters " and ' active by the `\activequotes` macro and leave quotes without first backslash. First, declare `\(iso-code)quotes`, then `\altquotes` (if needed) and finally `\activequotes`.

languages.opm

```

112 \def\activequotes{\ea\activequotesA\""\ea\activequotesA\''}
113 \def\activequotesA#1#2#3{\_bgroup\_lccode\~=#3\_lowercase{\_egroup\_adef#3##1~{#1##1#2}}
114
115 \_public \quoteschars \activequotes \enquotes \csquotes \skquotes \frquotes \plquotes
116 \esquotes \grquotes \ruquotes \itquotes \dequotes ;

```

## 2.37 Other macros

Miscellaneous macros are here.

others.opm

```

3 \codeldecl \uv {Miscellaneous <2020-04-02>} % preloaded in format

```

`\useOpTeX` and `\useoptex` are declared as `\relax`.

others.opm

```

9 \let \useOpTeX = \relax \let \useoptex = \relax

```

The `\lastpage` and `\totalpages` get the information from the `\_currpage`. The `\_Xpage` from `.ref` file sets the `\_currpage`.

others.opm

```

16 \def\totalpages {\_openref\_ea\_lastpageA\_currpage}
17 \def\lastpage {\_openref\_ea\_lastpageB\_currpage}
18 \def\_lastpageA #1#2{#1}
19 \def\_lastpageB #1#2{#2}
20 \def\_currpage {0}{?}
21 \_public \lastpage \totalpages ;

```

We need `\uv`, `\clqq`, `\crqq`, `\flqq`, `\frqq`, `\uslang`, `\ehyph` `\chyph`, `\shyph`, for backward compatibility with `Csplain`. Codes are set according to Unicode, because we are using Czech only in Unicode when `LuaTeX` is used.

others.opm

```

30
31 % for compatibility with csplain:
32
33 \chardef\clqq=8222 \chardef\crqq=8220
34 \chardef\flqq=171 \chardef\frqq=187
35 \chardef\promile=8240
36
37 \def\uv#1{\clqq#1\crqq}
38
39 \let\uslang=\enlang \let\ehyph=\enlang
40 \let\chyph=\cslang \let\shyph=\sklang
41 \let\csUnicode=\csPatt \let\czUnicode=\csPatt \let\skUnicode=\skPatt

```

The `\letfont` was used in `Csplain` instead of `\fontlet`.

others.opm

```

47 \let \letfont = \fontlet

```

Non breaking space in Unicode.

```
53 \let ^^a0=~
```

TikZ needs these funny control sequences.

```
59 \_ea\_toksdef \_csname toks@\_endcsname=0
60 \_ea\_let \_csname voidb@x\_endcsname=\_voidbox
```

We don't want to read `opmac.tex` unless `\input opmac` is specified.

```
66 \_def\OPmacversion{OpTeX}
```

We allow empty lines in math formulae. It is more comfortable.

```
72 \_suppressmathparerror = 1
```

Lorem ipsum can be printed by `\lipsum[⟨range⟩]` or `\lorem[⟨range⟩]`, for example `\lipsum[3]` or `\lipsum[112-121]`, `max=150`. The data are read from L<sup>A</sup>T<sub>E</sub>X file `lipsum.ltd.tex`.

```
80 \_def \_lipsum {%
81   {\_long\_def\ProvidesFile##1[##2]##3{\_ifx\_par##3\_relax\_else \_ea##3\_fi}\_tmpnum=0
82   \_def\NewLipsumPar{\_advance\_tmpnum by1
83     \_afterassignment\_negativermmn \_sxddef{lips:\_the\_tmpnum}}}%
84   \_opinput {lipsum.ltd.tex}%
85   \_global\_let \_lipsum=\_reallipsum
86 }\_lipsum
87 }
88 \_def\_negativermmn{\_romannumeral-`.}
89 \_def\_reallipsum[#1]{\_lipsumA #1\_empty\_empty\_end}
90 \_def\_lipsumA #1-#2\_empty#3\_end{\_tmpnum=#1 \_edef\_tmp{\_ifx`#2`#1\_else#2\_fi}%
91   \_loop \_csname lips:\_the\_tmpnum\_endcsname \_par % \_par is better here
92     \_ifnum\_tmpnum<\_tmp \_advance\_tmpnum by1 \_repeat
93 }
94 \def\lipsum {\_lipsum}
95 \def\lorem {\_lipsum}
```

## 2.38 Printing documentation

The `\printdoc ⟨filename⟩⟨space⟩` and `\printdoctail ⟨filename⟩⟨space⟩` commands are defined after the file `doc.opm` is load by `\load [doc]`.

The `\printcoc` starts reading of given `⟨filename⟩` from the second line. The file is read in *the listing mode*. The `\prindoctail` starts reading given `⟨filename⟩` from the first occurrence of the `\_endcode`. The file is read in normal mode (like `\input ⟨filename⟩`).

The *listing mode* prints the lines as listing of a code. This mode is finished when first `␣␣\_doc` occurs or first `\_endcode` occurs. At least two spaces must precede before such `\_doc`. On the other hand, the `\_endcode` must be at the left edge of the line without spaces. If this rule is not met then the listing mode continues.

If the first line or the last line of the listing mode is empty then such lines are not printed. The maximal number of printed lines in the listing mode is `\maxlines`. Is is set to almost infinity (100000). You can set it to a more sensible value. Such setting is valid only for the first following listing mode.

When the listing mode is finished by `\_doc` then next lines are read in the normal way, but the material between `\begtt ... \endtt` pair is shifted by three letters left. The reason is that the three spaces of indentation is recommended in the `\_doc ... \_cod` pair and this shifting is a compensation of this indentation.

The `\_cod` macro ignores the rest of current line and starts the listing mode again.

When the listing mode is finished by the `\_endcode` then the `\endinput` is applied, the reading of the file opened by `\printdoc` is finished.

You cannot reach the end of the file (without `\_endcode`) in the listing mode.

The listing mode creates all control sequences which are listed in the index as active link to the main documentation point of such control sequence and prints them in blue. Other text is printed in black.

The main documentation point is denoted by `\`⟨sequence⟩`` in red, for example `\`foo``. The user documentation point is the first occurrence of `\^⟨sequence⟩^`, for example `\^foo^`. There can be more such markups, all of them are hyperlinks to the main documentation point. And main documentation point is hyperlink to the user documentation point, if such point exists. Finally, the `\~⟨sequence⟩`` (for example `\~foo``) are hyperlinks to the user documentation point.

```
3 \_codedecl \printdoc {Macros for documentation printing <2020-04-28>}
```

General decalarations.

```
9 \_fontfam[lmfonts]
10 \_hyperlinks \Green \Green
11 \_enlang
12 \_enquotes
```

Maybe, somebody needs \seccc or \secccc?

```
18 \_eoldef\seccc#1{\_medskip \_noindent{\_bf#1}\_par\_nobreak\_firstnoindent}
19 \_def\secccc{\_medskip\_noindent $\_bullet$ }
```

\enddocument can be redefined.

```
25 \_let\enddocument=\_bye
```

Full page of listing causes underfill \vbox in output routine. We need to add a small tolerance.

```
32 \_pgbottomskip=0pt plus10pt minus2pt
```

The listing mode is implemented here. The \maxlines is maximal lines of code printed in the listing mode.

```
39 \_newcount \_maxlines \_maxlines=100000
40 \_public \_maxlines ;
41
42 \_eoldef\_cod#1{\_par \_wipepar
43 \_vskip\_parskip \_medskip \_ttskip
44 \_begingroup
45 \_typosize[8/10]
46 \_let\_printverblin=\_printcodeline
47 \_ttline=\_inputlineno
48 \_setverb
49 \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
50 \_adef{ }{\ }\_adef\^I{t}\_parindent=\_ttindent \_parskip=0pt
51 \_relax \_ttfont
52 \_endlinechar=^^J
53 \_def\_tmpb{\_start}%
54 \_readverblin
55 }
56 \_def\_readverblin #1^^J{%
57 \_def\_tmpa{\_empty#1}%
58 \_let\_next=\_readverblin
59 \_ea\_isinlist\_ea\_tmpa\_ea{\_Doc}\_iftrue \_let\_next=\_processinput \_fi
60 \_ea\_isinlist\_ea\_tmpa\_ea{\_Endcode}\_iftrue \_endinput \_let\_next=\_processinput \_fi
61 \_ifx\_next\_readverblin \_addto\_tmpb{#1^^J}\_fi
62 \_next
63 }
64 {\_catcode\ =13 \_gdef\_aspace{ }}\_def\_asp{\_ea\_noexpand\_aspace}
65 \_edef\_Doc{\_asp\_asp\_bslash\_doc}
66 \_edef\_Endcode{\_noexpand\_empty\_bslash\_endcode}
```

The scanner of the control sequences in the listing mode.

```
72 \_def\_makecs{\_def\_tmp{\_futurelet\_next\_makecsA}
73 \_def\_makecsA{\_ifcat a\_noexpand\_next \_ea\_makecsB \_else \_ea\_makecsF \_fi}
74 \_def\_makecsB#1{\_addto\_tmp{#1}\_futurelet\_next\_makecsA}
75 \_def\_makecsF{\_ifx\_tmp\_empty \_csstring\\%
76 \_else \_ifcsname \_tmp\_endcsname \_link[cs:\_tmp]{\Blue}{\_csstring\\\_tmp}%
77 \_else \_let\_next=\_tmp \_remfirstunderscore\_next
78 \_ifx\_next\_empty \_let\_next=\_tmp \_fi
79 \_ifcsname \_next\_endcsname \_link[cs:\_next]{\Blue}{\_csstring\\\_tmp}%
80 \_else \_csstring\\\_tmp \_fi\_fi\_fi
81 }
82 \_def\_processinput{%
83 \_let\_start=\_relax
84 \_ea\_replstring\_ea\_tmpb\_ea{\_aspace^^J}{^^J}
85 \_addto\_tmpb{\_end}%
86 \_isinlist\_tmpb{\_start^^J}\_iftrue \_advance\_ttline by1\_fi
```

```

87 \_replstring\_tmpb{\_start^^J}{\_start}%
88 \_replstring\_tmpb{\_start}{}%
89 \_replstring\_tmpb{^^J\_end}{\_end}%
90 \_replstring\_tmpb{^^J\_end}{}%
91 \_replstring\_tmpb{\_end}{}%
92 \_ea\_prepareverbdata\_ea\_tmpb\_ea{\_tmpb^^J}%
93 \_replthis{\_csstring\\}{\_noexpand\_makecs}%
94 \_ea\_printverb \_tmpb\_end
95 \_par
96 \_endgroup \_ttskip
97 \_isnextchar\_par{}{\_noindent}%
98 }

```

The lines in the listing mode have Yellow background.

doc.opm

```

104 \_def\Yellow{\_setcmykcolor{0.0 0.0 0.3 0.03}}
105
106 \_def\_printcodeline#1{\_advance \_maxlines by-1
107 \_ifnum \_maxlines<0 \_ea \_endverbprinting \_fi
108 \_penalty \_ttpenalty \_kern-4pt
109 \_noindent\_rlap{\Yellow \_vrule height8pt depth5pt width\_hsize}%
110 \_printfilename
111 \_indent \_printverblinenum #1\par}
112
113 \_def\_printfilename{\_hbox to0pt{%
114 \_hskip\_hsize\_vbox to0pt{\_vss\_llap{\Brown\docfile}\_kern7.5pt}\_hss}%
115 \_let\_printfilename=\_relax
116 }
117 \_everytt={\_let\_printverblinenum=\_relax}
118
119 \_long\_def\_endverbprinting#1\_end{\_fi \_global\_maxlines=100000
120 \_noindent\_typo\size[8/]\_dots etc. (see {\_tt\Brown\docfile})}

```

\docfile is currently documented file.

\printdoc and \printdoctail macros are defined here.

doc.opm

```

127 \_def\docfile{}
128 \_def\_printdoc #1 {\_par \_def\docfile{#1}%
129 \_everytt={\_ttshift=-15pt \_let\_printverblinenum=\_relax}%
130 \_ea\_cod \input #1
131 \_everytt={\_let\_printverblinenum=\_relax}%
132 \_def\docfile{}%
133 }
134 \_def\_printdoctail #1 {\_bgroup
135 \_everytt={}\_ttline=-1 \_ea\_printdoctailA \_input #1 \_egroup}
136 {\_long\_gdef\_printdoctailA#1\_endcode{}}
137
138 \_public \printdoc \printdoctail ;

```

You can do \verbinuput \vitt{<filename>} (<from>-<to>) <filename> if you need analogical design like in listing mode.

doc.opm

```

145 \_def\_vitt#1{\_def\docfile{#1}\_ttline=-1
146 \_everytt={\_typo\size[8/10]\_let\_printverblinenum=\_printcodeline \_medskip}}
147
148 \_public \vitt ;

```

The Index entries are without the trailing backslash. We must to add it when printing Index.

doc.opm

```

155 \_addto \_ignoredcharsen {\_} % \foo, \_foo is the same in the first pass of sorting
156 \_def\_printii #1#2&{%
157 \_ismacro\_lastii{#1}\_iffalse \_newiiletter{#1}{#2}\_def\_lastii{#1}\_fi
158 \_gdef\_currii{#1#2}\_the\_everyii\_noindent
159 \_hskip-\_iindent \_ignorespaces\_printiiA\bslash#1#2//}
160
161 \_def\_printiipages#1&{\_let\_pgtype=\_undefined \_tmpnum=0
162 {\_rm\_printpages #1,.,\_par}}
163
164 \_sdef{\_toc1:1}#1#2#3{\_nofirst\_bigskip
165 \_bf\_llaptoclink{#1}{#2}\_hfill \_pgn{#3}\_tocpar\_medskip}

```

The <something> will be print as  $\langle something \rangle$ .

doc.opm

```
171 \let\lt=<
172 \_catcode`\<=13
173
174 \_def<#1>{\langle\hbox{\it#1/}\rangle$}
175 \_everyintt{\_catcode`\<=13 }
```

If this macro is loaded by `\load` then we need to initialize catcodes using the `\_afteroad` macro.

doc.opm

```
182 \_def\_afterload{\_catcode`\<=13 \_catcode`\`=13 }
```

Main documentation point and hyperlinks to/from it.

doc.opm

```
188 \_activettchar`
189
190 \_def\`#1`{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
191 \_ifcsname cs:\_tmp\_endcsname\_else \_dest[cs:\_tmp]\_fi
192 \_sxdef{cs:\_tmp}{}%
193 \_hbox{\_ifcsname cs:\_tmp\_endcsname
194 \_link[cs:\_tmp]{\Red}{\_tt\_csstring\\\_tmp}\_else
195 {\_tt\Red\_csstring\\\_tmp}\_fi}%
196 }
197
198 \_def\~`#1`{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
199 \_hbox{\_ifcsname cs:\_tmp\_endcsname \_else \_dest[cs:\_tmp]\_sxdef{cs:\_tmp}{}\_fi
200 \_link[cs:\_tmp]{\Blue}{\_tt\_string#1}}%
201 \_futurelet\_next\_cslinkA
202 }
203 \_def\_cslinkA{\_ifx\_next`\_ea\_ignoreit \_else \_ea\_ea\_ea`\_ea\_string\_fi}
204
205 \_def\~`#1`{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
206 \_hbox{\_link[cs:\_tmp]{\Blue}{\_tt\_string#1}}%
207 \_futurelet\_next\_cslinkA
208 }
```



## Index

`\_aboveliskip` 112  
`\_abovetitle` 108, 110  
`\activequotes` 167  
`\activettchar` 16–17, 114  
`\_addcolor` 98  
`\_additcorr` 90  
`\address` 24–25, 158  
`\_addtabitemx` 128  
`\addto` 35, 91  
`\adef` 16–17, 35  
`\advancepageno` 91–92  
`\_afteritcorr` 90  
`\_afterload` 48  
`\_allocator` 37  
`\allowbreak` 51  
`\altquotes` 166–167  
`\_asciisortingtrue` 153  
`\_athe` 113  
`\b` 52  
`\_backgroundbox` 91  
`\backgroundpic` 122  
`\bbchar` 70, 84  
`\begitems` 13–14, 112–113  
`\begmulti` 19, 131  
`\_begoutput` 91, 105  
`\begtt` 16–17, 114  
`\_belowliskip` 112  
`\_belowtitle` 108, 110  
`\bf` 8–9, 70, 84  
`\bgroup` 35  
`\bi` 8–9, 70, 84  
`\bib` 20, 134  
`\bibmark` 132  
`\bibnum` 102, 132  
`\_bibskip` 135  
`\bibtexhook` 45  
`\big` 75  
`\Big` 75  
`\bigbreak` 51  
`\bigg` 75  
`\Bigg` 75  
`\biggl` 75  
`\Biggl` 75  
`\biggm` 75  
`\Biggm` 75  
`\biggr` 75  
`\Biggr` 75  
`\bigl` 75  
`\Bigl` 75  
`\bigm` 75  
`\Bigm` 75  
`\bigr` 75  
`\Bigr` 75  
`\bigskip` 50  
`\Black` 96  
`\Blue` 21, 96  
`\bmod` 77  
`\boldify` 90  
`\boldmath` 9, 70, 72, 82  
`\_boldunimath` 82  
`\bordermatrix` 78  
`\boxlines` 158  
`\bp` 49  
`\_bp` 49  
`\_bprinta` 138  
`\_bprintb` 138  
`\_bprintc` 138  
`\_bprintv` 138  
`\bracedparam` 48  
`\break` 51  
`\Brown` 96  
`\bslash` 35  
`\buildrel` 77  
`\bye` 53  
`\c` 52  
`\cal` 70, 84  
`\caption` 10–11, 111  
`\cases` 78  
`\catalogexclude` 69  
`\catalogmathsample` 69  
`\catalogonly` 69  
`\catalogsample` 69  
`\catcode` 49  
`\cdots` 76  
`\centerline` 51  
`\chap` 10–11, 17–18, 109  
`\_chapfont` 107  
`\_chapx` 108  
`\chyph` 24, 167  
`\_circle` 122–123  
`\circleparams` 46  
`\cite` 19–20, 132  
`\_citeA` 133  
`\citeborder` 12, 103  
`\clipincircle` 23, 125  
`\clipinoval` 23, 125  
`\_clipinpath` 125  
`\clqq` 167  
`\cmymcolordef` 97  
`\_cmymktoRGB` 96–97  
`\_cod` 31, 49  
`\code` 16–17, 114  
`\_codedecl` 31–32  
`\colnum` 128  
`\_colorcrop` 97  
`\colordef` 21, 95–97, 99  
`\_colordefFin` 97  
`\_colorstackpop` 96  
`\_colorstackpush` 96  
`\_colorstackset` 96  
`\colsep` 44  
`\_commoncolordef` 97  
`\_completepage` 91  
`\_compoundchars` 151  
`\_compoundcharscs` 151  
`\_compoundcharsen` 151  
`\cong` 77  
`\_corrmsizes` 72  
`\crl` 15, 129  
`\_crli` 15, 126–129  
`\_crl1` 129  
`\_crl1i` 15, 126, 129  
`\_crlp` 15, 126, 129  
`\crqq` 167  
`\cs` 35  
`\CS` 162  
`\cskip` 10, 111  
`\cslang` 23–24, 163  
`\csplain` 162  
`\csquotes` 24, 167  
`\_ctablelist` 47  
`\_currfamily` 63, 66  
`\_currpage` 101, 104, 167  
`\currstyle` 80  
`\_currV` 62, 66  
`\currvar` 8–9, 57, 59–61, 67  
`\Cyan` 21, 96  
`\d` 52  
`\_ddlinedata` 128  
`\ddots` 76  
`\_decdigits` 49  
`\_defaultfontfeatures` 69  
`\defaultitem` 14, 44, 113  
`\delang` 23, 163  
`\dequotes` 24, 167  
`\dest` 12, 102  
`\_destactive` 102  
`\_destheight` 102  
`\displaylines` 79  
`\do` 38  
`\_do` 38  
`\_doc` 31, 49  
`\_docompound` 152  
`\doloadmath` 81  
`\_doresizefont` 56, 65  
`\_doresizetfmfont` 56  
`\_doresizeunifont` 56, 65, 69  
`\_dosshadow` 124  
`\_dosorting` 153  
`\dospecials` 49  
`\dosupereject` 51, 91  
`\doteq` 77  
`\dotfill` 53  
`\dots` 52  
`\_douseK` 97  
`\_doverbininput` 115  
`\_dowhichtfm` 57  
`\draft` 7, 93

<code>\ea</code> 32	<code>\fnotemark</code> 7, 157	<code>\hphantom</code> 77
<code>\_ea</code> 32	<code>\fnotenun</code> 156	<code>\hrulefill</code> 53
<code>\ecite</code> 20, 132	<code>\fnotenunchapters</code> 7, 109, 156	<code>\hyperlinks</code> 12–13, 20, 102–103, 108
<code>\egroup</code> 35	<code>\fnotenumglobal</code> 7, 156	<code>\ialign</code> 51
<code>\ehyph</code> 24, 167	<code>\fnotenumpages</code> 7, 156, 161	<code>\_ifAleB</code> 152
<code>\eject</code> 51	<code>\_fnotestack</code> 97	<code>\_ifexistfam</code> 40, 62
<code>\em</code> 8, 90	<code>\fnotetext</code> 7, 157	<code>\_iflocalcolor</code> 95
<code>\empty</code> 35	<code>\_fnset</code> 157	<code>\_ifmathloading</code> 81
<code>\_endcode</code> 31–32	<code>\fntborder</code> 12, 103	<code>\_ifmathsb</code> 73
<code>\endgraf</code> 50	<code>\folio</code> 25, 92	<code>\_ifpgfnote</code> 156
<code>\endinsert</code> 11, 93	<code>\fontdef</code> 55–56, 58–60, 67	<code>\_ignoredchars</code> 151
<code>\enditems</code> 13, 112	<code>\fontfam</code> 5, 7, 9, 27, 55, 59, 61, 64, 68–70	<code>\ignoreit</code> 49
<code>\endline</code> 50	<code>\_fontfeatures</code> 63, 69	<code>\ignorept</code> 49
<code>\endmulti</code> 19, 131	<code>\fontlet</code> 55–59	<code>\ignoreslash</code> 162
<code>\_endoutput</code> 91	<code>\_fontnamegen</code> 62–63, 65–66	<code>\ii</code> 18–19, 156
<code>\endtt</code> 16–17, 114	<code>\_fontselector</code> 56	<code>\iid</code> 18–19, 156
<code>\enlang</code> 24, 163	<code>\footins</code> 91–92, 157	<code>\iindent</code> 14, 44
<code>\enquotes</code> 24, 167	<code>\footline</code> 6, 46, 91–92	<code>\iindex</code> 155–156
<code>\enskip</code> 50	<code>\footlinedist</code> 6, 46	<code>\iis</code> 19, 156
<code>\enspace</code> 50	<code>\footnote</code> 7, 91–92	<code>\iitype</code> 19, 156
<code>\_ensureblack</code> 91–92	<code>\_footnoterule</code> 91–92	<code>\_iitypesaved</code> 156
<code>\eoldef</code> 48	<code>\footstrut</code> 92	<code>\ilevel</code> 14, 44
<code>\eqalign</code> 9, 46, 78	<code>\foreach</code> 38	<code>\ilink</code> 13, 102
<code>\eqalignno</code> 10, 79	<code>\_foreach</code> 38	<code>\_inchap</code> 109
<code>\eqlines</code> 46, 79	<code>\_forlevel</code> 39	<code>\incircle</code> 23, 123
<code>\eqmark</code> 10–11, 111	<code>\_formatcmyk</code> 96	<code>\ingnslash</code> 162
<code>\eqspace</code> 46, 79	<code>\_formatgrey</code> 96	<code>\_initfontfamily</code> 63, 66
<code>\eqstyle</code> 46, 79	<code>\_formatrgb</code> 96	<code>\initunifonts</code> 65–66
<code>\everyii</code> 45, 154	<code>\fornum</code> 38	<code>\_inkdefs</code> 120
<code>\everyintt</code> 17, 43	<code>\_fornum</code> 38	<code>\inkinspic</code> 22, 120
<code>\everyitem</code> 44	<code>\fornumstep</code> 38	<code>\_inmath</code> 85
<code>\everylist</code> 14, 44	<code>\fR</code> 14, 129	<code>\inoval</code> 23, 123
<code>\everymnote</code> 45	<code>\frak</code> 70, 84	<code>\_inputref</code> 100
<code>\everytable</code> 45, 126	<code>\frame</code> 15, 22, 130	<code>\_insec</code> 109
<code>\everytocline</code> 45, 104	<code>\frqq</code> 167	<code>\_insecc</code> 109
<code>\everytt</code> 16–17, 43	<code>\frquotes</code> 167	<code>\_insertmark</code> 110
<code>\expr</code> 49	<code>\fS</code> 14, 127, 129	<code>\insertoutline</code> 13, 105
<code>\_expr</code> 49	<code>\fset</code> 66	<code>\_insertshadowresources</code> 124
<code>\_famalias</code> 65, 68	<code>\_fullrectangle</code> 113	<code>\inspic</code> 21–22, 120
<code>\_famdecl</code> 62–63, 66	<code>\_fvars</code> 62, 66	<code>\_inspicA</code> 120
<code>\_famdepend</code> 67	<code>\fX</code> 14, 129	<code>\_inspicB</code> 120
<code>\_faminfo</code> 65, 68–69	<code>\_getforstack</code> 39	<code>\_interliskip</code> 112
<code>\_famtext</code> 65, 68	<code>\_gfnotenun</code> 156	<code>\_isAleB</code> 152
<code>\famvardef</code> 58, 60, 62–63, 66–67	<code>\goodbreak</code> 51	<code>\isdefined</code> 40
<code>\_famvardefA</code> 67	<code>\gpageno</code> 91, 102	<code>\isempty</code> 40
<code>\fC</code> 14, 129	<code>\_greekdef</code> 83	<code>\isequal</code> 40
<code>\fcolor</code> 123	<code>\Green</code> 96	<code>\isfile</code> 40
<code>\filbreak</code> 51	<code>\Grey</code> 96	<code>\isfont</code> 40
<code>\_fillstroke</code> 96, 123	<code>\headline</code> 6, 46, 91–92	<code>\isinlist</code> 40
<code>\_firstnoindent</code> 10, 108, 110	<code>\headlinedist</code> 6, 46, 92	<code>\ismacro</code> 40
<code>\fixmnotes</code> 7, 157	<code>\hglue</code> 50	<code>\isnextchar</code> 40
<code>\fL</code> 14, 129	<code>\hhkern</code> 45	<code>\istokseempty</code> 40
<code>\flqq</code> 167	<code>\hicolor</code> 119	<code>\it</code> 8, 70, 84
<code>\fmtname</code> 28	<code>\hicolors</code> 44	<code>\item</code> 51
<code>\fnfborder</code> 12, 103	<code>\hidewidth</code> 51	<code>\itemitem</code> 51
<code>\fnote</code> 7, 91, 157	<code>\hisyntax</code> 17, 115, 117, 119	<code>\itemnum</code> 112
<code>\fnotelinks</code> 12, 157		

<code>\jointrel</code> 75	<code>\mathhexbox</code> 52	<code>\noloadmath</code> 9, 59, 81
<code>\_keepmeaning</code> 56	<code>\_mathloadingfalse</code> 81	<code>\nonfrenchspacing</code> 42, 164
<code>\label</code> 11–12, 101, 108	<code>\_mathloadingtrue</code> 81	<code>\nonum</code> 10, 108–109
<code>\langlist</code> 23, 165	<code>\mathpalette</code> 77	<code>\nonumcitations</code> 19–20, 133
<code>\_langw</code> 24, 166	<code>\mathsboff</code> 30, 72	<code>\nopagenumbers</code> 6, 92
<code>\lastpage</code> 25, 167	<code>\mathsbon</code> 30, 72	<code>\normalbottom</code> 92
<code>\LaTeX</code> 162	<code>\mathstrut</code> 77	<code>\normalcatcodes</code> 47
<code>\layernum</code> 160	<code>\mathstyles</code> 9, 80	<code>\normalmath</code> 9, 70, 72, 82, 88
<code>\layers</code> 160–161	<code>\matrix</code> 78	<code>\_normalunimath</code> 82
<code>\_layertext</code> 160	<code>\maxlines</code> 168–169	<code>\_nospaceafter</code> 48
<code>\lcolor</code> 123	<code>\_maybetod</code> 146	<code>\nospec</code> 23, 122
<code>\ldots</code> 76	<code>\medbreak</code> 51	<code>\not</code> 80, 88
<code>\leavevmode</code> 51	<code>\medskip</code> 50	<code>\notin</code> 77
<code>\leftarrowfill</code> 53	<code>\_mergesort</code> 153	<code>\notoc</code> 10, 109
<code>\leftline</code> 51	<code>\_mfontfeatures</code> 82	<code>\novspaces</code> 14, 113
<code>\letfont</code> 167	<code>\midinsert</code> 11, 93	<code>\null</code> 35
<code>\letter</code> 24, 158	<code>\mit</code> 70	<code>\numberedpar</code> 11, 111
<code>\_lfnotenum</code> 156	<code>\mnote</code> 7, 157	<code>\obeylines</code> 50
<code>\LightGrey</code> 96	<code>\_mnoteA</code> 158	<code>\obeyspaces</code> 50
<code>\line</code> 51	<code>\mnoteindent</code> 45	<code>\offinterlineskip</code> 50
<code>\link</code> 102	<code>\_mnotesfixed</code> 157	<code>\oldaccents</code> 27, 52
<code>\_linkactive</code> 102–103	<code>\mnotesize</code> 7, 45	<code>\onlycmyk</code> 21, 95–96
<code>\lipsum</code> 25, 168	<code>\mnoteskip</code> 7, 45	<code>\_onlyif</code> 66
<code>\_listfamnames</code> 68	<code>\moddef</code> 58, 62–63, 66–67	<code>\onlyrgb</code> 21, 95–96
<code>\_listskipA</code> 112	<code>\morecolors</code> 21, 99	<code>\oalign</code> 52
<code>\listskipamount</code> 44, 113	<code>\mspan</code> 15, 130	<code>\_openfnostack</code> 97
<code>\_listskipB</code> 112	<code>\_mtext</code> 166	<code>\_openfnostackA</code> 97
<code>\llap</code> 51	<code>\multispan</code> 15, 51	<code>\openref</code> 91, 100
<code>\_llaptoclink</code> 104	<code>\_mv</code> 122	<code>\_openrefA</code> 100
<code>\load</code> 25, 31, 48, 168, 171	<code>\_namespace</code> 31–32	<code>\openup</code> 79
<code>\loadboldmath</code> 81–82	<code>\narrower</code> 51	<code>\_opfootnote</code> 92, 157
<code>\loadmath</code> 9, 59, 81, 83	<code>\_narrowlastlinecentered</code> 111	<code>\opinput</code> 47
<code>\_loadmathfamily</code> 71	<code>\nbb</code> 35	<code>\OPmac</code> 162
<code>\_loadpattrs</code> 164	<code>\nbpar</code> 108, 110	<code>\opt</code> 48
<code>\_loadumathfamily</code> 82	<code>\negthinspace</code> 50	<code>\optdef</code> 48
<code>\localcolor</code> 95	<code>\newattribute</code> 37	<code>\OpTeX</code> 162
<code>\loggingall</code> 36	<code>\newbox</code> 37	<code>\optexcatcodes</code> 47
<code>\loop</code> 38	<code>\newcatodetable</code> 37	<code>\_optexoutput</code> 91
<code>\_loop</code> 38	<code>\newcount</code> 37	<code>\optexversion</code> 28
<code>\lorem</code> 25, 168	<code>\newcurrfontsize</code> 57, 90	<code>\_optfontalias</code> 64, 66
<code>\_lrmnote</code> 158	<code>\newdimen</code> 37	<code>\_optname</code> 64, 66
<code>\LuaTeX</code> 162	<code>\newfam</code> 37	<code>\_optnameA</code> 66
<code>\linewidth</code> 123	<code>\newif</code> 31, 38	<code>\_optsize</code> 56
<code>\Magenta</code> 96	<code>\_newifi</code> 31, 38	<code>\opwarning</code> 35
<code>\magnification</code> 53	<code>\_newiiletter</code> 154	<code>\_othe</code> 109
<code>\magyscale</code> 6, 94	<code>\newinsert</code> 37	<code>\outlines</code> 13, 105
<code>\magstep</code> 50	<code>\newmuskip</code> 37	<code>\_outlinesA</code> 105
<code>\magstephalf</code> 50	<code>\newread</code> 37	<code>\_outlinesB</code> 105
<code>\mainbaselineskip</code> 8, 89	<code>\newskip</code> 37	<code>\_oval</code> 122–123
<code>\_mainfamcommand</code> 66	<code>\newtoks</code> 37	<code>\ovalparams</code> 23, 46
<code>\mainfosize</code> 8, 89	<code>\newwrite</code> 37	<code>\overbrace</code> 76
<code>\_makefootline</code> 92	<code>\nextpages</code> 46	<code>\overlapmargins</code> 123
<code>\_makeheadline</code> 91–92	<code>\nl</code> 10, 110	<code>\overleftarrow</code> 76
<code>\makeindex</code> 18–19, 24, 150, 154	<code>\nobreak</code> 51	<code>\overrightarrow</code> 76
<code>\maketoc</code> 18, 105, 108	<code>\nocite</code> 20, 132	<code>\_pagecontents</code> 91–92
<code>\margins</code> 5–6, 27, 91, 93	<code>\_nofirst</code> 104	<code>\_pagedest</code> 91–92
<code>\mathbox</code> 9, 80	<code>\nointerlineskip</code> 50	<code>\pageinsert</code> 93
		<code>\pageno</code> 25, 91–92

<code>\paramtabdeclarep</code> 129	<code>\qqquad</code> 50	<code>\_seccx</code> 108
<code>\pcent</code> 35	<code>\quad</code> 50	<code>\_secfont</code> 107
<code>\_pdfborder</code> 103	<code>\quoteschars</code> 166	<code>\_sectionlevel</code> 108
<code>\pdfrotate</code> 22, 121	<code>\raggedbottom</code> 92	<code>\_secx</code> 108
<code>\pdfscale</code> 22, 121	<code>\raggedright</code> 51	<code>\_setbaselineskip</code> 89
<code>\pdfunidef</code> 105–106	<code>\ratio</code> 23, 123	<code>\setcmkcolor</code> 21, 95–96
<code>\_pdfunidefB</code> 106	<code>\rcite</code> 19, 132	<code>\_setcolor</code> 96–97
<code>\pg</code> 160	<code>\_readverb</code> 114	<code>\setctable</code> 47
<code>\pgbackground</code> 7, 46, 91	<code>\Red</code> 21, 96	<code>\setff</code> 60–61, 63, 69
<code>\pgborder</code> 12, 103	<code>\ref</code> 11–12, 101	<code>\_setflcolor</code> 123
<code>\pgbottomskip</code> 46, 91–92	<code>\refborder</code> 12, 103	<code>\setfontcolor</code> 61, 63, 69
<code>\_pgn</code> 104	<code>\refdecl</code> 100	<code>\setfontsize</code> 9, 54–56, 58–61, 63, 89
<code>\_pgprint</code> 155	<code>\regmacro</code> 13, 18, 91, 105–106	<code>\setgreycolor</code> 95–96
<code>\pgref</code> 11–12, 101	<code>\_regmark</code> 91, 105	<code>\setletterspace</code> 61, 63, 69
<code>\phantom</code> 77	<code>\_regoptsizes</code> 55, 64, 66	<code>\_setlistskip</code> 112
<code>\picdir</code> 21, 43	<code>\_regoul</code> 105, 114	<code>\_setmainvalues</code> 89
<code>\picheight</code> 21, 43	<code>\_regtfm</code> 55, 57	<code>\_setmathdimens</code> 72, 82
<code>\_picparams</code> 120	<code>\_regtoc</code> 105	<code>\_setmathfamily</code> 71
<code>\picw</code> 21, 43	<code>\_reloading</code> 57	<code>\setmathsizes</code> 70–71, 82
<code>\picwidth</code> 21, 43	<code>\_remifirstunderscore</code> 67	<code>\_setprimarysorting</code> 152
<code>\plaintexcatocdes</code> 47	<code>\removelastskip</code> 51	<code>\setrgbcolor</code> 95–96
<code>\pllang</code> 23, 163	<code>\removespaces</code> 49	<code>\_setsecondarysorting</code> 152
<code>\pmatrix</code> 78	<code>\repeat</code> 38	<code>\_setunimathdimens</code> 82
<code>\pmod</code> 77	<code>\_repeat</code> 38	<code>\_setverb</code> 114
<code>\_preparesorting</code> 152	<code>\replfromto</code> 118	<code>\setwordspace</code> 61, 69
<code>\_prepareverbdata</code> 115, 119	<code>\replstring</code> 48, 98, 106, 127	<code>\setwsp</code> 69
<code>\preplang</code> 163	<code>\replthis</code> 119	<code>\shadow</code> 123
<code>\_prepoffsets</code> 91	<code>\report</code> 24, 158	<code>\_shadowb</code> 124
<code>\prime</code> 75	<code>\resetmod</code> 59, 62	<code>\shadowlevels</code> 124
<code>\_printbib</code> 135	<code>\_resetnonumnotoc</code> 109	<code>\_shadowmoveto</code> 124
<code>\_printcaptionf</code> 111	<code>\_resizefont</code> 56–57	<code>\shordcitations</code> 133
<code>\_printcaptiont</code> 111	<code>\resizethefont</code> 54, 57	<code>\shortcitations</code> 19, 133
<code>\_printchap</code> 10, 108	<code>\restorectable</code> 47	<code>\_showcolor</code> 99
<code>\printdoc</code> 168, 170	<code>\_reversetfm</code> 57	<code>\showlabels</code> 12, 102
<code>\printdoctail</code> 168, 170	<code>\_rfontskipat</code> 56–57	<code>\shyph</code> 24, 167
<code>\_printfnotemark</code> 156	<code>\rgbcolordef</code> 21, 95–97	<code>\_sizemscript</code> 71, 89
<code>\_printii</code> 154	<code>\_rgbtocmyk</code> 97	<code>\_sizemsscript</code> 71, 89
<code>\_printiipages</code> 154–155	<code>\rightarrowfill</code> 53	<code>\_sizemtext</code> 71, 89
<code>\_printindexitem</code> 154	<code>\rightleftharpoons</code> 77	<code>\_sizespec</code> 56–57, 63
<code>\_printinverbatim</code> 114	<code>\rightline</code> 51	<code>\skew</code> 76
<code>\_printitem</code> 113	<code>\rlap</code> 51	<code>\skiptoeol</code> 48
<code>\_printlabel</code> 102	<code>\rm</code> 8, 70, 84	<code>\sklang</code> 24, 163
<code>\_printnumberedpar</code> 112	<code>\rmfixed</code> 90	<code>\_slantcorr</code> 162
<code>\_printrefnum</code> 108, 110	<code>\rotbox</code> 22, 121	<code>\slash</code> 50
<code>\_printsavedcites</code> 133	<code>\rulewidth</code> 16, 130	<code>\slet</code> 35
<code>\_printsec</code> 10, 108, 160	<code>\_savedcites</code> 132–133	<code>\_slidelayer</code> 160
<code>\_printsecc</code> 10, 108	<code>\_savedttchar</code> 114	<code>\_slidepage</code> 160
<code>\_printverb</code> 115	<code>\_savedttcharc</code> 114	<code>\slides</code> 24–25, 159
<code>\_printverblin</code> 115	<code>\sb</code> 75	<code>\slideshow</code> 160–161
<code>\_printverblinenum</code> 115	<code>\scalebig</code> 75	<code>\smallbreak</code> 51
<code>\private</code> 30–32	<code>\scalemain</code> 8, 89	<code>\smallskip</code> 50
<code>\pshow</code> 160	<code>\scantabdata</code> 128, 130	<code>\smash</code> 77
<code>\ptmunit</code> 71	<code>\scantwodimens</code> 122	<code>\sortcitations</code> 19, 133
<code>\ptunit</code> 8, 71	<code>\script</code> 70, 84	<code>\_sortingdata</code> 151
<code>\public</code> 30–32	<code>\sdef</code> 14, 24, 35	<code>\_sortingdatacs</code> 151
<code>\_putforstack</code> 39	<code>\sec</code> 10–11, 17–18, 109	<code>\sp</code> 75
<code>\putpic</code> 23, 122	<code>\secc</code> 10–11, 18, 109	<code>\space</code> 35
<code>\puttext</code> 23, 122	<code>\_seccfont</code> 107	

<code>\_startitem</code> 112	<code>\_titfont</code> 107	<code>\_uselang</code> 164
<code>\_startverb</code> 114–115	<code>\titskip</code> 44	<code>\uselanguage</code> 24, 164
<code>\_stripzeros</code> 97	<code>\tocborder</code> 12, 103	<code>\useoptex</code> 26, 167
<code>\strutbox</code> 51, 89	<code>\_tocdotfill</code> 104	<code>\useOpTeX</code> 26, 167
<code>\style</code> 13, 113	<code>\_tocline</code> 103–104	<code>\uslang</code> 167
<code>\stylenum</code> 80	<code>\_toclist</code> 103, 105	<code>\uv</code> 167
<code>\subject</code> 24, 158	<code>\_tocpar</code> 104	<code>\vdots</code> 76
<code>\subtit</code> 159	<code>\tocrefnum</code> 102, 104	<code>\_verbatimcatcodes</code> 114
<code>\supereject</code> 51	<code>\today</code> 166	<code>\verbinput</code> 17, 115
<code>\sxdef</code> 35	<code>\topglue</code> 50	<code>\vfootnote</code> 92, 157
<code>\_tabdata</code> 128	<code>\topins</code> 91–93	<code>\vglue</code> 50
<code>\tabdeclarec</code> 16, 129	<code>\topinsert</code> 11, 91, 93	<code>\_vidolines</code> 115
<code>\tabdeclarel</code> 129	<code>\totalpages</code> 25, 167	<code>\_vifile</code> 113
<code>\tabdeclarer</code> 129	<code>\tracingall</code> 36	<code>\_viline</code> 113
<code>\tabiteml</code> 15, 45, 126	<code>\transformbox</code> 22, 121	<code>\_vinolines</code> 115
<code>\tabitemr</code> 15, 45, 126	<code>\trycs</code> 35	<code>\_viscanminus</code> 115
<code>\table</code> 14–15, 126	<code>\_tryloadfamslocal</code> 68	<code>\_viscanparameter</code> 115
<code>\_tableA</code> 127	<code>\tsize</code> 45, 126–127	<code>\visiblesp</code> 116
<code>\_tableB</code> 127	<code>\tskip</code> 15, 129	<code>\vphantom</code> 77
<code>\_tableC</code> 127	<code>\tt</code> 13, 70, 84	<code>\vspan</code> 16, 130
<code>\_tablew</code> 127	<code>\_ttfont</code> 113	<code>\vvkern</code> 45
<code>\_tableW</code> 127	<code>\ttindent</code> 16–17, 44	<code>\_whatresize</code> 56
<code>\tablinespace</code> 45, 127, 129	<code>\ttline</code> 16–17, 44	<code>\White</code> 96
<code>\tabskip1</code> 45, 126	<code>\_ttpenalty</code> 113	<code>\_wichtfm</code> 57
<code>\tabskipr</code> 45, 126	<code>\ttraggedright</code> 51	<code>\_wipeepar</code> 110
<code>\tabspaces</code> 44	<code>\ttshift</code> 44	<code>\wlabel</code> 101
<code>\tabstrut</code> 45, 127	<code>\_ttskip</code> 113	<code>\wlog</code> 35
<code>\tenbf</code> 54	<code>\typoscale</code> 8–9, 89	<code>\_wref</code> 100, 110
<code>\tenbi</code> 54	<code>\typosize</code> 8–9, 89–90	<code>\_writeXcite</code> 135
<code>\tenit</code> 54	<code>\ulink</code> 12–13, 102–103	<code>\wterm</code> 32
<code>\tenrm</code> 54	<code>\_umahrangegreek</code> 83	<code>\_wtotoc</code> 110
<code>\tentt</code> 54	<code>\_umahrangegreek</code> 83	<code>\xargs</code> 32
<code>\_testAleB</code> 152	<code>\_umathcharholes</code> 82	<code>\_Xbib</code> 133–134
<code>\TeX</code> 162	<code>\_umathrange</code> 82–83	<code>\_Xcite</code> 135
<code>\textindent</code> 51	<code>\underbar</code> 51	<code>\XeTeX</code> 162
<code>\_thechapnum</code> 108–109	<code>\underbrace</code> 76	<code>\_Xfnote</code> 157
<code>\_thednum</code> 109	<code>\_unimathboldfont</code> 82	<code>\_Xindex</code> 155
<code>\_thefnum</code> 109	<code>\_unimathfont</code> 81	<code>\_Xlabel</code> 101
<code>\thefontscale</code> 9, 90	<code>\_unsskip</code> 129	<code>\_Xmnote</code> 157
<code>\thefontsize</code> 9, 90	<code>\url</code> 12–13, 103	<code>\_Xpage</code> 101, 104, 157, 167
<code>\_theseccnum</code> 108–109	<code>\_urlactive</code> 102–103	<code>\Xrefversion</code> 100
<code>\_theseccnum</code> 108–109	<code>\urlborder</code> 12, 103	<code>\_xscan</code> 119
<code>\_thetnum</code> 109	<code>\_urlfont</code> 103	<code>\_xscanR</code> 119
<code>\thinspace</code> 50	<code>\usebib</code> 20, 135	<code>\_Xtoc</code> 103
<code>\thistable</code> 45, 126	<code>\_usedirectly</code> 52	<code>\Yellow</code> 21, 96
<code>\tit</code> 10, 107	<code>\useK</code> 95, 97, 99	<code>\_zerotabrule</code> 129