

# OpTeX

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

Version 1.02

*Petr Olšák, 2020, 2021*

<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 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 namespace of control sequences is separated from the internal namespace of OpTeX and primitives (`\foo` versus `\_foo`). The namespaces 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 to these macros here. This is a significant difference from L<sup>A</sup>TeX or ConTeXt, which is 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 do not create a new user level for controlling your document. We are using TeX directly in this case. 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 the user knows TeX basics. They are described in many books. You can see a short document [TeX in nutshell](#) too.

---

<sup>1</sup> 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 the 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 the document . . . . .	10
1.4.1	Chapters and sections . . . . .	10
1.4.2	Another numbered objects . . . . .	10
1.4.3	References . . . . .	12
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 . . . . .	20
1.6	Graphics . . . . .	21
1.6.1	Colors . . . . .	21
1.6.2	Images . . . . .	22
1.6.3	PDF transformations . . . . .	22
1.6.4	Ovals, circles . . . . .	23
1.6.5	Putting images and texts wherever . . . . .	24
1.7	Others . . . . .	24
1.7.1	Using more languages . . . . .	24
1.7.2	Pre-defined styles . . . . .	25
1.7.3	Loading other macro packages . . . . .	25
1.7.4	Lorem ipsum dolor sit . . . . .	26
1.7.5	Logos . . . . .	26
1.7.6	The last page . . . . .	26
1.7.7	Use OpTeX . . . . .	26
1.8	Summary . . . . .	26
1.9	API for macro writers . . . . .	27
1.10	Compatibility with Plain TeX . . . . .	29
<b>2</b>	<b>Technical documentation</b>	<b>30</b>
2.1	The main initialization file . . . . .	30
2.2	Concept of namespaces of control sequences . . . . .	32
2.2.1	Prefixing internal control sequences . . . . .	32
2.2.2	Namespace of control sequences for users . . . . .	32
2.2.3	Macro files syntax . . . . .	33
2.2.4	Name spaces for package writers . . . . .	33
2.2.5	Summary about rules for external macro files published for OpTeX . . . . .	33
2.2.6	The implementation of the namespaces . . . . .	34

2.3	pdfTeX initialization . . . . .	35
2.4	Basic macros . . . . .	37
2.5	Allocators for TeX registers . . . . .	39
2.6	If-macros, loops, is-macros . . . . .	41
2.6.1	Classical <code>\newif</code> . . . . .	41
2.6.2	Loops . . . . .	41
2.6.3	Is-macros . . . . .	43
2.7	Setting parameters . . . . .	44
2.7.1	Primitive registers . . . . .	45
2.7.2	Plain TeX registers . . . . .	45
2.7.3	Different settings than in plain TeX . . . . .	46
2.7.4	OpTeX parameters . . . . .	46
2.8	More OpTeX macros . . . . .	51
2.9	Using key=value format in parameters . . . . .	54
2.10	Plain TeX macros . . . . .	55
2.11	Preloaded fonts for text mode . . . . .	59
2.12	Scaling fonts in text mode (low-level macros) . . . . .	60
2.12.1	The <code>\setfontsize</code> macro . . . . .	60
2.12.2	The <code>\font</code> primitive . . . . .	60
2.12.3	The <code>\fontdef</code> declarator . . . . .	60
2.12.4	The <code>\fontlet</code> declarator . . . . .	61
2.12.5	Optical sizes . . . . .	61
2.12.6	Implementation notes . . . . .	61
2.13	The Font Selection System . . . . .	64
2.13.1	Terminology . . . . .	64
2.13.2	Font families, selecting fonts . . . . .	64
2.13.3	Math Fonts . . . . .	65
2.13.4	Declaring font commands . . . . .	65
2.13.5	The <code>\fontdef</code> declarator in detail . . . . .	66
2.13.6	The <code>\famvardef</code> declarator . . . . .	66
2.13.7	The <code>\tt</code> variant selector . . . . .	67
2.13.8	Font commands defined by <code>\def</code> . . . . .	67
2.13.9	Modifying font features . . . . .	67
2.13.10	Special font modifiers . . . . .	68
2.13.11	How to create the font family file . . . . .	68
2.13.12	How to write the font family file with optical sizes . . . . .	71
2.13.13	How to register the font family in the Font Selection System . . . . .	73
2.13.14	Notices about extension of <code>\font</code> primitive . . . . .	74
2.13.15	Implementation of the Font Selection System . . . . .	74
2.14	Preloaded fonts for math mode . . . . .	79
2.15	Math macros . . . . .	82
2.16	Unicode-math fonts . . . . .	90
2.16.1	Unicode-math macros preloaded in the format . . . . .	91
2.16.2	Macros and codes set when <code>\loadmatfont</code> is processed . . . . .	93
2.16.3	More Unicode-math examples . . . . .	99
2.16.4	Printing all Unicode math slots in used math font . . . . .	100
2.17	Scaling fonts in document (high-level macros) . . . . .	100
2.18	Output routine . . . . .	103
2.19	Margins . . . . .	106
2.20	Colors . . . . .	107
2.21	The <code>.ref</code> file . . . . .	112

2.22	References	113
2.23	Hyperlinks	115
2.24	Making table of contents	116
2.25	PDF outlines	118
2.25.1	Nesting PDF outlines	118
2.25.2	Strings in PDF outlines	119
2.26	Chapters, sections, subsections	121
2.27	Lists, items	126
2.28	Verbatim, listings	128
2.28.1	Inline and “display” verbatim	128
2.28.2	Listings with syntax highlighting	132
2.29	Graphics	136
2.30	The <code>\table</code> macro, tables and rules	141
2.30.1	The boundary declarator <code>:</code>	141
2.30.2	Usage of the <code>\tabskip</code> primitive	141
2.30.3	Tables to given width	142
2.30.4	<code>\eqbox</code> : boxes with equal width across the whole document	142
2.30.5	Implementation of the <code>\table</code> macro and friends	143
2.31	Balanced multi-columns	147
2.32	Citations, bibliography	149
2.32.1	Macros for citations and bibliography preloaded in the format	149
2.32.2	The <code>\usebib</code> command	152
2.32.3	Notes for bib-style writers	153
2.32.4	The <code>usebib.opm</code> macro file loaded when <code>\usebib</code> is used	154
2.32.5	Usage of the <code>bib-iso690</code> style	157
2.32.6	Implementation of the <code>bib-iso690</code> style	163
2.33	Sorting and making Index	167
2.34	Footnotes and marginal notes	173
2.35	Styles	175
2.35.1	<code>\report</code> and <code>\letter</code> styles	175
2.35.2	<code>\slides</code> style for presentations	176
2.36	Logos	180
2.37	Multilingual support	180
2.37.1	Lowercase, uppercase codes	180
2.37.2	Hyphenations	181
2.37.3	Multilingual phrases and quotation marks	184
2.38	Other macros	186
2.39	Lua code embedded to the format	188
2.40	Printing documentation	193
	<b>Index</b>	<b>197</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 the 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 other languages (for example `Ahoj světe!`) where Unicode fonts are needed because 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 a 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, a3, a3l, 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 format's dimensions stay 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 the review is done, the second line can be commented out.

## 1.2.2 Concept of the default page

OpTeX uses “output routine” for page design. It is very similar to the 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 into top insertions (floating tables and figures) followed by a real text and followed by footnotes. Typically, the only real text is here.

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

More about the page layout is documented in sections 2.7.4 and 2.18.

### 1.2.3 Footnotes and marginal notes

The Plain T<sub>E</sub>X'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 section 2.34.

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 a 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⟩` has to be 1 if only one such command is in the box. Second `\fnotemark` inside the same box has 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 T<sub>E</sub>X 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 using the up keyword: `\mnote up⟨dimen⟩{⟨text⟩}`. You can set such `⟨dimen⟩` to each `\mnote` manually in final printing in order to margin notes do not overlap. The positive value of `⟨dimen⟩` shifts the note up and negative value shifts it down. For example `\mnote up 2\baselineskip{⟨text⟩}` shifts this marginal 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 OpT<sub>E</sub>X 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 on how to register your own font family are appended in the 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 keep their internal setting until the group ends or until another modifier that negates the previous feature is used. So `{\caps \rm First text \it Second text}` gives `FIRST TEXT SECOND TEXT`.

The font modifier without following variant selector does not change the font actually, it only prepares data used by next variant selectors. 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 (emphasize text). It switches from current `\rm` to `\it`, from current `\it` to `\rm`, from current `\bf` to `\bi` and from current `\bi` to `\bf`. The italics correction `\/` is inserted automatically, 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 in the technical documentation in the section 2.13. 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 with respect to current values by default. If you want to do it with respect to the main values, type `\scalemain` immediately before `\typoscale` command.

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

The `\typosize` and `\typoscale` macros initialize the font family by `\rm`. You can re-size only the current font by the command `\thefontsize[⟨font-size⟩]` or the font 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 the 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; a 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 2.12.

### 1.3.3 Typesetting math

See the additional document [Typesetting Math with OpTeX](#) for more details about this issue.

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 a 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`, `\geq`, `\sum`, `\sphericalangle`, `\bumpeq`, `\simeq`. See AMSTeX manual or [Typesetting Math with OpTeX](#) for complete list of math 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	abc-xyz, ABC-XYZ
<code>\bbchar</code>	% double stroked letters	ABC-XYZ
<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 of the current text font family. This is a common notation for vectors and matrices. You can re-declare them, see section 2.16.2 where definitions of Unicode math variants of `\bf` and `\bi` selectors are documented.

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

## 1.4 Typical elements of the document

### 1.4.1 Chapters and sections

The documents 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⟩
```

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 sections have only one number and subsections two.

The implicit design of the titles of chapter etc. is 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 into more lines in the output. It is better to hint at the breakpoints because  $\TeX$  does not interpret the meaning of the title. Users can put the `\nl` (means newline) to the breakpoints.

If you want to arrange a title to more lines in your source file then you can use `^^J` at the end of each line (except the last one). When `^^J` is used, then the reading of the title continues at the next line. The “normal” comment character `%` doesn’t work in titles. You can use `\nl_^^J` if you want to have corresponding lines in the source and the output.

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 objects.

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 is reset 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 selected language (see section 1.7.1 about languages).

If you wish to make the table or figure as a 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

```

The pair `\midinsert... \endinsert` prefers to put the enclosed object to the current place. Only if this is unable due to page breaking, it behaves like `\topinsert... \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 1.4.1.** 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 1.4.2 (L'Hôpital's rule).** Let  $f$ ,  $g$  be...

<sup>3</sup> This feature can be changed, see the section 2.26 in the technical documentation.

### 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:

```
\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 users have to run  $\text{\TeX}$  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 with `\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 a 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 and texts generated by `\ref` or `\pgref`,
- numbers of chapters, sections, subsections, and page numbers 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*. Other 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 `\fntelinks`  $\langle color-fnt \rangle$   $\langle color-fnf \rangle$  where footnote marks in the text have  $\langle color-fnt \rangle$  and the same footnote marks in footnotes have  $\langle color-fnf \rangle$ . 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` [ $\langle type \rangle$  :  $\langle label \rangle$ ]. The active text linked to the `\dest` can be created by `\ilink` [ $\langle type \rangle$  :  $\langle label \rangle$ ] {  $\langle text \rangle$  }. The  $\langle type \rangle$  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 the `\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` [ $\langle url \rangle$ ] {  $\langle text \rangle$  } macro. For example: `\ulink{http://petr.olsak.net/optex}{\OpTeX/ page}` outputs to the text [OpTeX page](http://petr.olsak.net/optex). The characters %, \, #, {, and } must be escaped in the  $\langle url \rangle$  parameter.

The PDF format provides *outlines* which are notes placed in the special frame of the PDF viewer. These notes can be managed as a structured and hyperlinked table of contents of the document. The command `\outlines` {  $\langle level \rangle$  } creates such outlines from data used for the table of contents in the document. The  $\langle level \rangle$  parameter gives the level of opened sub-outlines in the default view. The deeper levels can be opened 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 a variant of the macro for outlines case which is expandable. Use `\regmacro` in this case. See the section 1.5.1 for more information about `\regmacro`.

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

The command `\thisoutline` {  $\langle text \rangle$  } uses  $\langle text \rangle$  in the outline instead of default title text for the first following `\chap`, `\sec`, or `\secc`. Special case: `\thisoutline{\relax}` doesn't create any outline for the following `\chap`, `\sec`, or `\secc`.

### 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
```

---

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

For example:

```
\beginitems
* First idea
* Second idea in subitems:
  \beginitems \style i
  * First sub-idea
  * Second sub-idea
  * Last sub-idea
  \enditems
* 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{<item:<style>}{<text>}`. Default item can be set by `\defaultitem={<text>}`. The list environments can be nested. Each new level of items 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 `\beginitems` starts `\everylist` tokens register. You can set, for example:

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

You can say `\beginitems \novspaces` if you don't want vertical spaces above and below the list. The nested item list is without vertical spaces automatically. More information about the design of lists of items should be found in the section 2.27.

A “selected block of text” can be surrounded by `\begblock... \endblock`. The default design of blocks of text is indented text in smaller font. The blocks of text can be nested.

### 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{||l|c|r||}{\cr
  Month      & commodity & price \cr
  January    & notebook  & \$ 700 \cr
  February   & skateboard & \$ 100 \cr
  July       & yacht      & k\$ 170 \cr}
```

generates the 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 a multiline 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 paragraph centered, long paragraph normal,
- `p{<size>\fX}` fit extra, left-right justified but last line centered.

You can use `<text>` in the `<declaration>`. Then this text is applied in each line of the table. For example `r{\kern10pt}l` adds more 10pt space between `r` and `l` rows.

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

The command `\cr` used in the `<data>` part of the table is generally known from Plain  $\text{\TeX}$ . It marks the end of each row in the table. Moreover  $\text{\OpTeX}$  defines following similar commands:

- `\crl` ... the end of the row with a horizontal line after it.
- `\crl1` ... the end of the row with a double horizontal line after it.
- `\crli` ... like `\crl` but the horizontal line doesn't intersect the vertical double lines.
- `\crl1i` ... like `\crli` but horizontal line is doubled.
- `\crlp{<list>}` ... like `\crli` but the lines are drawn only in the columns mentioned in comma-separated `<list>` of their numbers. The `<list>` can include `<from>-<to>` declarators, for example `\crlp{1-3,5}` is equal to `\crlp{1,2,3,5}`.

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

You can use the 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
\tabskip1=0pt \tabskipr=0pt % space before first and after last column
```

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

If there is an item that spans to more than one column in the table then the macro `\multispan{<number>}` (from Plain  $\text{\TeX}$ ) can help you. Another alternative is the command `\mspan{<number>}[<declaration>]{<text>}` which spans `<number>` columns and formats the `<text>` by the `<declaration>`. The `<declaration>` must include a declaration of only one column with the same syntax as common `\table <declaration>`. If your table includes vertical rules and you want to create continuous vertical rules by `\mspan`, then use rule declarators `|` after `c`, `l` or `r` letter in `\mspan <declaration>`. The exception is only in the case when `\mspan` includes the first column and the table have rules on the left side. The example of `\mspan` usage is below.

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

```
\frame{\table{|c||l||r|}{ \crl
  \mspan3[|c|]{\bf Title} \crl \noalign{\kern\hhkern}\crl1
  first & second & third \crl1i
  seven & eight & nine \crl1}}
```



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
}
```

You can use `\vspan` with non-integer parameter too if you feel that the result looks better, for example `\vspan2.1{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 2.30.5. See the definition of the `\tabdeclarec` macro, for example.

The `:` columns boundary declarator is described in section 2.30.1. The tables with given width can be declared by `to<size>` or `pxto<size>`. More about it is in section 2.30.3. Many tips about tables can be seen on the site <http://petr.olsak.net/optex/optex-tricks.html>.

### 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 `\verbchar<char>`. For example `\verbchar`` 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 the 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. Users can change the values of the `\parindent` and `\ttindent` independently.

The `\begtt` command starts the internal group in which the catcodes are changed. Then the `\everytt` tokens register is run. It is empty by default and the user can control fine behavior by

		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				

it. For example, the catcodes can be re-declared here. If you need to define an 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 `\verbchar` 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 a `\verbchar` doesn't work in parameter of macros and macro definitions. (It works in titles declared by `\chap`, `\sec` etc. and in `\fnotes`, because these macros are specially defined in OpTeX). You can use more robust command `\code{<text>}` in problematic situations, but you have to escape the following characters in the `<text>`: `\`, `#`, `%`, braces (if the braces are unmatched in the `<text>`), and space or `^` (if there are more than one subsequent spaces or `^` in the `<text>`). Examples:

```
\code{\\text, \%#} ... prints \text, %#
\code{@{...}*~$ $} ... prints @{...}*~$ $ without escaping, but you can
                        escape these characters too, if you want.
\code{a \ b}         ... two spaces between a b, the second 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 `\verbinput` command. Examples:

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

You can insert additional commands for `\verbinput` before the first opening bracket. They are processed in the local group. For example, `\verbinput \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 the default). If `\ttline<-1` then no line numbers are printed.

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

The `\begtt... \endtt` pair or `\verbatim` can be used for listings of codes. Automatic syntax highlighting is possible, for example `\begtt \hisyntax{C}` activates colors for C programs. Or `\verbatim \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 2.28.2.

If the code is read by `\verbatim` and there are comment lines prefixed by two characters then you can set them by `\commentchars⟨first⟩⟨second⟩`. Such comments are fully interpreted by TeX (i.e. not verbatim). Section 2.28.1 (page 131) says more about this feature.

## 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 the external `*.ref` file, so you have to run TeX more than once (typically three times if the table of contents is at the beginning of the document).

Typically, we don't want to repeat the name of the section "Table of contents" in the table of contents again. 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 like this:

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

If you need a customization of the design of the TOC, read the section 2.24.

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 its 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 PDF outlines
```

### 1.5.2 Making the index

The index can be included in the document by the `\makeindex` macro. No external program is needed, the alphabetical sorting is done inside TeX at macro level.

The `\ii` command (insert to index) declares the word separated by the space as the index item. This declaration is represented as an invisible item on the page connected to the next visible word. The page number of the page where this item 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 of \ii:
```

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

```
or:
```

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

Note that the dot or comma has 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 arranged 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 (separated by /) 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 of 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 the 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. The default index type is empty, which prints page numbers in normal font. The T<sub>E</sub>Xbook index is a 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 other 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 a complex index entry. Use “pure equivalent” in the `\ii` parameter and map this equivalent to a real word that is printed in the index. Such mapping is done by `\iis` command. Example:

```
The \ii chiquadrat  $\chi$ -quadrat method is ...
If the \ii relax \relax command is used then TEX/ is relaxing.
...
\iis chiquadrat  $\chi$ -quadrat
\iis relax {\code{\relax}}
```

The `\iis` *<equivalent>* *<text>* creates one entry in the “dictionary of the exceptions”. The sorting is done by the *<equivalent>* but the *<text>* is printed in the index entry list.

The sorting rules when `\makeindex` runs depends on the current language. See section 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 of 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]{$\sim\rcite[#1]$} % \cite[⟨label⟩] creates $\sim$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 BibT<sub>E</sub>X program) isn't supported because BibT<sub>E</sub>X is an old program that does not support Unicode. And Biber seems to be not compliant with Plain T<sub>E</sub>X.

#### 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:
\nocite[*] \usebib/s (simple) op-biblist % prints all from op-biblist.bib
```

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. OpT<sub>E</sub>X supports `simple` or `iso690` styles. The features of the `iso690` style is documented in the section 2.32.5 in detail. The `\usebib` command is more documented in section 2.32.2.

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 tells only that the mentioned bib-record should be printed in the reference list. If `\nocite[*]` is used then all records from `⟨bib-base⟩` are printed.

You can create more independent lists of references (you are creating proceedings, for example). Use `\bibpart {<name>}` to set the scope where `\cites` and references list are printed (and interconnected) independent of another parts of your document. The `\cite` labels used in different parts can be the same and they are not affected. References lists can be created manually by `\bib` or from a database by `\usebib`. Example:

```
\bibpart {AA}
... \cite[labelX] ... \cite[labelY] ... % They belong to AA bib-list
\usebib/c (simple) file.bib             % generates AA bib-list numbered 1, 2, ...
                                         % \cite prints [1], [2], ... by bib-list AA

\bibpart {BB}
... \cite[labelZ] ... \cite[labelX] ... % They belong to BB bib-list
\bibnum=0 \usebib/c (simple) my.bib     % generates BB bib-list numbered 1, 2, ...
                                         % \cite prints [1], [2], ... by bib-list BB
```

By default, `\bibpart` is empty. So `\cites` and the references list are connected using this empty internal name.

## 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 from the L<sup>A</sup>T<sub>E</sub>X file `x11nam.def`. There are 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 2.20 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. Thus, there may be a mix of color models in the PDF output which is not a good idea. You can overcome this problem by declaration `\onlyrgb` or `\onlycmyk`. Then only the selected color model is used for PDF output and if a 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

---

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



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 of `\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.

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  $\text{\TeX}$  to print labels in the same font as document font.  $\text{\OpTeX}$  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 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 “program” such pictures then Tikz package is recommended. It works in Plain  $\text{\TeX}$  and  $\text{\OpTeX}$ .

## 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 position of the current point. This position has to be the same from  $\text{\TeX}$ 's point of view as from the transformation point of view when `\pdfrestore` is processed. Due to this fact the `\pdfsave\rlap{<transformed text>}\pdfrestore` or something similar is recommended.

$\text{\OpTeX}$  provides two special transformation macros `\pdfscale` and `\pdfrotate`:

```
\pdfscale{<horizontal-factor>}{<vertical-factor>}
\pdfrotate{<angle-in-degrees>}
```

These macros simply call the properly `\pdfsetmatrix` command.

<sup>6</sup> A powerful and free Wysiwyg editor for creating vector graphics.

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



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: 


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. The `\transformbox{⟨transformation⟩}{⟨text⟩}` macro solves the problem. This macro puts the transformed material into a box with relevant dimensions. The `⟨transformation⟩` parameter includes one or more transformation commands `\pdfsetmatrix`, `\pdfscale`, `\pdfrotate` with their parameters. The `⟨text⟩` is transformed text.

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



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


#### 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
              \overlapmargins=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 \ignoremargins=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 `⟨width⟩` 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`  $\langle x \rangle$   $\langle y \rangle$   $\{\langle text \rangle\}$  puts the  $\langle text \rangle$  shifted by  $\langle x \rangle$  right and by  $\langle y \rangle$  up from the current point of typesetting and does not change the position of the current point. Assume a coordinate system with origin in the current point. Then `\puttext`  $\langle x \rangle$   $\langle y \rangle$   $\{\langle text \rangle\}$  puts the text at the coordinates  $\langle x \rangle$ ,  $\langle y \rangle$ . More exactly the left edge of its baseline is at that position.

The `\putpic`  $\langle x \rangle$   $\langle y \rangle$   $\langle width \rangle$   $\langle height \rangle$   $\{\langle image-file \rangle\}$  puts an image given by  $\langle image-file \rangle$  (including extension) of given  $\langle width \rangle$  and  $\langle height \rangle$  at given position (its left-bottom corner). You can write `\nospec` instead  $\langle width \rangle$  or  $\langle height \rangle$  if this parameter is not specified.

## 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 other languages are loaded on demand when you first use the `\langle iso-code \rangle lang` command in your document. For example `\delang` for German, `\cslang` for Czech, `\pllang` for Polish. The  $\langle iso-code \rangle$  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) hu(Hungarian) tr(Turkish)
et(Estonian) eu(Basque) ga(Irish) nb(Bokmal) nn(Nynorsk) nl(Dutch) pt(Portuguese) ro(Romanian) hr(Croatian)
zh(Pinyin) is(Icelandic) hsb(Uppersorbian) af(Afrikaans) gl(Galician) kmr(Kurmanji) tk(Turkmen) la(Latin) lac(classicLatin)
lal(liturgicalLatin) elm(monoGreek) elp(Greek) grc(ancientGreek) ca(Catalan) cop(Coptic) mn(Mongolian)
sa(Sanskrit) ru(Russian) uk(Ukrainian) hy(Armenian) as(Assamese) hi(Hindi) kn(Kannada) lv(Latvian) lt(Lithuanian)
ml(Malayalam) mr(Marathi) or(Oriya) pa(Panjabi) ta(Tamil) te(Telugu) be(Belarusian) bg(Bulgarian) bn(Bengali)
cu(churchslavonic) deo(oldGerman) gsw(swissGerman) eo(Esperanto) fur(Friulan) gu(Gujarati) ka(Georgian)
mk(Macedonian) oc(Occitan) pi(Pali) pms(Piedmontese) rm(Romansh) sr(Serbian) sv(Swedish) th(Thai) ethi(Ethiopic)
fis(schoolFinnish)
```

For compatibility with e-plain macros, there is the command `\uselanguage{\langle language \rangle}`. The parameter  $\langle language \rangle$  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 `\langle iso-code \rangle 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 `\langle iso-code \rangle 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 other 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 section 2.37.3.

The `\makeindex` command needs to know the sorting rules used in your language. OpTeX defines only a few language rules for sorting: Czech, Slovak and English. How to declare sorting rules for more languages are described in the section 2.33.

If you declare `\langle iso-code \rangle 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  $\langle quoted text \rangle$ . The language is specified by  $\langle iso-code \rangle$ . 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 section 2.37.3.

The `\` is used for quotes visually more similar to the `"` character which can be primary quotes or secondary quotes depending on the language rules. Maybe you want to alternate the 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 documents and you don't need to create your own macros.

The `\report` declaration is intended to create reports. It sets default font size to 11pt and `\parindent` (paragraph indentation) to 1.2em. 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 the next pages are numbered (from number 2). Footnotes are numbered from one in the whole document. The macro `\author <authors><end-line>` can be used when `\report` is declared. It prints `<authors>` in italics at the 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 11pt and `\parindent` to 0pt. 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 use any special mark at the end of lines: the ends 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 widest line used in it. So, you can use `\hfill\address...` to put the address box to the right side of the document. Or you can use `<prefixed text>\address...` to put `<prefixed text>` before the 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 documentation of `\slides` style in the file `op-slides.pdf`.

Analogical declaration macro `\book` is not prepared. Each book needs 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{<file-name>}` or by `\load[<file-names>]`. The first case (`\input`) is TeX primitive command, it can be used in the alternative old syntax `\input <filename><space>` too. The second case (`\load`) allows specifying a comma-separated list of included files. Moreover, it loads each macro file only once, it sets temporarily standard category codes during loading and it tries to load `<filename>.opm` or `<filename>.tex` or `<filename>`, the 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 source document is structured to more files (with individual chapters or sections), use simply the `\input` primitive.

The macro packages intended to OpTeX have the name `*.opm`. The following packages are distributed as part of OpTeX:

- `qrcode.opm` enables to create QR codes.
- `vlna.opm` enables to protect of one-letter prepositions and more things automatically.
- `emoji.opm` defines `\emoji{<name>}` command for colored emoticons.
- `plain-at.opm` defines the old names from plain TeX.

See the directory `optex/pkg/` and these files for more information about them.

### 1.7.4 Lorem ipsum dolor sit

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

### 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 the 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 the number of pages) is expanded by `\lastpage` macro. It expands to `?` in first TeX run and to the last page in next TeX runs.

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

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

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

### 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 a list of the items
\enditems       % end of list of the items
\beginblock     % start a block of text
\endblock       % end of block of text
\beginverbatim  % start a verbatim text
\endverbatim    % end verbatim text
\verbchar X     % initialization character X for in-text verbatim
\code          % another alternative for in-text verbatim
\verbatiminput  % verbatim extract from the external file
\beginmulti 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
\typo size [font-size/baselineskip] % size setting of typesetting
\typo scale [factor-font/factor-baselineskip] % size scaling
\the font size [size] \the font scale [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

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

\report \letter \slides % style declaration macros

```

## 1.9 API for macro writers

All TeX primitives and almost all OpTeX macros are accessible by two names: `\foo` (public or user name space) and `\_foo` (private name space). For example `\hbox` and `\_hbox` means the same TeX primitive. More about it is documented in section 2.2.

If this manual refers `\foo` then `\_foo` equivalent exists too. For example, we mention the `\addto` macro below. The `\_addto` equivalent exists too, but it is not explicitly mentioned here. If we refer only `\_foo` then its public equivalent does not exist. For example, we mention the `\_codedec1` macro below, so this macro is not available as `\codedec1`.

If you are writing a document or macros specific for the document, then use simply user namespace (`\foo`). If you are writing more general macros, then use private namespace (`\_foo`), but you should declare your own namespace by `\_namespace` macro and you have to follow the naming discipline described in section 2.2.4.



The alphabetically sorted list of macros typically usable for macro writers follows. More information about such macros can be found in the technical documentation. You can use hyperlinks here in order to go to the appropriate place of the technical documentation.

`\addto \macro{<text>}` adds `<text>` at the end of `\macro` body.

`\edef <char>{<body>}` defines `<char>` active character with meaning `<body>`.

`\afterfi {<text>}<ignored>\fi` expands to `\fi<text>`.

`\bp {<dimen expression>}` expands T<sub>E</sub>X dimension to decimal number in bp without unit.

`\_codedecl <sequence> {<info>}` is used at beginning of macro files.

`\colordef \macro {<mix of colors>}` declares `\macro` as color switch.

`\cs {<string>}` expands `\<string>`.

`\_doc ... \_cod` encloses documentation text in the macro code.

`\eoldef \macro #1{<body>}` defines `\macro` with parameter separated to end of line.

`\_endcode` closes the part of macro code in macro files.

`\_endnamespace` closes name space declared by `\_namespace`.

`\eqbox [<label>]{<text>}` creates `\hbox{<text>}` with common width across whole document.

`\expr {<expression>}` expands to result of the `<expression>` with decimal numbers.

`\fontdef \f {<font spec.>}` declares `\f` as font switch.

`\fontlet \fa=\fb <sizespec.>` declares `\fa` as the same font switch like `\fb` at given `<sizespec.>`.

`\foreach <list>\do <parameters>{<what>}` is expandable loop over `<list>`.

`\foreachdef \macro <parameters>{<what>}` declares expandable `\macro` as loop over `<list>`.

`\fornum <from>..<to>\do {<what>}` is expandable loop with numeric variable.

`\incr <counter>` increases and `\decr <counter>` decreases `<counter>` by one globally.

`\ignoreit <one>`, `\ignoresecond <one><two>` ignores given parameter.

`\expandafter \ignorept \the<dimen>` expands to decimal number `<dimen>` without pt.

`\isempty`, `\istokseempty`, `\isequal`, `\ismacro`, `\isdefined`, `\isinlist` `\isfile`, `\isfont` do various tests. Example: `\isinlist\list{<text>}\iftrue` does `\iftrue` if `<text>` is in `\list`.

`\isnextchar <char>{<text1>}{<text2>}` performs `<text1>` if next character is `<char>`, else `<text2>`.

`\kv {<key>}` expands to value when key-value parameters are used.

`\loop ... \repeat` is classical Plain T<sub>E</sub>X loop.

`\mathstyles {<math list>}` enables to create macros dependent on current math style.

`\_namespace {<pkg>}` declares name space used by package writers.

`\newcount`, `\newdimen` etc. are classical Plain T<sub>E</sub>X allocators.

`\newif \iffoo` declares boolean `\iffoo` as in Plain T<sub>E</sub>X.

`\_newifi \_iffoo` declares boolean `\_iffoo`.

`\opinput {<filename>}` reads file like `\input` but with standard catcodes.

`\optdef \macro [<opt-default>] <parameters>{<body>}` defines `\macro` with [opt.parameter].

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

`\private <sequence> <sequence> ...` ; declares `<sequence>`s for private name space.

`\public <sequence> <sequence> ...` ; declares `<sequence>`s for public name space.

`\readkv \macro` reads parameters from `\macro` in key-value format.

`\replstring \macro{<stringA>}{<stringB>}` replaces all `<stringA>` to `<stringB>` in `\macro`.

`\sdef {<string>}<parameters>{<body>}` behaves like `\def\<string><parameters>{<body>}`.

`\setctable` and `\restorectable` manipulate with stack of catcode tables.

`\slet {<stringA>}{<stringB>}` behaves like `\let\<stringA>=\<stringB>`

`\sxdef {<string>}<parameters>{<body>}` behaves like `\xdef\<string><parameters>{<body>}`.

`\trycs {<string>}{<text>}` expands `\<string>` if it is defined else expands `<text>`.

`\useit <one>`, `\usessecond <one><two>` uses given parameter.

`\wlog {<text>}` writes `<text>` to .log file.

`\wterm {<text>}` writes `<text>` to the terminal and .log file.

`\xargs <what> <token> <token> ...` ; repeats `<what><token>` for each `<token>`.

## 1.10 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`). If you need to use the basic set of old Plain T<sub>E</sub>X control sequences like `\p@` (for example you are reading an old macro file), use `\load[plain-at]`.

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 users document too). User should re-define unprefixed forms of control sequences without worries that something internal will be broken (only the sequence `\par` cannot be re-defined without change of internal T<sub>E</sub>X behavior because it is hard-coded in T<sub>E</sub>X, unfortunately).

The Latin Modern 8bit fonts instead Computer Modern 7bit fonts are preloaded in the format, but only a 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 the 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 the typographical area is not at the top left 1 in 1 in coordinates but at the top left paper corner exactly. For example, `\hoffset` includes directly left margin.

The tabbing macros `\settabs` and `\+` (from Plain T<sub>E</sub>X) are not defined in OpT<sub>E</sub>X because they are obsolete. But you can use the [OpT<sub>E</sub>X trick 0021](#) if you really need such feature.

The `\sec` macro is reserved for 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 an active link that points (usually) to the main documentation point. The main documentation point can be an active link that 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 the main file when the format is generated.

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

`optex.ini`

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 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
```

`optex.ini`

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

```
22 % OpTeX version
23
24 \def\optexversion{1.02 Apr.2021}
25 \def\fmtname{OpTeX}
26 \let\fmtversion=\optexversion
```

`optex.ini`

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

```

28 % Engine testing:
29
30 \newlinechar=`^^J
31 \ifx\directlua\undefined
32   \message{This format is based only on LuaTeX, use luatex -ini optex.ini^^J}
33 \endinput \fi
34
35 \ifx\bgroup\undefined \else
36   \message{This file can be used only for format initialisation, use luatex -ini^^J}
37 \endinput \fi

```

The basic macros for macro file syntax is defined, i.e. `\endcode`, `\_doc` and `\_cod`. The `\_codedec1` will be re-defined later.

```

39 % Basic .opm syntax:
40
41 \let\_endcode =\endinput
42 \def\_codedec1 #1#2{\message{#2^^J}}% information about .opm file
43 \long\def\_doc#1\_cod#2 {} % skip documentation

```

Individual \*.opm macro files are read.

```

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

```

The file `optex.lua` is embedded into the format as byte-code. It is documented in section 2.39.

```

91 \_directlua{
92   % preload OpTeX's Lua code into format as bytecode
93   lua.bytecode[1] = assert(loadfile(kpse.find_file("optex", "lua")))
94 }

```

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

```

96 \_everyjob = {%
97   \_message{This is OpTeX (Olsak's Plain TeX), version <\optexversion>^^J}%
98   \_mathchardef\_fnotestack=\_pdfcolorstackinit page {0 g 0 G}%
99   \_directlua{lua.bytecode[1]()}% load OpTeX's Lua code
100   \_mathsbon % replaces \int_a^b to \int _a^b
101   \_inputref % inputs \jobname.ref if exists
102 }
103
104 \_dump

```

## 2.2 Concept of namespaces 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 neither internal macros of OpTeX nor TeX primitives will be 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 a 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 knows what happens.

We never change catcode of `_`, so internal macros can be redefined by user without problems if it is desired. We don't need something like `\makeatletter` 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 their non-prefixed versions. 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 which macros are defined for public usage in that code segment.

The macro `\private` does the reverse job of `\public` with the same syntax. For example `\private \foo \bar ;` does `\let\_foo=\foo`, `\let\_bar=\bar`. This should be used when an unpre-fixed variant of a 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 unprefixed), then the accompanying text mentions only the unprefixed variant. The code typically defines the prefixed variant and then the `\public` (or `\_public`) macro is used.

### 2.2.2 Namespace of control sequences for users

Users 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 the 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 packages internal macros.
- Multiletter control sequences 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 2.15 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. Users 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 a 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 protects 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 does not need to 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 `\_endnamespace` command or when another `\_namespace` is used. This command checks if the same package label is not declared by the `\_namespace` twice.

The `\_nspublic` macro does `\let\_foo = \_⟨pkg⟩_foo` when `\_namespace{⟨pkg⟩}` is declared. Moreover, it prints a warning if `\_foo` is defined already. The `\_nsprivate` macro does reverse operation to it without warnings. Example: you can define `\def\_macro{...}` and then set it to the user name space by `\_nspublic \_macro;`.

Don't load other packages (which are using their own namespace) inside your namespace. Do load them before your `\_namespace {⟨pkg⟩}` is initialized. Or close your namespace by `\_endnamespace` and open it again (after other packages are loaded) by `\_resetnamespace {⟨pkg⟩}`.

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

- Don't use control sequences from the user namespace in the macro bodies if there is no explicit and documented reason to do this.
- Don't declare control sequences in the user namespace if there are no explicit and documented reasons to do this.

---

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

- Use control sequences from OpTeX and primitive namespace in read-only mode, if there is not an 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 `\_codedecl` 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.
- You can write more documentation after the `\_endcode` command.
- The OpTeX catcodes are set when `\load` your package (i.e. plain TeX catcodes plus catcode of `_` is 11). If a catcode is changed during loading your package then it is forgot because `\load` returns to catcodes used before loading package. If you want to offer a catcode changing for users then insert it to a macro which can be used after loading.

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 precedence before `.tex` when the `\load` macro is used.

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

## 2.2.6 The implementation of the namespaces

```
3 \_codedecl \_public {Prefixing and code syntax <2020-02-25>} % preloaded in format prefixed.opm
```

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

```
9 \let\_directlua = \directlua prefixed.opm
10 \_directlua {
11   % enable all TeX primitives with _ prefix
12   tex.enableprimitives('_', tex.extraprimitives('tex'))
13   % enable all primitives without prefixing
14   tex.enableprimitives('', tex.extraprimitives())
15   % enable all primitives with _ prefix
16   tex.enableprimitives('_', tex.extraprimitives())
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 <sequence> <sequence> ... ;` does `\let \_<sequence> = \_<sequence>` for all sequences.

`\_private <sequence> <sequence> ... ;` does `\let \_<sequence> = \_<sequence>` for all sequences.

`\_checkexists <where> <prefix><sequence>` prints error if the control sequence propagated to a new name space by `\_public` etc. macros is not declared.

`\_xargs <what> <sequence> <sequence> ... ;` does `<what><sequence>` for each sequences.

```
38 \_let\_ea = \_expandafter % usefull shortcut prefixed.opm
39
40 \_long\_def \_xargs #1#2{\_ifx #2;\_else \_ea#1\_ea#2\_ea\_xargs \_ea #1\_fi}
41
42 \_def \_pkglabel{}
43 \_def \_public {\_xargs \_publicA}
44 \_def \_publicA #1{%
45   \_checkexists \_public #1%
46   \_ea\_let \_ea#1\_csname \_csstring #1\_endcsname
47 }
48 \_def \_private {\_xargs \_privateA}
49 \_def \_privateA #1{%
50   \_checkexists \_private {}#1%
51   \_ea\_let \_csname \_csstring #1\_endcsname =#1%
52 }
53 \_def\_checkexists #1#2#3{\_unless \_ifcsname #2\_csstring#3\_endcsname
54   \_errmessage {\_string#1: \_bslash#2\_csstring#3 must be declared}\_fi
55 }
56 \_public \_public \_private \_xargs \_ea ;
```

Each macro file should begin with `\_codedecl \macro {⟨info⟩}`. If the `\macro` is defined already then the `\endinput` protects to read such file more than once. Else the `⟨info⟩` is printed to the terminal and the file is read.

The `\_endcode` is defined as `\endinput` in the `optex.ini` file. `\wterm {⟨text⟩}` prints `⟨text⟩` to the terminal and to the `.log` file (as in plain TeX).

prefixed.opm

```
68 \_def \_codedecl #1#2{%
69   \_ifx #1\_undefined \_wterm{#2}%
70   \_else \_ea \_endinput \_fi
71 }
72 \_def \_wterm {\_immediate \_write16 }
73
74 \_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

```
81 \_private \optexversion \fmtname ;
```

The `\_mathsbon` and `\_mathsboff` are defined in `math-macros.opm` file. Now, we define the macros `\_namespace {⟨pkg label⟩}`, `\_resetnamespace {⟨pkg label⟩}`, `\_endnamespace`, `\_nspublic` and `\_nsprivate` for package writers, see section 2.2.4.

prefixed.opm

```
91 \_def \_pkglabel{}
92 \_def \_namespace #1{%
93   \_ifcsname namesp:#1\_endcsname \_errmessage
94   {The name space "#1" is used already, it cannot be used twice}%
95   \_endinput
96   \_else \_resetnamespace{#1}\_fi
97 }
98 \_def \_resetnamespace #1{%
99   \_ea \_gdef \_csname namesp:#1\_endcsname {}%
100   \_gdef \_pkglabel{#1}%
101   \_directlua{
102     callback.add_to_callback("process_input_buffer",
103       function (str)
104         return string.gsub(str, "\_nbb[.]( [a-zA-Z])", "\_nbb _#1\_pcent 1")
105       end, "\_namespace")
106   }%
107 }
108 \_def \_endnamespace {%
109   \_directlua{ callback.remove_from_callback("process_input_buffer", "\_namespace") }%
110   \_gdef \_pkglabel{}%
111 }
112
113 \_def \_nspublic {\_xargs \_nspublicA}
114 \_def \_nspublicA #1{%
115   \_checkexists \_nspublic {\_pkglabel _}#1%
116   \_unless\_ifx #1\_undefined
117   \_opwarning{\_ea\_ignoreit\_pkglabel\_space redefines the meaning of \_string#1}\_fi
118   \_ea\_let \_ea#1\_csname \_pkglabel \_csstring #1\_endcsname
119 }
120 \_def \_nsprivate {\_xargs \_nsprivateA}
121 \_def \_nsprivateA #1{%
122   \_checkexists \_nsprivate {}#1%
123   \_ea\_let \_csname \_pkglabel \_csstring #1\_endcsname =#1%
124 }
```

## 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 \_def \pdfcreationdate {\_pdffeedback creationdate}
53 \_def \pdffontname {\_pdffeedback fontname}
54 \_def \pdffontobjnum {\_pdffeedback fontobjnum}
55 \_def \pdffontsize {\_pdffeedback fontsize}
56 \_def \pdfpageref {\_pdffeedback pageref}
57 \_def \pdfcolorstackinit {\_pdffeedback colorstackinit}
58 \_protected\_def \pdfliteral {\_pdfextension literal}
59 \_protected\_def \pdfcolorstack {\_pdfextension colorstack}
60 \_protected\_def \pdfsetmatrix {\_pdfextension setmatrix}
61 \_protected\_def \pdfsave {\_pdfextension save\_relax}
62 \_protected\_def \pdfrestore {\_pdfextension restore\_relax}
63 \_protected\_def \pdfobj {\_pdfextension obj }
64 \_protected\_def \pdfrefobj {\_pdfextension refobj }
65 \_protected\_def \pdfannot {\_pdfextension annot }
66 \_protected\_def \pdfstartlink {\_pdfextension startlink }
67 \_protected\_def \pdfendlink {\_pdfextension endlink\_relax}
68 \_protected\_def \pdfoutline {\_pdfextension outline }
69 \_protected\_def \pdfdest {\_pdfextension dest }
70 \_protected\_def \pdfthread {\_pdfextension thread }
71 \_protected\_def \pdfstartthread {\_pdfextension startthread }
72 \_protected\_def \pdfendthread {\_pdfextension endthread\_relax}
73 \_protected\_def \pdfinfo {\_pdfextension info }
74 \_protected\_def \pdfcatalog {\_pdfextension catalog }
75 \_protected\_def \pdfnames {\_pdfextension names }
76 \_protected\_def \pdfincludechars {\_pdfextension includechars }

```



```

77 \_protected\_def \_pdffontattr      {\_pdfextension fontattr }
78 \_protected\_def \_pdfmapfile       {\_pdfextension mapfile }
79 \_protected\_def \_pdfmapline       {\_pdfextension mapline }
80 \_protected\_def \_pdftrailer        {\_pdfextension trailer }
81 \_protected\_def \_pdfglyphtounicode {\_pdfextension glyphtounicode }
82
83 \_protected\_edef \_pdfcompresslevel {\_pdfvariable compresslevel}
84 \_protected\_edef \_pdfobjcompresslevel {\_pdfvariable objcompresslevel}
85 \_protected\_edef \_pdfdecimaldigits {\_pdfvariable decimaldigits}
86 \_protected\_edef \_pdfgamma         {\_pdfvariable gamma}
87 \_protected\_edef \_pdfimageresolution {\_pdfvariable imageresolution}
88 \_protected\_edef \_pdfimageapplygamma {\_pdfvariable imageapplygamma}
89 \_protected\_edef \_pdfimagegamma     {\_pdfvariable imagegamma}
90 \_protected\_edef \_pdfimagehicolor   {\_pdfvariable imagehicolor}
91 \_protected\_edef \_pdfimageaddfilename {\_pdfvariable imageaddfilename}
92 \_protected\_edef \_pdfpkresolution   {\_pdfvariable pkresolution}
93 \_protected\_edef \_pdfinclusioncopyfonts {\_pdfvariable inclusioncopyfonts}
94 \_protected\_edef \_pdfinclusionerrorlevel {\_pdfvariable inclusionerrorlevel}
95 \_protected\_edef \_pdfgentounicode   {\_pdfvariable gentounicode}
96 \_protected\_edef \_pdfpagebox        {\_pdfvariable pagebox}
97 \_protected\_edef \_pdfminorversion   {\_pdfvariable minorversion}
98 \_protected\_edef \_pdfuniqueresname  {\_pdfvariable uniqueresname}
99 \_protected\_edef \_pdfhorigin         {\_pdfvariable horigin}
100 \_protected\_edef \_pdfvorigin         {\_pdfvariable vorigin}
101 \_protected\_edef \_pdflinkmargin      {\_pdfvariable linkmargin}
102 \_protected\_edef \_pdfdestmargin      {\_pdfvariable destmargin}
103 \_protected\_edef \_pdfthreadmargin    {\_pdfvariable threadmargin}
104 \_protected\_edef \_pdfpagesattr       {\_pdfvariable pagesattr}
105 \_protected\_edef \_pdfpageattr        {\_pdfvariable pageattr}
106 \_protected\_edef \_pdfpageresources   {\_pdfvariable pageresources}
107 \_protected\_edef \_pdfxformattr       {\_pdfvariable xformattr}
108 \_protected\_edef \_pdfxformresources {\_pdfvariable xformresources}
109 \_protected\_edef \_pdfpkmode          {\_pdfvariable pkmode}
110
111 \_public
112 \pdftexversion \pdftexrevision \pdflastlink \pdfretval \pdflastobj
113 \pdflastannot \pdfxformname \pdfcreationdate \pdffontname \pdffontobjnum
114 \pdffontsize \pdfpageref \pdfcolorstackinit \pdfliteral \pdfcolorstack
115 \pdfsetmatrix \pdfsave \pdfrestore \pdfobj \pdfrefobj \pdfannot
116 \pdfstartlink \pdfendlink \pdfoutline \pdfdest \pdfthread \pdfstartthread
117 \pdfendthread \pdfinfo \pdfcatalog \pdfnames \pdfincludechars \pdffontattr
118 \pdfmapfile \pdfmapline \pdftrailer \pdfglyphtounicode \pdfcompresslevel
119 \pdfobjcompresslevel \pdfdecimaldigits \pdfgamma \pdfimageresolution
120 \pdfimageapplygamma \pdfimagegamma \pdfimagehicolor \pdfimageaddfilename
121 \pdfpkresolution \pdfinclusioncopyfonts \pdfinclusionerrorlevel
122 \pdfgentounicode \pdfpagebox \pdfminorversion \pdfuniqueresname \pdfhorigin
123 \pdfvorigin \pdflinkmargin \pdfdestmargin \pdfthreadmargin \pdfpagesattr
124 \pdfpageattr \pdfpageresources \pdfxformattr \pdfxformresources \pdfpkmode ;
125
126 \_pdfminorversion      = 5
127 \_pdfobjcompresslevel = 2
128 \_pdfcompresslevel     = 9
129 \_pdfdecimaldigits     = 3
130 \_pdfpkresolution      = 600

```

## 2.4 Basic macros

We define first bundle of basic macros.

```

3 \_codedecl \sdef {Basic macros for OpTeX <2021-02-03>} % loaded in format

```

basic-macros.opm

`\bgroup`, `\egroup`, `\empty`, `\space`, `\null` and `\wlog` are classical macros from plain TeX.

basic-macros.opm

```

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 \_public \bgroup \egroup \empty \space \null \wlog ;

```

`\ignoreit` ignores next token or `{\text}`, `\useit{\text}` expands to `\text` (removes outer braces), `\ignoresecond` uses first, ignores second parameter and `\usessecond` ignores first, uses second parameter.

basic-macros.opm

```
24 \_long\_def \_ignoreit #1{
25 \_long\_def \_useit #1{#1}
26 \_long\_def \_ignoresecond #1#2{#1}
27 \_long\_def \_usessecond #1#2{#2}
28 \_public \_ignoreit \_useit \_ignoresecond \_usessecond ;
```

`\bslash` is “normal backslash” with category code 12. `\nbb` is double backslash and `\pcent` is normal %.

They can be used in Lua codes, for example.

basic-macros.opm

```
37 \_edef \_bslash {\_csstring\}
38 \_edef \_nbb {\_bslash\_bslash}
39 \_edef \_pcent{\_csstring\%}
40 \_public \_bslash \_nbb \_pcent ;
```

`\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`.

basic-macros.opm

```
52 \_def \_sdef #1{\_ea\_def \_csname#1\_endcsname}
53 \_def \_sxdef #1{\_ea\_xdef \_csname#1\_endcsname}
54 \_def \_slet #1#2{\_ea\_let \_csname#1\_ea\_endcsname \_csname#2\_endcsname}
55 \_public \_sdef \_sxdef \_slet ;
```

`\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...}`.

basic-macros.opm

```
63 \_def \_adef #1{\_catcode`#1=13 \_begingroup \_lccode`~=#1\_lowercase{\_endgroup\_def~}}
64 \_public \_adef ;
```

`\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`.

basic-macros.opm

```
74 \_def \_cs #1{\_csname#1\_endcsname}
75 \_def \_trycs#1#2{\_ifcsname #1\_endcsname \_csname #1\_ea\_endcsname \_else #2\_fi}
76 \_public \_cs \_trycs ;
```

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

basic-macros.opm

```
82 \_long\_def \_addto #1#2{\_ea\_def \_ea#1\_ea{#1#2}}
83 \_public \_addto ;
```

`\incr\counter` increases `\counter` by one globally. `\decr\counter` decreases `\counter` by one globally.

basic-macros.opm

```
90 \_def \_incr #1{\_global\_advance#1by1 }
91 \_def \_decr #1{\_global\_advance#1by-1 }
92 \_public \_incr \_decr ;
```

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

basic-macros.opm

```
98 \_def \_opwarning #1{\_wterm{WARNING 1.\_the\_inputlineno: #1.}}
99 \_public \_opwarning ;
```

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

basic-macros.opm

```
107 \_def \_loggingall{\_tracingcommands=3 \_tracingstats=2 \_tracingpages=1
108 \_tracingoutput=1 \_tracinglostchars=1 \_tracingmacros=2
109 \_tracingparagraphs=1 \_tracingrestores=1 \_tracingscantokens=1
110 \_tracingifs=1 \_tracinggroups=1 \_tracingassigns=1 }
111 \_def \_tracingall{\_tracingonline=1 \_loggingall}
112 \_public \_loggingall \_tracingall ;
```

Write a warning if the user did not load a Unicode Font *or* if there were unresolved references. `\byehook` is used in the `\bye` macro.

```

119 \_def\_byehook{%
120   \_ifx\_initunifonts\_relax \_relax\_else \_opwarning{Unicode font was not loaded}\_fi
121   \_ifnum\_unresolvedrefs>0 \_opwarning{Try to rerun to get references right}\_fi
122 }

```

## 2.5 Allocators for T<sub>E</sub>X registers

Like plainT<sub>E</sub>X, the allocators `\newcount`, `\newwrite`, etc. are defined. The registers are allocated from 256 to the `\mai<type>` which is 65535 in LuaT<sub>E</sub>X.

Unlike in PlainT<sub>E</sub>X, 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 as in old plainT<sub>E</sub>X, 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 the top of the 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]
  \global\advance\_maicount by -#2\relax
  \ifnum \_countalloc > \_maicount
    \errmessage{No room for a new array of \string\count}%
  \else
    \global\chardef#1=\_maicount
  \fi
}
\_def\_usecount #1[#2]{% \_usecount \foo[2]
  \count\numexpr#1+#2\relax
}

```

alloc.opm

```

3 \_codedecl \_newdimen {Allocators for registers <2021-02-15>} % 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 \<sequence> {\<type>} \<primitive declarator>` is defined. This idea was used in classical plain T<sub>E</sub>X by Donald Knuth too but the macro from plain T<sub>E</sub>X seems to be more complicated:).

alloc.opm

```

41 \_def\_allocator #1#2#3{%
42   \_incr{\_cs{#2alloc}}}%
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   \_decr\_insertalloc
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 `\newcatcodetable` have their counter allocated by the `\newcount` macro.

alloc.opm

```

93 \_newcount \_attributealloc \_attributealloc=0
94 \_chardef\_maiattribute=\_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 $\TeX$ . We absolutely don't support the `@category` dance, so `\z@skip` `\z@`, `\p@` etc. are not defined in Op $\TeX$ . If you need such control sequences then you can initialize them by `\load[plain-at]`.

Only the `\_zo` and `\_zoskip` (equivalents to `\z@` and `\z@skip`) are declared here and used in some internal macros of Op $\TeX$  for improving speed.

alloc.opm

```

122 \_newdimen\_maxdimen \_maxdimen=16383.99999pt % the largest legal <dimen>
123 \_newdimen\_zo \_zo=0pt
124 \_newskip\_hideskip \_hideskip=-1000pt plus 1fill % negative but can grow

```

```

125 \_newskip\_centering \_centering=0pt plus 1000pt minus 1000pt
126 \_newskip\_zoskip \_zoskip=0pt plus0pt minus0pt
127 \_newbox\_voidbox % permanently void box register
128
129 \_public \maxdimen \hideskip \centering \voidbox ;

```

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

```

3 \_codedecl \_newif {Special if-macros, is-macros and loops <2021-02-03>} % 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 `\_newifi` enables to declare `\_ifxxx` and to use `\_xxxtrue` and `\_xxxfalse`. This means that it is usable for the internal namespace (`_`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 ;

```

`\afterfi {<what to do>}<ignored>\fi` closes condition by `\fi` and processes `<what to do>`. Usage:

```
\if<something> \afterfi{<result is true>} \else \afterfi{<resut is false>} \fi
```

```

40 \_def\_afterfi#1#2\_fi{\_fi#1}
41 \_def\_afterfi#1#2\_fi{\_fi#1}

```

### 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.

```

57 \_long\_def \_loop #1\_repeat{\_def\_body{#1}\_iterate}
58 \_long\_def \_loop #1\_repeat{\_def\_body{#1}\_iterate}
59 \_let \_repeat=\_fi % this makes \loop...\if...\repeat skippable
60 \_let \_repeat=\_fi
61 \_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.

`\foreach <list>\do <parameter-mask>{<what>}` reads parameters from `<list>` repeatedly and does `<what>` for each such reading. The parameters are declared by `<parameter-mask>`. Examples:

```

\foreach (a,1)(b,2)(c,3)\do (#1,#2){#1=#2 }
\foreach word1,word2,word3,\do #1,{Word is #1.}
\foreach A=word1 B=word2 \do #1=#2 {"#1 is set as #2".}

```

Note that `\foreach <list>\do {<what>}` is equivalent to `\foreach <list>\do #1{<what>}`.

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

if-macros.opm

```

84 \_newcount\_frnum % the numeric variable used in \fornum
85 \_def\_do{\_doundefined} % we need to ask \_ifx#1\_do ...
86
87 \_long\_def\_foreach #1\_do #2#{\_isempty{#2}\_iftrue
88   \_afterfi{\_foreachA{#1}{#1}}\_else\_afterfi{\_foreachA{#1}{#2}}\_fi}
89 \_long\_def\_foreachA #1#2#3{\_putforstack
90   \_immediateassignment \_long\_gdef\_fbody#2{\_testparam#1..\_iftrue #3\_ea\_fbody\_fi}%
91   \_fbody #1#2\_finbody\_getforstack
92 }
93 \_def\_testparam#1#2#3\_iftrue{\_ifx##1\_empty\_ea\_finbody\_else}
94 \_def\_finbody#1\_finbody{}
95
96 \_long\_def\_foreach #1\_do#2#{\_isempty{#2}\_iftrue
97   \_afterfi{\_foreachA{#1}{#1}}\_else\_afterfi{\_foreachA{#1}{#2}}\_fi}

```

`\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 `<from>`, `<to>`, `<step>` parameters can be numeric expressions. The macro is expandable.

The test in the `\_fornumB` says: if (`<to> < current number>` AND `<step>` is positive) or if (`<to> > <current number>` AND `<step>` is negative) then close loop by `\_getforstack`. Sorry, the condition is written by somewhat cryptoid T<sub>E</sub>X language.

if-macros.opm

```

112 \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
113 \_long\_def\_fornumstep#1:#2..#3\_do#4{\_putforstack
114   \_immediateassigned{%
115     \_gdef\_fbody##1{#4}%
116     \_global\_frnum=\_numexpr#2\_relax
117   }%
118   \_ea\_fornumB\_ea{\_the\_numexpr#3\_ea}\_ea{\_the\_numexpr#1}%
119 }
120 \_def\_fornumB #1#2{\_ifnum#1\_ifnum#2>0<\_else>\_fi \_frnum \_getforstack
121   \_else \_ea\_fbody\_ea{\_the\_frnum}%
122   \_immediateassignment\_global\_advance\_frnum by#2
123   \_afterfi{\_fornumB{#1}{#2}}\_fi
124 }
125 \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
126 \_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 T<sub>E</sub>X-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

```

144 \_newcount\_forlevel
145 \_def\_putforstack{\_immediateassigned{%
146   \_ifnum\_forlevel>0
147     \_sxddef\_frnum:\_the\_forlevel\_ea{\_the\_frnum}%
148     \_global\_slet{\_fbody:\_the\_forlevel}{\_fbody}%
149   \_fi
150   \_incr\_forlevel
151 }}
152 \_def\_getforstack{\_immediateassigned{%
153   \_decr\_forlevel
154   \_ifnum\_forlevel>0
155     \_global\_slet{\_fbody}{\_fbody:\_the\_forlevel}%

```



```

156 \global\frnum=\cs{frnum:\the\forlevel}\space
157 \fi
158 }}
159 \ifx\immediateassignment\undefined % for compatibility with older LuaTeX
160 \let\immediateassigned=\useit \let\immediateassignment=\empty
161 \fi

```

User can define own expandable “foreach” macro by `\foreachdef \macro <parameter-mask>{<what>}` which can be used by `\macro {<list>}`. The macro reads repeatedly parameters from `<list>` using `<parameter-mask>` and does `<what>` for each such reading. For example

```

\foreachdef\mymacro #1,{[#1]}
\mymacro{a,b,cd,efg,}

```

expands to `[a][b][cd][efg]`. Such user defined macros are more effective during processing than `\foreach` itself because they need not to operate with the for-stack.

```

176 \def\foreachdef#1#2#{\toks0{#2}%
177 \long\edef#1#1{\_ea\_noexpand\_csname\_body:\_csstring#1\_endcsname
178 \_the\_toks0\_noexpand\_finbody}%
179 \foreachdefA#1{#2}}
180 \def\foreachdefA#1#2#3{%
181 \long\sdef{body:\_csstring#1}#2{\_testparam##1..\_iftrue #3\_cs{body:\_csstring#1\_ea}\_fi}}
182
183 \_public \foreachdef ;

```

if-macros.opm

### 2.6.3 Is-macros

There are a collection of macros `\isempty`, `\istoksemt`, `\isequal`, `\ismacro`, `\isdefined`, `\isinlist`, `\isfile` and `\isfont` 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 {<text>}` \iftrue is true if the `<text>` is empty. This macro is expandable.

`\istoksemt <tokens variable>` \iftrue is true if the `<tokens variable>` is empty. It is expandable.

```

214 \long\def \isempty #1#2{\_if\_relax\_detokenize{#1}\_relax \_else \_ea\_unless \_fi#2}
215 \def \istoksemt #1#2{\_ea\_isempty\_ea{\_the#1}#2}
216 \_public \isempty \istoksemt ;

```

if-macros.opm

`\isequal {<textA>}{<textB>}` \iftrue is true if the `<textA>` and `<textB>` are equal, only from strings point of view, category codes are ignored. The macro is expandable.

```

225 \def\isequal#1#2#3{\_directlua{%
226 if "\luaescapestring{\_detokenize{#1}}"=="\luaescapestring{\_detokenize{#2}}"
227 then else tex.print("\_nbb unless") end}#3}
228 \_public \isequal ;

```

if-macros.opm

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

```

235 \def\ismacro#1{\_ea\_isequal\_ea{#1}}
236 \_public \ismacro ;

```

if-macros.opm

`\isdefined {<csname>}` \iftrue is true if `\<csname>` is defined. The macro is expandable.

```

243 \def\isdefined #1#2{\_ifcsname #1\_endcsname \_else \_ea\_unless \_fi #2}
244 \_public \isdefined ;

```

if-macros.opm

`\isinlist` `\list{⟨text⟩}``\iftrue` is true if the `⟨text⟩` is included the macro body of the `\list`. The category codes are relevant here. The macro is not expandable.

if-macros.opm

```
252 \_long\_def\_isinlist#1#2{\_begingroup
253   \_long\_def\_tmp##1#2##2\_end/\_%
254   {\_endgroup\_if\_relax\_detokenize{##2}\_relax \_ea\_unless\_fi}%
255   \_ea\_tmp#1\_endlistsep#2\_end/\_%
256 }
257 \_public \isinlist ;
```

`\isfile` `{⟨filename⟩}``\iftrue` is true if the file `⟨filename⟩` exists and are readable by  $\TeX$ .

if-macros.opm

```
264 \_newread \_testin
265 \_def\_isfile #1{%
266   \_openin\_testin =#{1}\_relax
267   \_ifeof\_testin \_ea\_unless
268   \_else \_closein\_testin
269   \_fi
270 }
271 \_public \isfile ;
```

`\isfont` `{⟨fontname or [fontfile]⟩}``\iftrue` is true if a given font exists. The result of this testing is saved to the `\_ifexistfam`.

if-macros.opm

```
279 \_newifi \_ifexistfam
280 \_def\_isfont#1#2{%
281   \_begingroup
282     \_suppressfontnotfounderror=1
283     \_font\_testfont=#{1}\_relax
284     \_ifx\_testfont\_nullfont \_def\_tmp{\_existfamfalse \_unless}
285     \_else \_def\_tmp{\_existfamtrue}\_fi
286     \_ea \_endgroup \_tmp #2%
287 }
288 \_public \isfont ;
```

The last macro `\isnextchar` `⟨char⟩``{⟨codeA⟩}``{⟨codeB⟩}` has a different syntax than all other is-macros. It executes `⟨codeA⟩` if next character is equal to `⟨char⟩`. Else the `⟨codeB⟩` is executed. The macro is not expandable.

if-macros.opm

```
297 \_long\_def\_isnextchar#1#2#3{\_begingroup\_toks0={\_endgroup#2}\_toks1={\_endgroup#3}%
298   \_let\_tmp= #1\_futurelet\_next\_isnextcharA
299 }
300 \_def\_isnextcharA{\_the\_toks\_ifx\_tmp\_next0\_else1\_fi\_space}
301
302 \_public \isnextchar ;
```

## 2.7 Setting parameters

The behavior of document processing by  $\text{Op}\TeX$  is controlled by *parameters*. The parameters are

- primitive registers used in build-in algorithms of  $\TeX$ ,
- registers declared and used by  $\text{Op}\TeX$  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 this control sequence then the appropriate register exists steadily and build-in algorithms are using it without change. But user cannot access its value in this case.  $\text{Op}\TeX$  declares two control sequences for each register: prefixed (private) and unprefixed (public).  $\text{Op}\TeX$  macros use only prefixed variants of control sequences. The user should use the unprefixed variant with the same meaning and set or read the values of registers using the unprefixed variant. If the user re-defines the unprefixed control sequence of a register then  $\text{Op}\TeX$  macros still work without change.

parameters.opm

```
3 \_codedecl \normalbaselineskip {Parameter settings <2021-04-13>} % preloaded in format
```

## 2.7.1 Primitive registers

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

parameters.opm

```
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 \_predisdisplaypenalty=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
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 Europe. If you set `\enlang` hyphenation patterns then `\nonfrenchspacing` is set.

```

100 \_normalbaselines % baseline setting, 10 pt font size

```

parameters.opm

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

```

109 \_emergencystretch=20pt % we want to use third pass of paragraph building algorithmh
110 % we need not 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

```

parameters.opm

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

```

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 ;

```

parameters.opm

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

The main principle of 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 an important difference: all "string-like" parameters are token lists in OpTeX (OPmac uses macros for them). The reason of this difference: if a user sets parameter by unprefixed (public) control sequence, an OpTeX macro can read *the same data* using a prefixed (private) control sequence.

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 TeX system where LuaTeX can find them). If you set a 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.

```

175 \_newtoks\_picdir
176 \_public \picdir ;

```

parameters.opm

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 the height is given, the 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 setting these parameters locally in the group where `\inspic` is used in order to not influence the dimensions of other 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.

```

194 \_newdimen\_picwidth \_picwidth=0pt \_let\picw=\_picwidth
195 \_newdimen\_picheight \_picheight=0pt
196 \_public \picwidth \picheight ;

```

parameters.opm

The `\everytt` is the token list used in `\begtt...\endtt` environment and in the verbatim group opened by `\verbinput` 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 the 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 `\verbchar` characters or by `\code{<text>}`. You can set `\everyintt={\Red}` for example if you want in-line verbatim in red color.

```

219 \_newtoks\_everytt
220 \_newtoks\_everyintt
221 \_public \everytt \everyintt ;

```

parameters.opm

The `\ttline` is used in `\begtt...\endtt` environment or in the code printed by `\verbinput`. If `\ttline` is positive or zero, then the verbatim code has numbered lines from `\ttline+1`. The `\ttline` register is re-set to a 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 `\verbinput` lines show the line numbers of inputted file. If `\ttline<-1` then no line numbers are printed.

```

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

```

parameters.opm

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

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

The `\iindent` gives default indentations used in the 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`.

parameters.opm

```

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

```

The tabulator `\tt` has its category code like space: it behaves as a space in normal text. This is a common 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 tabulator behaves like three spaces in multiline verbatim.

parameters.opm

```

271 \_newcount \_tabspaces \_tabspaces=3
272 \_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. The full color set must be declared here.

parameters.opm

```

282 \_newtoks\_hicolors
283 \_public \hicolors ;

```

The default item mark used between `\begitems` and `\enditems` is the bullet. The `\defaultitem` tokens list declares 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 the group is opened by `\begitems`

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

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

parameters.opm

```

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

```

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

parameters.opm

```

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

```

The `\begmulti` and `\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 the definite result of the width of the columns.

parameters.opm

```

321 \_newdimen\_colsep \_colsep=20pt % space between columns
322 \_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.

parameters.opm

```

330 \_newtoks \_everytocline
331 \_public \everytocline ;

```

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 the style file here or you can modify the more declaration for printing (fonts, baselineskip, etc.) or you can define specific macros used in your `.bib` file.

The `\biboptions` is used in the iso690 bib-style for global options, see section 2.32.5.

The `\bibpart` saves the name of bib-list if there are more bib-lists in single document, see section 2.32.1.

parameters.opm

```

345 \_newtoks\_bibtexhook
346 \_newtoks\_biboptions
347 \_newtoks\_bibpart
348 \_public \bibtexhook \biboptions \bibpart ;

```

`\everycapitonf` is used before printing caption in figures and `\everycapitont` is used before printing caption in tables.



parameters.opm

```

355 \newtoks\everycaptiont \newtoks\everycaptionf
356 \public \everycaptiont \everycaptionf ;

```

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

parameters.opm

```

363 \newtoks\everyii
364 \public \everyii ;

```

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

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

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

parameters.opm

```

375 \newtoks\everymnote
376 \newdimen\mnotesize \mnotesize=20mm % the width of the mnote paragraph
377 \newdimen\mnoteindent \mnoteindent=10pt % distance between mnote and text
378 \public \everymnote \mnotesize \mnoteindent ;

```

The `\table` parameters follow. 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 an 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 ppto<size>` is used.

parameters.opm

```

412 \newtoks\everytable \newtoks\thistable
413 \newtoks\tabiteml \newtoks\tabitemr \newtoks\tabstrut
414 \newdimen\tablinespace \newdimen\vvkern \newdimen\hhkern \newdimen\tsize
415 \newskip\tabskipl \newskip\tabskipr
416 \everytable={ } % code used after settings in \vbox before table processing
417 \thistable={ } % code used when \vbox starts, is removed after using it
418 \tabstrut={\strut}
419 \tabiteml={\enspace} % left material in each column
420 \tabitemr={\enspace} % right material in each column
421 \tablinespace=2pt % additional vertical space before/after horizontal rules
422 \vvkern=1pt % space between double vertical line and used in \frame
423 \hhkern=1pt % space between double horizontal line and used in \frame
424 \tabskipl=0pt\relax % \tabskip used before first column
425 \tabskipr=0pt\relax % \tabskip used after the last column
426 \public \everytable \thistable \tabiteml \tabitemr \tabstrut \tablinespace
427 \vvkern \hhkern \tsize \tabskipl \tabskipr ;

```

The `\eqalign` macro can be configured by `\eqlines` and `\eqstyle` tokens lists. The default values are set in order these macro behaves as in Plain T<sub>E</sub>X. The `\eqspace` is horizontal space put between equation systems if more columns in `\eqalign` are used.

parameters.opm

```

436 \newtoks \eqlines \eqlines={\openup\jot}
437 \newtoks \eqstyle \eqstyle={\strut\displaystyle}
438 \newdimen \eqspace \eqspace=20pt
439 \public \eqlines \eqstyle \eqspace ;

```

`\lmfil` is “left matrix filler” (for `\matrix` columns). The default value does centering because the right matrix filler is directly set to `\hfil`.

parameters.opm

```

446 \newtoks \lmfil \lmfil={\hfil}
447 \public \lmfil ;

```

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: \firstmark \hss}
\footline={\rmfixed \ifodd\pageno \hfill\fi \folio \hfil}
```

parameters.opm

```
465 \newtoks\headline \headline={}
466 \newtoks\footline \footline={\hss\rmfixed \folio \hss}
467 \public \headline \footline ;
```

The distance between the `\headline` and the top of the page text is controlled by the `\headlinedist` register. The distance between the bottom of page-text and `\footline` is `\footlinedist`. More precisely: baseline of headline and baseline of the 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 by plain TeX.

parameters.opm

```
481 \newdimen \headlinedist \headlinedist=14pt
482 \newdimen \footlinedist \footlinedist=24pt
483 \public \headlinedist \footlinedist ;
```

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

parameters.opm

```
491 \newskip \pgbottomskip \pgbottomskip=0pt \relax
492 \public \pgbottomskip ;
```

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={\rmfixed \firstmark \hfil}}
```

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

parameters.opm

```
506 \newtoks \nextpages
507 \public \nextpages ;
```

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}
```

parameters.opm

```
519 \newtoks \pgbackground \pgbackground={} % for page background
520 \public \pgbackground ;
```

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

parameters.opm

```
528 \newtoks \ovalparams
529 \newtoks \circleparams
530 %\ovalparams={\_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
531 % \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt }
532 %\circleparams={\_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
533 % \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt}
534
535 \newdimen \roundness \roundness=5mm % used in \clippingoval macro
536
537 \public \ovalparams \circleparams \roundness ;
```

OpTeX defines “Standard OpTeX markup language”<sup>2</sup> which lists selected commands from chapter 1 and gives their behavior when a converter from OpTeX document to HTML or Markdown or LaTeX is used.

<sup>2</sup> Will be developed in 2021.

The structure-oriented commands are selected here, but the commands which declare typographical appearance (page layout, dimensions, selected font family) are omitted. More information for such a converter should be given in `\cnvinfo{<data>}`. OpTeX simply ignores this but the converter can read its configuration from here. For example, a user can write:

```
\cnvinfo {type=html, <cnv-to-html-data>}
\cnvinfo {type=markdown, <cnv-to-markdown-data>}
```

and the document can be processed by OpTeX to create PDF, or by a converter to create HTML, or by another converter to create Markdown.

```
558 \_let\cnvinfo=\_ignoreit
```

parameters.opm

## 2.8 More OpTeX macros

The second bundle of OpTeX macros is here.

```
3 \_codedecl \eoldef {OpTeX useful macros <2021-04-25>} % preloaded in format
```

more-macros.opm

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 is returned back to the current values when the file is read. You can use `\opinput` in any situation inside the document and you will be sure that the file is read correctly with correct catcode settings.

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.

The `\opinput` works inside the catcode table stack. It reads `\optexcatcodes` table and stores it to `\tmpcatcodes` table. This table is actually used during `\input` (maybe catcodes are changed here). Finally, `\_restoretable` pops the stacks and returns to the catcodes used before `\opinput` is run.

```
29 \_def\_opinput #1{\_setctable\_optexcatcodes
30 \_savecatcodetable\_tmpcatcodes \_catcodetable\_tmpcatcodes
31 \_input {#1}\_relax\_restorectable}
32
33 \_newcatcodetable \_optexcatcodes
34 \_newcatcodetable \_plaintexcatcodes
35 \_newcatcodetable \_tmpcatcodes
36
37 \_public \optexcatcodes \plaintexcatcodes \opinput ;
38
39 \_savecatcodetable\_optexcatcodes
40 {\\_catcode\_8 \savecatcodetable\plaintexcatcodes}
```

more-macros.opm

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.

```
54 \_catcodetable0
55
56 \_def\_setctable#1{\_edef\_ctablelist{\\_the\_catcodetable}\\_ctablelist}%
57 \_catcodetable#1\_relax
58 }
59 \_def\_restorectable{\_ea\_restorectableA\_ctablelist\_relax}
60 \_def\_restorectableA#1#2\_relax{%
61 \_ifx^#2\_opwarning
62 {You can't use \_noindent\_restorectable without previous \_string\_setctable}%
63 \_else \_def\_ctablelist{#2}\_catcodetable#1\_relax \_fi
64 }
65 \_def\_ctablelist{.}
66
67 \_public \setctable \restorectable ;
```

more-macros.opm

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.

```
77 \def\normalcatcodes {\catcodetable\optecatodes \savecatcodetable0 \catcodetable0 }
78 \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 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`.

```
92 \def \load [#1]{\loadA #1,,,\end}
93 \def \loadA #1#2,{\ifx,#1 \ea \loadE \else \loadB{#1#2}\ea\loadA\fi}
94 \def \loadB #1{%
95   \ifcsname _load:#1\endcsname \else
96     \isfile {#1.opm}\iftrue \opinput {#1.opm}\else \opinput {#1}\fi
97     \sdef{_load:#1}{}%
98     \trycs{afterload}{\let\afterload=\undefined
99   \fi
100 }
101 \def \loadE #1\end{}
102 \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.

```
118 \def\optdef#1[#2]{%
119   \def#1{\opt={#2}\isnextchar[{\cs{oA:\string#1}}{\cs{oB:\string#1}}}%
120   \sdef{oA:\string#1}[##1]{\opt={##1}\cs{oB:\string#1\nospaceafter}}%
121   \sdef{oB:\string#1\nospaceafter}%
122 }
123 \def\nospaceafter#1{\ea#1\romannumeral-`\.}
124 \newtoks\opt
125
126 \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 that 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.

```
144 \def\eoldef #1{\def #1{\begingroup \catcode\`^M=12 \eoldefA #1}%
145   \ea\def\csname _\csstring #1:M\endcsname}
146 \catcode\`^M=12 %
147 \def\eoldefA #1#2^M{\endgroup\csname _\csstring #1:M\endcsname{#2}}%
148 \normalcatcodes %
149
150 \eoldef\skiptoel#1{}
151 \def\bracedparam#1{\ifcsname _\csstring #1:M\endcsname
152   \csname _\csstring #1:M\ea \endcsname
153   \else \csname __in\csstring #1:M\ea \endcsname \fi
154 }
155 \public \eoldef \skiptoel \bracedparam ;
```

`\scantoel\macro` *<text to end of line>* scans the *<text to end of line>* in verbatim mode and runs the `\macro`{*<text to end of line>*}. The `\macro` can be defined `\def\macro#1{... \scantextokens{#1}...}`.

The new tokenization of the parameter is processed when the parameter is used, no when the parameter is scanned. This principle is used in definition of `\chap`, `\sec`, `\secc` and `\_Xtoc` macros. It means that user can write `\sec text & text` for example. Inline verbatim works in title sections.

The verbatim scanner of `\scantoeol` keeps category 7 for `^` in order to be able to use `^^J` as comment character which means that the next line continues.

```

173 \_def\_scantoeol#1{\_def\_tmp{#1}\_begingroup \_setscancatcodes \_scantoeolA}
174 \_def\_setscancatcodes{\_setverb \_catcode\`^M=12\_catcode\`^=7\_catcode\`=10\_catcode\`^^J=14 }
175 \_catcode\`^M=12 %
176 \_def\_scantoeolA#1^^M{\_endgroup \_tmp{#1}}%
177 \_normalcatcodes %
178
179 \_public \scantoeol ;
more-macros.opm
```

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.

How it works: `\replstring\foo{<textA>}{<textB>}` prepares `\_replacestringsA#1<textA>{...}` and runs `\_replacestringsA<foo-body>?<textA>!<textA>`. So, `#1` includes the first part of `<foo-body>` before first `<textA>`. It is saved to `\_tmptoks` and `\_replacestringsB` is run in a loop. It finishes processing or appends the next part to `\_tmptoks` separated by `<textB>` and continues loop. The final part of the macro removes the last `?` from resulting `\_tmptoks` and defines a new version of the `\foo`.

```

199 \_newtoks\_tmptoks
200 \_catcode\`!=3 \_catcode\`?=3
201 \_def\_replstring #1#2#3{% \_replstring #1{stringA}{stringB}
202 \_long\_def\_replacestringsA##1#2{\_tmptoks{##1}\_replacestringsB}%
203 \_long\_def\_replacestringsB##1#2{\_ifx!##1\_relax \_else \_toksapp\_tmptoks{#3##1}%
204 \_ea\_replacestringsB\_fi}%
205 \_ea\_replacestringsA #1?#2!#2%
206 \_long\_def\_replacestringsA##1?{\_tmptoks{##1}\_edef#1{\_the\_tmptoks}}%
207 \_ea\_replacestringsA \_the\_tmptoks}
208 \_normalcatcodes
209
210 \_public \replstring ;
more-macros.opm
```

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 `\verbchar` macro). Nothing problematic happens if re-defined `\catcode` is used in this case.

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

```

222 \_def\_catcode#1{\_catcode \_if'\_noexpand#1\_ea\_else\_if"\_noexpand#1\_else
223 \_if'\_noexpand#1\_else \_ea\_ea\_ea\_ea\_ea\_ea\_ea#1\_fi\_fi\_fi}
more-macros.opm
```

The `\removespaces <text with spaces>{< >}` expands to `<textwithoutspaces>`.

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

```

232 \_def\_removespaces #1 {\_isempty{#1}\_iffalse #1\_ea\_removespaces\_fi}
233 \_ea\_def \_ea\_ignorept \_ea#\_ea1\_detokenize{pt}{#1}
234
235 \_public \removespaces \ignorept ;
more-macros.opm
```

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 the decimal point. The `<expression>` can include `++*/()` and decimal numbers in common syntax.

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

```

254 \def\decdigits{3} % digits after decimal point in \bp and \expr outputs.
255 \def\pttopb{%
256   \directlua{tex.print(string.format('\pcent.\decdigits f',
257     token.scan_dimen()/65781.76))}% pt to bp conversion
258 }
259 \def\bp#1{\ea\pttopb\dimexpr#1\relax}
260 \def\expr#1{\directlua{tex.print(string.format('\pcent.\decdigits f',#1))}}
261
262 \public \expr \bp ;

```

The pair `\_doc ... \_cod` is used for documenting macros and to printing the technical documentation of the OpTeX. 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.

```

277 \long\def\_doc #1\_cod {\_skiptoeol}

```

## 2.9 Using key=value format in parameters

Users or macro programmers can define macros with options in key=value format. It means a comma-separated list of equations key=value. First, we give an example.

Suppose that you want to define a macro `\myframe` with options: color of rules, color of text inside the frame, rule-width, space between text and rules. You want to use this macro as:

```

\myframe [margins=5pt,rule-width=2pt,frame-color=\Red,text-color=\Blue] {text1}
or
\myframe [frame-color=\Blue] {text2} % other parameters are default

```

You can define `\myframe` as follows:

```

\def\myframedefaults{% defaults:
  frame-color=\Black, % color of frame rules
  text-color=\Black, % color of text inside the frame
  rule-width=0.4pt, % width of rules used in the frame
  margins=2pt, % space between text inside and rules.
}
\optdef\myframe [] #1{\bgroup
  \ea\addto\ea\myframedefaults\ea{\the\opt}%
  \readkv\myframedefaults
  \rulewidth=\kv{rule-width}
  \hhkern=\kv{margins}\vvhkern=\kv{margins}\relax
  \kv{frame-color}\frame{\kv{text-color}\strut #1}%
  \egroup}

```

We recommend using `\optdef` for defining macros with optional parameters written in `[]`. Then the optional parameters are saved in the `\opt` tokens register. First: we append the `\opt` (actual optional parameters) to `\myframedefault` by `\addto` macro. Second: we read the parameters by `\readkv{<parameters list>}` macro. Third: the values can be used by expandable `\kv{<key>}` macro. The `\kv{<key>}` returns `???` if such key is not declared.

You can use keys without values in the parameters list too, but with additional care. For example, suppose `draft` option without parameter. If a user writes `\myframe [..., draft, ...]{text}` then `\myframe` should behave differently. We have to add `DRAFTv=0`, in `\myframedefault` macro. Moreover, `\myframe` macro must include preprocessing of `\myframedefault` using `\replstring` which replaces the occurrence of `draft` by `DRAFTv=1`.

```

\optdef\myframe [] #1{...
  \ea\addto\ea\myframedefaults\ea{\the\opt}%
  \replstring\myframedefaults{draft}{DRAFTv=1}%

```



keyval.opm

**Implementation.** The `\readkv` expands its parameter and does replace-strings in order to remove spaces around equal signs and after commas. Double commas are removed. Then `\kvscan` reads the parameters list finished by the double comma and saves values to `\kv{key}` macros. The `\kv{key}` expands the `\kv{key}` macro. If this macro isn't defined then `\kvunknown` is processed. You can re-define it if you want.

keyval.opm

All macros from plain `TEX` are rewritten here. Differences are mentioned in the documentation below.

plain-macros.opm

plain-macros.opm

```
plain-macros.opm
```

plain-macros.opm

Plain T<sub>F</sub>X classical `\obeylines` and `\obeyspaces`.

plain-macros.opm

55

Spaces. `\thinspace`, `\negthinspace`, `\enspace`, `\enskip`, `\quad`, `\qquad`, `\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\qquad {\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 \qquad \smallskip
81 \medskip \bigskip \nointerlineskip \offinterlineskip ;
82
83 \def\topglue {\nointerlineskip\vglue-\topskip\vglue} % for top of page
84 \def\vglue {\afterassignment\vglA \skip0=}
85 \def\vglA {\par \dimen0=\prevdepth \hrule height0pt
86 \nobreak\vskip\skip0 \prevdepth=\dimen0 }
87 \def\hglue {\afterassignment\hglA \skip0=}
88 \def\hglA {\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=\zo \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 to\zo{#1\hss}}
133 \def \llap #1{\hbox to\zo{\hss#1}}
134 \def\underbar #1{\setbox0=\hbox{#1}\dp0=\zo \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=\zskip \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 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 Unencoded 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.

```

228 \def \oalign #1{\leavevmode\vtop{\baselineskip=\_zo \lineskip=.25ex
229 \ialign{##\_crr\#1\_crr}}}
230 \def \oalignA {\lineskiplimit=\_zo \oalign}
231 \def \oalign {\lineskiplimit=-\_maxdimen \oalign} % chars over each other
232 \def \shiftx #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\_shiftx{-1ex}.\_hidewidth}}}
235 \def \_b #1{{\oalignA{\relax#1\_crr\_hidewidth\_shiftx{-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\_ifmode\_ldots\_else$\_math\_ldots\_thinsk$\_fi}
240 \_public \oalign \oalign \_d \_b \_c \_dots ;

```

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

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

```

250 \def \oldaccents {%
251 \def\##1{{\_accent\_tgrave ##1}}%
252 \def\'##1{{\_accent\_tacute ##1}}%
253 \def\v##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\_tilde ##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\_tilde=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\_tilde="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 plain T<sub>E</sub>X macros `\hrulefill`, `\dotfill`, `\rightarrowfill`, `\leftarrowfill`, `\downbracefill`, `\upbracefill`. The last four are used in non-Unicode variants of `\overrightarrow`, `\overleftarrow`, `\overbrace` and `\underbrace` macros, see section 2.15.

```

305 \def \hrulefill {\_leaders\_hrule\_hfill}

```

```

306 \def \dotfill {\cleaders\hbox{$\math \mkern1.5mu.\mkern1.5mu$\}\hfill}
307 \def \rightarrowfill {\math\smash-\mkern-7mu%
308 \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$\}\hfill
309 \mkern-7mu\mathord\rightarrow$}
310 \def \leftarrowfill {\math\mathord\leftarrow\mkern-7mu%
311 \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$\}\hfill
312 \mkern-7mu\smash-$}
313
314 \mathchardef \braceld="37A \mathchardef \bracerd="37B
315 \mathchardef \bracelu="37C \mathchardef \braceru="37D
316 \def \downbracefill {\math \setbox0=\hbox{$\braceld$}%
317 \braceld \leaders\vrule height\ht0 depth\zo \hfill \braceru
318 \bracelu \leaders\vrule height\ht0 depth\zo \hfill \bracerd$}
319 \def \upbracefill {\math \setbox0=\hbox{$\braceld$}%
320 \bracelu \leaders\vrule height\ht0 depth\zo \hfill \bracerd
321 \braceld \leaders\vrule height\ht0 depth\zo \hfill \braceru$}
322
323 \public \hrulefill \dotfill
324 \rightarrowfill \leftarrowfill \downbracefill \upbracefill ;

```

The last part of plain TeX macros: `\magnification`, `\bye`. Note that math macros are defined in the `math-macros.opm` file (section 2.15).

plain-macros.opm

```

332 \def \magnification {\afterassignment \magA \count255 }
333 \def \magA {\mag=\count255 \truedimen\hsize \truedimen\vsize
334 \dimen\footins=8truein
335 }
336 % only for backward compatibility, but \margins macro is preferred.
337 \public \magnification ;
338
339 \def \showhyphens #1{\setbox0=\vbox{\parfillskip=0pt \hsize=\maxdimen \tenrm
340 \pretolerance=-1 \tolerance=-1 \hbadness=0 \showboxdepth=0 \ #1}}
341
342 \def \bye {\par \vfill \supereject \byehook \end}
343 \public \showhyphens \bye ;

```

## 2.11 Preloaded fonts for text mode

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

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

fonts-preload.opm

```

3 \codedec1 \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 ;

```

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

### 2.12.1 The `\setfontsize` macro

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:

- `at` $\langle dimen\rangle$ , for example `\setfontsize{at12pt}`. It gives the desired font size directly.
- `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` $\langle number\rangle$ .
- `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 OpTeX is given by `\setfontsize{at10pt}`.

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

```
                The 10 pt text is here,  
\setfontsize{at12pt} the 10 pt text is here unchanged...  
\resizethefont    and the 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.
```

### 2.12.2 The `\font` primitive

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

The example above uses the 8 bit tfm font. You can use Unicode font too, of course. The `\fontfam` macro initializes the extended `\font` primitive features for LuaTeX (see section 2.13.14). If you didn't use this command, you must initialize these features by the `\initunifonts` command explicitly, for example:

```
\initunifonts  
\font\tencyklop=[cyklop-regular] at10pt % the font cyklop-regular.otf is loaded  
\def\cyklop{\tencyklop\resizethefont}  
  
\cyklop The 10 pt text is here  
\setfontsize{at12pt}  
\cyklop The 12 pt text is here
```

### 2.12.3 The `\fontdef` declarator

You can declare  $\langle newfont\rangle$  by the `\fontdef` command.

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

This command runs  $\langle font\ modifiers\rangle\ \langle variant-selector\rangle$  in an internal group and sets the resulting selected font as  $\langle newfont\rangle$ .



The resulting `\langle newfont \rangle` declared by `\fontdef` is “fixed font switch” independent of `\setfontsize` and other font modifiers. More exactly, it is a 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 2.13.

## 2.12.4 The `\fontlet` declarator

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

```
\fontlet \langle newfont \rangle = \langle fontswitch \rangle \langle sizespec \rangle
example:
\fontlet \bigfont = \_tenbf at15pt
```

The resulted `\bigfont` is the same as in the 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.

## 2.12.5 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\langle dimen \rangle` or `mag\langle coefficient \rangle` is used. When `scaled\langle scale factor \rangle` is used then optical size is chosen using the value of the `\defaultoptsize` register and such font is scaled by the specified `\langle scale factor \rangle`. 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.

## 2.12.6 Implementation notes

```
3 \_codedecl \setfontsize {Font resizing macros <2021-04-17>} % preloaded in format
fonts-resize.opm
```

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\langle number \rangle` format then the contents of the `\_sizespec` macro is re-calculated to the `at\langle dimen \rangle` format using previous `\_optsize` value.

```
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>\_zo \_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 ;
fonts-resize.opm
```

`\_resizefont`  $\langle variant-name \rangle$   $\langle font switch \rangle$ , resizes the font given by the variant. The  $\langle variant-name \rangle$  is `rm` or `bf` or `it` or `bi`. The new  $\langle font-switch \rangle$  is declared (roughly speaking) by:

`\_font`  $\langle font switch \rangle$  =  $\langle fontname of \rangle$  `\_ten` $\langle variant-name \rangle$  `\_sizepec`

If the `\_fontselector` is defined then it has priority before  $\langle font-switch \rangle$  and the font is loaded by:

`\_ea\_font` `\_fontselector` =  $\langle fontname of \rangle$  `\_ten` $\langle variant-name \rangle$  `\_sizepec`

The font is loaded by `\_doresizefont` $\langle font switch \rangle$ . This macro has meaning `\_doresizetfmfont` in TFM mode and it switches to `\_doresizeunifont` when `\_initunifonts` is used.

The  $\langle fontname of \rangle$  is generated by the `\fontname` TeX primitive where `\_rfontskipat` removes the  $\langle dimen \rangle$  part of the `\fontname` output. The  $\langle fontname of \rangle$  is generated differently in OTF mode, see `\_doresizeunifont` macro.

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

The `\_fontloaded` $\langle font switch \rangle$  is a macro which can be used for post-processing when a font is loaded.

fonts-resize.opm

```
60 \_def\_resizefont#1#2{%
61   \_edef\_whatresize{#1}%
62   \_ifx \_fontselector \_undefined
63     \_doresizefont#2\_relax \_ea\_fontloaded \_csname\_ten#1\_endcsname
64   \_else \_ea\_doresizefont \_fontselector \_ea\_fontloaded \_fontselector \_fi
65   \_lastmagsize=\_zo
66   \_slet{\_tryload#1}{\_relax}%
67 }
68 \_def\_doresizetfmfont#1{\_logfont{#1}%
69   \_ea\_font\_ea#1\_ea\_rfontskipat
70   \_fontname \_cs{\_ten\_whatresize} \_relax\_space \_sizepec \_relax
71 }
72 \_let\_doresizefont=\_doresizetfmfont
73 \_def\_logfont#1{} % default is no logging of used fonts
74
75 \_def\_rfontskipat#1{\_ifx#1"\_ea\_rfskipatX \_else\_ea\_rfskipatN\_ea#1\_fi}
76 \_def\_rfskipatX #1" #2\_relax{"\_whichftm{#1}"}
77 \_def\_rfskipatN #1 #2\_relax{\_whichftm{#1}}
```

If a font is loaded by `\_resizefont` or `\_resizethefont` then the `\_fontloaded` $\langle font switch \rangle$  is called immediately after it. If the font is loaded first then its `\skewchar` is equal to `-1`. We run `\_newfontloaded` $\langle font switch \rangle$  and set `\skewchar=-2` in this case. A user can define a `\_newfontloaded` macro. We are sure that `\_newfontloaded` macro is called only once for each instance of the font given by its name, OTF features and size specification. The `\skewchar` value is globally saved to the font (like `\fontdimen`). If it is used in math typesetting then it is set to a positive value.

The `\_newfontloaded` should be defined for micro-typographic configuration of fonts, for example. See [OpTeX/ trick 0058](#).

fonts-resize.opm

```
94 \_def\_fontloaded #1{\_ifnum\_skewchar#1=-1 \_skewchar#1=-2 \_newfontloaded#1\_fi}
95 \_def\_newfontloaded #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`  $\langle font switch B \rangle$ .

fonts-resize.opm

```
112 \_def \_fontdef #1#2{\_begingroup
113   \_ifx \_fontselector \_undefined \_def \_fontselector{#1}\_fi
114   \_reloading #2%
115   \_ea \_keepmeaning \_fontselector \_endgroup
116 }
117 \_def \_fontlet#1#2{\_ifx #2=\_ea\_fontlet \_ea#1\_else
118   \_ea\_font\_ea#1\_ea\_rfontskipat \_fontname#2 \_relax\_space \_fi
119 }
120 \_def \_keepmeaning #1#2{\_global\_let\_keepmeaningdata=#1%
```

```

121 #2\let#1=\_keepmeaningdata \_global\_let\_keepmeaningdata=\_undefined
122 }
123 \_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.

```

137 \_def \_newcurrfontsize #1{% \newcurrfontsize{at25pt}
138   \_edef\_tmp{\_ea\_csstring \_the\_font}%
139   \_ea \_fontlet \_csname \_tmp\_ea\_endcsname \_the\_font \_space #1\_relax
140   \_ea\_fontloaded \_csname \_tmp\_ea\_endcsname \_csname \_tmp\_endcsname
141 }
142 \_protected\_def \_resizethefont{\_newcurrfontsize\_sizespec}
143
144 \_public \newcurrfontsize \resizethefont ;

```

fonts-resize.opm

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.

```

153 \_def\_reloading{\_loadf{rm}\_tenrm \_loadf{bf}\_tenbf
154   \_loadf{it}\_tenit \_loadf{bi}\_tenbi
155 }
156 \_def\_loadf#1#2{\_sdef\_tryload#1}{\_ifmode \_else \_resizefont{#1}#2\_fi}}
157 \_def\_tryloadtt{\_resizefont{tt}\_tentt} % only in TFM mode
158
159 \_let\_tryloadrm=\_relax
160 \_let\_tryloadbf=\_relax
161 \_let\_tryloadit=\_relax
162 \_let\_tryloadbi=\_relax

```

fonts-resize.opm

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.

```

171 \_protected \_def \_currvar{\_cs\_currvar:\_ea \_csstring \_the\_font}}
172 \_sdef{\currvar:\_tenrm}{\_rm}
173 \_sdef{\currvar:\_tenbf}{\_bf}
174 \_sdef{\currvar:\_tenit}{\_it}
175 \_sdef{\currvar:\_tenbi}{\_bi}
176 \_sdef{\currvar:\_tentt}{\_tt}
177 \_public \currvar ;

```

fonts-resize.opm

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 shown 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 have the same meaning because of the `\_reversetfm` macro. The `\_wichtfm` expands this data line and apply `\_dowichtfm`. This macro selects the right result from the data line by testing with the current `\_optsize` value.

```

202 \_def\_regtfm #1 0 #2 *{\_ea\_def \_csname \_#1:reg\_endcsname{#2 16380 \_relax}%
203   \_def\_tmpa{#1}\_reversetfm #2 * %
204 }
205 \_def\_reversetfm #1 #2 {% we need this data for \_setmathfamily
206   \_ea\_let\_csname \_#1:reg\_ea\_endcsname
207   \_csname \_tmpa:reg\_endcsname
208   \_if#2\_else \_ea\_reversetfm \_fi
209 }
210 \_def\_wichtfm #1{%
211   \_ifcsname \_#1:reg\_endcsname

```

fonts-resize.opm

```

212     \_ea\_ea\_ea \_dowhichtfm
213     \_csname _#1:reg\_ea\_endcsname
214     \_else
215     #1%
216     \_fi
217 }
218 \_def\_dowhichtfm #1 #2 {%
219     \_ifdim\_optsize<#2pt #1\_ea\_ignoretfm\_else \_ea\_dowhichtfm
220     \_fi
221 }
222 \_def\_ignoretfm #1\_relax{}

```

Optical sizes data for preloaded 8bit Latin Modern fonts:

fonts-resize.opm

```

228 \_regtfm lmr 0 ec-lmr5 5.5 ec-lmr6 6.5 ec-lmr7 7.5 ec-lmr8 8.5 ec-lmr9 9.5
229             ec-lmr10 11.1 ec-lmr12 15 ec-lmr17 *
230 \_regtfm lmbx 0 ec-lmbx5 5.5 ec-lmbx6 6.5 ec-lmbx7 7.5 ec-lmbx8 8.5 ec-lmbx9 9.5
231             ec-lmbx10 11.1 ec-lmbx12 *
232 \_regtfm lmri 0 ec-lmri7 7.5 ec-lmri8 8.5 ec-lmri9 9.5 ec-lmri10 11.1 ec-lmri12 *
233 \_regtfm lmtt 0 ec-lmtt8 8.5 ec-lmtt9 9.5 ec-lmtt10 11.1 ec-lmtt12 *
234
235 \_setfontsize {at10pt} % default font size

```

## 2.13 The Font Selection System

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

### 2.13.1 Terminology

We distinguish between

- *font switchers*, they are declared by the `\font` primitive or by `\fontlet` or `\fontdef` macros, they select given font.
- *variant selectors*, there are four basic variant selectors `\rm`, `\bf`, `\it`, `\bi`, there is a special selector `\currvar`. More variant selectors can be declared by the `\famvardef` macro. They select the font depending on the given variant and on the *font context* (i.e. on current family and on more features given by font modifiers). In addition, OpTeX defines `\tt` as variant selector independent of chosen font family. It selects typewriter-like font.
- *font modifiers* are declared in a family (`\cond`, `\caps`) or are “build in” (`\setfontsize{<size spec>}`, `\setff{<features>}`). They do appropriate change in the *font context* but do not select the font.
- *family selectors* (for example `\Termes`, `\LMfonts`), they are declared typically in the *font family files*. They enable to switch between font families, they do appropriate change in the *font context* but do not select the font.

These commands set their values locally. When the TeX group is left then the 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 context* is a set of macro values that will affect the selection of real font when the variant selector is processed. It includes the value of *current family*, current font size, and more values stored by font modifiers.

The *family context* is the current family value stored in the font context. The variant selectors declared by `\famvardef` and font modifiers declared by `\moddef` are dependent on the *family context*. They can have the same names but different behavior 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 users can give more declarations in the file `fams-local.opm`.

### 2.13.2 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 2.13.11. Font modifiers and variant selectors may be declared here. The loaded family is set as current and `\rm` variant selector is processed.

The available declared font modifiers and declared variant selectors are listed in the log file when the 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 arbitrarily combined (in arbitrary order) and if the font family disposes of all such sub-variants then the desired font is selected (after variant selector is used). On the other hand, there are font modifiers that 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 a variant selector. It keeps the current variant and the font of such variant is reloaded with respect to the current font context by the previously given family selector and font modifiers.

You can use the `\setfontsize` *`{\sizen spec}`* command in the same sense as other font modifiers. It saves information about font size to the font context. See section 2.12. Example:

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

A much more comfortable way to resize fonts is using OPmac-like commands `\typosize` and `\typoscale`. These commands prepare the right sizes for math fonts too and they re-calculate many internal parameters like `\baselineskip`. See section 2.17 for more information.

### 2.13.3 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.

### 2.13.4 Declaring font commands

Font commands can be font switches, variant selectors, font modifiers, family selectors and defined font macros doing something with fonts.

- Font switches can be declared by `\font` primitive (see section 2.12.2) or by `\fontlet` command (see section 2.12.4) or by `\fontdef` command (see sections 2.13.5 and 2.12.3). When the font switches are used then they select the given font independently of the current font context. They can be used in `\output` routine (for example) because we need to set fixed fonts in headers and footers.
- Variant selectors are `\rm`, `\bf`, `\it`, `\bi`, `\tt` and `\currvar`. More variant selectors can be declared by `\famvardef` command. They select a font dependent on the current font context, see section 2.13.6. The `\tt` selector is documented in section 2.13.7.
- Font modifiers are “build in” or declared by `\moddef` command. They do modifications in the font context but don't select any font.
  - “build-in” font modifiers are `\setfontsize` (see section 2.12), `\setff` (see section 2.13.9), `\setfontcolor`, `\setletterspace` and `\setwordspace` (see section 2.13.10). They are independent of font family.
  - Font modifiers declared by `\moddef` depend on the font family and they are typically declared in font family files, see section 2.13.11.

- Family selectors set the given font family as current and re-set data used by the family-dependent font modifiers to initial values and to the currently used modifiers. They are declared in font family files by `\_famdecl` macro, see section 2.13.11.
- Font macros can be defined arbitrarily by `\def` primitive by users. See an example in section 2.13.8.

All declaration commands mentioned here: `\font`, `\fontlet`, `\fontdef`, `\famvardef`, `\moddef`, `\_famdecl` and `\def` make local assignment.

### 2.13.5 The `\fontdef` declarator in detail

The general format for `\fontdef` usage is

```
\fontdef\⟨font switch⟩ { \⟨family selector⟩ ⟨font modifiers⟩ \⟨variant selector⟩ }
```

where `\⟨family selector⟩` and `⟨font modifiers⟩` are optional and `\⟨variant selector⟩` is mandatory.

The `\fontdef` does the following steps. It pushes the current font context to a stack, it does modifications of the font context by given `\⟨family selector⟩` and/or `⟨font modifiers⟩` and it finds the real font by `\⟨variant selector⟩`. This font is not selected but it is assigned to the declared `\⟨font switch⟩` (like `\font` primitive does it). Finally, `\fontdef` pops the font context stack, so the current font context is the same as it was before `\fontdef` is used.

More about `\fontdef` command including examples is written in section 2.12.3.

### 2.13.6 The `\famvardef` declarator

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

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

where `\⟨family selector⟩` and `⟨font modifiers⟩` are optional and `\⟨variant selector⟩` is mandatory. The `\⟨new variant selector⟩` should be used in the same sense as `\rm`, `\bf` etc. It can be used as the final command in next `\fontdef` or `\famvardef` declarators too. When the `\⟨new variant selector⟩` is used in the normal text then it does the following steps: pushes current font context to a stack, modifies font context by declared `\⟨family selector⟩` and/or `⟨font modifiers⟩`, runs following `\⟨variant selector⟩`. This last one selects a real font. Then pops the font context stack. The new font is selected but the font context has its original values. This is main difference between `\famvardef\foo{...}` and `\def\foo{...}`.

Moreover, the `\famvardef` creates the `\⟨new variant selector⟩` family dependent. When the selector is used in another family context than it is defined then a warning is printed on the terminal “`⟨var selector⟩` is undeclared in the current family” and nothing happens. But you can declare the same variant selector by `\famvardef` macro in the context of a new family. Then the same command may 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. Then you can declare:

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

Now, you can use six independent variant selectors `\rm`, `\bf`, `\it`, `\bi`, `\mf` 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 a 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.



The variant selectors (declared by `\famvardef`) or font modifiers (declared by `\moddef`) are (typically) control sequences in user name space (`\mf`, `\caps`). They are most often declared in font family files and they are loaded by `\fontfam`. A conflict with such names in user namespace can be here. For example: if `\mf` is defined by a user and then `\fontfam[Roboto]` is used then `\famvardef\mf` is performed for Roboto family and the original meaning of `\mf` is lost. But OpTeX prints warning about it. There are two cases:

```
\def\mf{Metafont}
\fontfam[Roboto] % warning: "The \mf is redefined by \famvardef" is printed
or
\fontfam[Roboto]
\def\mf{Metafont} % \mf variant selector redefined by user, we suppose that \mf
                  % is used only in the meaning of "Metafont" in the document.
```

### 2.13.7 The `\tt` variant selector

`\tt` is an additional special variant selector which is defined as “select typewriter font independently of the current font family”. By default, the typewriter font-face from LatinModern font family is used.

The `\tt` variant selector is used in OpTeX internal macros `\_ttfont` (verbatim texts) and `\_urlfont` (printing URL’s).

You can redefine the behavior of `\tt` by `\famvardef`. For example:

```
\fontfam[Cursor]
\fontfam[Heros]
\fontfam[Termes]
\famvardef\tt{\Cursor\setff{-liga;-tlig}\rm}

Test in Termes: {\tt text}. {\Heros\rm Test in Heros: {\tt text}}.
Test in URL \url{http://something.org}.
```

You can see that `\tt` stay family independent. This is a special feature only for `\tt` selector. New definition is used in `\_ttfont` and `\_urlfont` too. It is recommended to use `\setff{-liga;-tlig}` to suppress the ligatures in typewriter fonts.

If Unicode math font is loaded then the `\tt` macro selects typewriter font-face in math mode too. This face is selected from used Unicode math font and it is independent of `\famvardef\tt` declaration.

### 2.13.8 Font commands defined by `\def`

Such font commands can be used as fonts selectors for titles, footnotes, citations, etc. Users can define them.

The following example shows how to define a “title-font selector”. Titles are not only bigger but they are typically in the bold variant. When a user puts `{\it...}` into the title text 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 Title in bold 14pt font and {\it bold 14pt italics} too}
```

OpTeX defines similar internal commands `\_titfont`, `\_chapfont`, `\_secfont` and `\_seccfont`, see section 2.26. The commands `\typosize` and `\boldify` are used in these macros. They set the math fonts to given size too and they are defined in section 2.17.

### 2.13.9 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 support the given font feature then the 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 create a new modifier, just use `{\caps\currvar 012345}`.

### 2.13.10 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>} % 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 2.12. 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 the 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 the backslash) instead of the hexadecimal specification. The empty parameter `<color>` means default black color.

These colors of fonts are implemented using LuaTeX internal font feature. This is different approach than using colors in section 2.20.

`\setletterspace {<number>}` specifies the letter spacing of the font. The `<number>` is a 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 the default.

`\setwordspace {<scaling>}` scales the default interword space (defined in the font) and its stretching and shrinking parameters by given `<scaling>` factor. For example `\setwordspace{2.5}` multiplies interword space by 2.5. `\setwordspace` can use different multiplication factors if its parameter is in the format `{<default>/<stretching>/<shrinking>}`. For example, `\setwordspace{/1/2.5/1}` enlarges only stretching 2.5 times.

You can use `\setff` with other font features provided by LuaTeX and luaotfload package (see documentation of luaotfload package for more information):

```

\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
\setff{upper}\rm        % to uppercase (lower=lowecase) conversion at font level
\setff{fallback=name}\rm % use fonts from a list given by name if missing chars

```

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

All mentioned font modifiers (except for `\setfontsize`) work only with Unicode fonts loaded by `\fontfam`.

### 2.13.11 How to create the font family file

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

```

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;\ffonum; }
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 \ifmathloading
23   \loadmath {[FiraMath-Regular]}
24   \addto\normalmath{\loadumathfamily 5 {xitsmath-regular}{}} }
25   \addto\boldmath {\loadumathfamily 5 {xitsmath-bold}{}} }
26   \addto\frak{\fam5 }\addto\cal{\fam5 }
27   \normalmath
28   \wterm{MATH-FONT(5): "[XITSMath-Regular/Bold]" -- used for \string\cal, \string\frak}
29   % \bf, \bi from FiraMath:
30   \let\bsansvariables=\bfvariables
31   \let\bsansgreek=\bfGreek
32   \let\bsansgreek=\bfgreek
33   \let\bsansdigits=\bfdigits
34   \let\bisansvariables=\bivvariables
35   \let\bisansgreek=\bigreek
36 \fi

```

If you want to write such a font family file, you need to keep the 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 this 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 a 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.opm` 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 do 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` to start the family (it is mandatory).
- If math font should be loaded, use `\loadmath{\<math font>}`.

The `\fontnamegen` macro (declared in the last parameter of the `\famdecl`) must expand (at the expand processor level only) to a file name of the loaded font (or to its font name) and to optional font features appended. The Font Selection System uses this macro at the 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 1

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 are in `f-⟨family⟩.opm` files.

### Example 2

The `f-heros.opm` is listed here. Look at it. 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 3

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

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

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

### The `\_onlyif` macro

has the syntax `\_onlyif ⟨word⟩=⟨value-a⟩,⟨value-b⟩,...⟨value-n⟩: {⟨what⟩}`. It can be used inside `\moddef` as simple IF statement: the `⟨what⟩` is processed only if `⟨word⟩` has `⟨value-a⟩` or `⟨value-b⟩` ... or `⟨value-n⟩`. See `f-roboto.opm` for examples of usage of many `\_onlyif`'s.

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

### The `\moddef` macro

has the syntax `\moddef<modifier>{⟨what to do⟩}`. It does more things than simple `\_def`:

- The modifier macros are defined as `\_protected`.
- The modifier macros are defined as family-dependent.
- If the declared control sequence is defined already (and it is not a font modifier) then it is re-defined with a warning.

The `\famvardef` macro has the same features.

The `\(Familyselector)` is defined by the `\_famdecl` macro as:

```
\protected\def\(

```

The `\_initfontfamily`

must be run after modifier's declaration. It runs the `\(Familyselector)` and it runs `\_rm`, so the first font from the 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. in user namespace without `_` prefix. We assume that if a user re-defines them then he/she needs not them, so we have no problems. If the user's definition was done before loading the font family file then it is re-defined and `OpTeX` warns about it. See the end of section 2.13.4.

The name of `\(Familyselector)` should begin with an uppercase letter.

Please, look at [OpTeX font catalogue](#) before you will create your font family file and use the same names for analogical font modifiers (like `\cond`, `\caps`, `\sans`, `\mono` etc.) and for extra variant selectors (like `\lf`, `\li`, `\kf`, `\ki` etc. used in Roboto font family).

If you are using the same font modifier names to analogical font shapes then such modifiers are kept when the family is changed. For example:

```
\fontfam [Termes] \fontfam[Heros]
\caps\cond\it Caps+Cond italic in Heros \Termes\currvar Caps italic in Termes.
```

The family selector first resets all modifiers data by `\resetmod` and then it tries to run all currently used family-dependent modifiers before the family switching (without warnings if such modifier is unavailable in the new family). In this example, `\Termes` does `\resetmod` followed by `\caps\cond`. The `\caps` is applied and `\cond` is silently ignored in `Termes` family.

If you need to declare your private modifier (because it is used in other modifiers or macros, for example), use the name `\_wordM`. You can be sure that such a name does not influence the private namespace 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` or `f-poltawski.opm` where you can find the the example of the optical sizes declaration including documentation about it.

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

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

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

You can declare a collection of `\(size-dependent-template)s` for one given `\(internal-template)` 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-ouput-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 with an 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 us to use shortcuts instead of 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}
```

Another example of a font family with optical sizes is Antykwa Półtawskiego. The optical sizes feature is deactivated by default and it is switched on by  $\_osize$  font modifier:

```
3 \_famdecl [Poltawski] \Poltawski {Antykwa Półtawskiego, Polish traditional font family}
4     {\_light \noexpd \expd \eexpd \cond \ccond \osize \caps} {\rm \bf \it \bi} {}
5     {[antpolt-regular]}
6     {\_def\_fontnamegen {[antpolt\_liV\_condV-\_currV]\_capsV\_fontfeatures}}
7
8 \_wlog{\_detokenize{
9 Modifiers:^^J
10 \light ..... light weight, \bf,\bi=semibold^^J
11 \noexpd .... no expanded, no condensed, designed for 10pt size (default)^^J
12 \eexpd ..... expanded, designed for 6pt size^^J
13 \expd ..... semi expanded, designed for 8pt size^^J
14 \cond ..... semi condensed, designed for 12pt size^^J
15 \ccond ..... condensed, designed for 17pt size^^J
16 \osize ..... auto-sitches between \ccond \cond \noexpd \expd \eexpd by size^^J
17 \caps ..... caps & small caps^^J
18 }}
19
20 \_moddef \resetmod {\_fsetV li={},cond={},caps={} \_fvars regular bold italic bolditalic }
21 \_moddef \light     {\_fsetV li=lt }
22 \_moddef \noexpd    {\_fsetV cond={} }
23 \_moddef \eexpd     {\_fsetV cond=expd }
24 \_moddef \expd      {\_fsetV cond=semiexpd }
25 \_moddef \cond      {\_fsetV cond=semicond }
26 \_moddef \ccond     {\_fsetV cond=cond }
27 \_moddef \caps      {\_fsetV caps+=smcp;\_ffonum; }
28 \_moddef \nocaps    {\_fsetV caps={} }
29 \_moddef \osize      {\_def\_fontnamegen{[antpolt\_liV\_optname{x}-\_currV]:\_capsV\_fontfeatures}%
30                      \_regoptsizes x ? expd <7 semiexpd <9 {} <11.1 semicond <15 cond <*>}
31
32 \_initfontfamily % new font family must be initialized
```

`f-poltawski.opm`



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

Once you have prepared a font family file with the name `f-⟨famname⟩.opm` and TeX can see it in your filesystem then you can type `\fontfam[⟨famname⟩]` and the file is read, so the information about the font family is loaded. The name `⟨famname⟩` must be lowercase and without spaces in the file name `f-⟨famname⟩.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 `\fontfam [LMfonts]` and both commands load the file `f-lmfonts.opm`.

You can use your font file in sense of the previous paragraph without registering it. But problem is that such families are not listed when `\fontfam[?]` is used and it is not included in the font catalog when `\fontfam[catalog]` is printed. The list of families taken in the catalog and listed on the terminal is declared in two files: `fams-ini.opm` and `fams-local.opm`. The second file is optional. Users 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 `\_faminfo`, `\_famalias` and `\_famtext`. See the example from `fams-ini.tex`:

```

3 % Version <2020-02-28>. Loaded in format and secondly on demand by \fontfam[catalog]
4
5 \_famtext {Special name for printing a catalog :}
6
7 \_faminfo [Catalogue] {Catalogue of all registered font families} {fonts-catalog} {}
8 \_famalias [Catalog]
9
10 \_famtext {Computer Modern like family:}
11
12 \_famfrom {GUST}
13 \_faminfo [Latin Modern] {TeX Gyre fonts based on Coputer Modern} {f-lmfonts}
14   { -, \nbold, \sans, \sans\nbold, \slant, \ttset, \ttset\slant, \ttset\caps, %
15     \ttprop, \ttprop\bolder, \quotset: {\rm\bf\it\bi}
16     \caps: {\rm\it}
17     \ttligh, \ttcond, \dunhill: {\rm\it} \upital: {\rm} }
18 \_famalias [LMfonts] \_famalias [Latin Modern Fonts] \_famalias [lm]
19
20 \_famtext {TeX Gyre fonts based on Adobe 35:}
21
22 \_faminfo [Termes] {TeX Gyre Termes fonts based on Times} {f-termes}
23   { -, \caps: {\rm\bf\it\bi} }
24 \_famalias [Times]
25
26 \_faminfo [Heros] {TeX Gyre Heros fonts based on Helvetica} {f-heros}
27   { -, \caps, \cond, \caps\cond: {\rm\bf\it\bi} }
28 \_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 the catalog. It consists of one or more pairs `⟨mods⟩: {⟨vars⟩}`. For each pair: each modifier (separated by comma) is applied to each variant selector in `⟨vars⟩` and prepared samples are 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 the log file when all families are listed.

The `\_famfrom` saves the information about font type foundry or manufacturer or designer or license owner. You can use it before `\_faminfo` to print `\_famfrom` info into the catalog. The `\_famfrom` data is applied to each following declared families until new `\_famfrom` is given. Use `\_famfrom {}` if the information is not known.

### 2.13.14 Notices about extension of `\font` primitive

Unicode fonts are loaded by extended `\font` primitive. This extension is not activated in OpTeX by default, `\initunifonts` macro activates it. You need not use `\initunifonts` explicitly if `\fontfam` macro is used because `\fontfam` runs it internally.

The `\initunifonts` loads the Lua code from the Luaotfload package which implements the `\font` primitive extension. See its documentation `luaotfload-latex.pdf` for information about all possibilities of extended `\font` primitive.

The OpTeX format is initialized by `luatex` engine by default but you can initialize it by `luahtex` engine too. Then the `harfbuzz` library is ready to use for font rendering as an alternative to build-in font renderer from Luaotfload. The `harfbuzz` library gives more features for rendering Indic and Arabic scripts. But it is not used as default, you need to specify `mode=harf` in the `fontfeatures` field when `\font` is used. Moreover, when `mode=harf` is used, then you must specify `script` too. For example

```
\font\devafont=[NotoSansDevanagari-Regular]:mode=harf;script=dev2
```

If the `luahtex` engine is not used then `mode=harf` is ignored. See Luaotfload documentation for more information.

### 2.13.15 Implementation of the Font Selection System

```
3 \_codedecl \fontfam {Fonts selection system <2021-04-22>} % preloaded in format
```

`\initunifonts` macro extends LuaTeX's font capabilities, in order to be able to load Unicode fonts. Unfortunately, this part of OpTeX depends on `luaotfload` package, which adapts ConTeXt's generic font loader for plain TeX and L<sup>A</sup>T<sub>E</sub>X. `luaotfload` uses Lua functions from L<sup>A</sup>T<sub>E</sub>X's `luatexbase` namespace, we provide our own replacements. Moreover, `\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, `\initunifonts` sets itself to relax because we don't want to do this work twice.

`\_ttunifont` is default font for `\tt` variant if `f-lmfonts.opm` (or another font file where `\tt` is declared) is not loaded.

```
21 \_def\initunifonts {%
22   \directlua{%
23     require('luaotfload-main')
24     luaotfload.main()
25   }%
26   \_gdef\_rfskipatX ##1" ##2\_relax{"##1"}%
27   \_global\_let \doresizefont=\doresizeunifont
28   \_gdef\_tryloadtt {\fontdef\_tentt{\_let\_fontnamegen=\ttunifont\_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 \_def\_ttunifont{[lmmono10-regular]:\_fontfeatures-tlig;}
38
39 \_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 internal `\_f:<currfamily>.main.fam` command. The macro `\_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 font family context. The font family context is given by the current `\_currfamily` value and by the actual meaning of the `\_fontnamegen` macro. The `\_mathfaminfo` is saved for usage in the catalog.

```

56 \def\famdecl [#1]#2#3#4#5#6#7#8{%
57   \_initunifonts \uniaccents
58   \unless\ifcsname _f:\csstring#2:main.fam\endcsname
59     \_isfont{#7}\_iffalse
60     \opwarning{Family [#1] skipped, font "#7" not found}\_ea\_ea\_ea\_endinput \_else
61     \edef\_currfamily {\_csstring #2}\_def\_mathfaminfo{#6}%
62     \wterm {FONT: [#1] -- \_string#2 \_detokenize{(#3)^J mods:{#4} vars:{#5} math:{#6}}}%
63     \unless \ifx #2\_undefined
64     \opwarning{\_string#2 is redefined by \_string\_famdecl\_space[#1]}\_fi
65     \protected\edef#2{\_def\_noexpand\_currfamily{\_csstring #2}\_unexpanded{#8\_resetfam}}%
66     \_ea \_let \_csname _f:\_currfamily:main.fam\endcsname =#2%
67     \_fi
68   \_else \_csname _f:\csstring#2:main.fam\endcsname \_reloading \_rm \_ea \_endinput \_fi
69 }
70 \def\_initfontfamily{%
71   \_csname _f:\_currfamily:main.fam\endcsname \_reloading \_rm
72 }

```

**\\_regoptsizes**  $\langle internal-template \rangle$   $\langle left-output \rangle?$   $\langle right-output \rangle$   $\langle resizing-data \rangle$  prepares data for using by the **\\_optname**  $\langle internal-template \rangle$  macro. The data are saved to the **\\_oz:** $\langle internal-template \rangle$  macro. When the **\\_optname** is expanded then the data are scanned by the macro **\\_optnameA**  $\langle left-output \rangle?$   $\langle right-output \rangle$   $\langle mid-output \rangle$   $\langle size \rangle$  in the loop.

**\\_optfontalias**  $\{ \langle template A \rangle \} \{ \langle template B \rangle \}$  is defined as **\\_let** **\\_oz:** $\langle template A \rangle$  = **\\_oz:** $\langle template B \rangle$ .

```

85 \def\_regoptsizes #1 #2?#3 #4*{\_sdef{\_oz:#1}{#2?#3 #4* }}
86 \def\_optname #1{\_ifcsname \_oz:#1\endcsname
87   \_ea\_ea\_ea \_optnameA \_csname \_oz:#1\_ea\endcsname
88   \_else \_failedoptname{#1}\_fi
89 }
90 \def\_failedoptname #1{optname-fails:(#1)}
91 \def\_optnameA #1?#2 #3 <#4 {\_ifx*#4#1#3#2\_else
92   \_ifdim\_optsize<#4pt #1#3#2\_optnameC
93   \_else \_afterfifi \_optnameA #1?#2 \_fi\_fi
94 }
95 \def\_optnameC #1* {\_fi\_fi}
96 \def\_afterfifi #1\_fi\_fi{\_fi\_fi #1}
97 \def\_optfontalias #1#2{\_slet{\_oz:#1}{\_oz:#2}}

```

**\\_fvars**  $\langle rm-template \rangle$   $\langle bf-template \rangle$   $\langle it-template \rangle$   $\langle bi-template \rangle$  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  $\langle variant name \rangle$ . Usable only of standard four variants. Next variants can be declared by the **\famvardef** macro.

**\\_fsetV**  $\langle key \rangle = \langle value \rangle, \dots, \langle key \rangle = \langle value \rangle$  expands to **\def** **\\_keyV** $\{ \langle value \rangle \}$  in the loop.

**\\_onlyif**  $\langle key \rangle = \langle value-a \rangle, \langle value-b \rangle, \dots, \langle value-z \rangle$ :  $\{ \langle what \rangle \}$  runs  $\langle what \rangle$  only if the **\\_keyV** is defined as  $\langle value-a \rangle$  or  $\langle value-b \rangle$  or ... or  $\langle value-z \rangle$ .

**\\_prepcmmalist**  $ab, \{ \}, cd, \_end$ , expands to  $ab, , cd$ , (auxiliary macro used in **\\_onlyif**).

**\\_ffonum** is a shortcut for oldstyle digits font features used in font family files. You can do **\\_let** **\\_ffonum** = **\\_ignoreit** if you don't want to set old digits together with **\caps**.

```

123 \def\_fvars #1 #2 #3 #4 {%
124   \_sdef{\_fvar:rm}{#1}%
125   \_sdef{\_fvar:bf}{#2}%
126   \_ifx.#2\_slet{\_fvar:bf}{\_fvar:rm}\_fi
127   \_sdef{\_fvar:it}{#3}%
128   \_ifx.#3\_slet{\_fvar:it}{\_fvar:rm}\_fi
129   \_sdef{\_fvar:bi}{#4}%
130   \_ifx.#4\_slet{\_fvar:bi}{\_fvar:it}\_fi
131 }
132 \def\_currV{\_cs{\_fvar:\_whatresize}}
133 \def\_V{ }
134 \def \_fsetV #1 {\_fsetVa #1,=,}
135 \def \_fsetVa #1=#2,{\_isempty{#1}\_iffalse
136   \_ifx,#1\_else\_sdef{\_#1V}{#2}\_ea\_ea\_ea\_fsetVa\_fi\_fi
137 }
138 \def \_onlyif #1=#2:#3{%
139   \_edef\_act{\_noexpand\_isinlist{\_prepcmmalist #2,\_end},{\_cs{\_#1V}},}\_act

```

```

140 \_iftrue #3\_fi
141 }
142 \_def\_prepcommalist#1,{\_ifx\_end#1\_empty\_else #1,\_ea\_prepcommalist\_fi}
143 \_def\_ffonum {+onum;+pnum}

```

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

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

`\setnewmeaning <cs-name>=\_tmpa <by-what>` does exactly `\let <csname>=\_tmpa` but warning is printed if `<cs-name>` is defined already and it is not a variant selector or font modifier.

`\addtomodlist <font modifier>` adds given modifier to `\modlist` macro. This list is used after `\resetmod` when a new family is selected by a family selector, see `\resetfam` macro. This allows reinitializing the same current modifiers in the font context after the family is changed.

fonts-select.opm

```

173 \_def \_moddef #1#2{\_edef\_tmp{\_csstring#1}%
174 \_sdef\_f:\_currfamily:\_tmp}{\_addtomodlist#1#2\_reloading}%
175 \_protected \_edef \_tmpa{\_noexpand\_famdepend\_noexpand#1\_f:\_noexpand\_currfamily:\_tmp}}%
176 \_setnewmeaning #1=\_tmpa \_moddef
177 }
178 \_protected \_def \_resetmod {\_cs\_f:\_currfamily:resetmod}} % private variant of \resetmod
179 \_def \_resetfam{\_def\_addtomodlist##1{\_resetmod
180 \_edef \_modlist{\_ea}\_modlist
181 \_let\_addtomodlist=\_addtomodlistb
182 }
183 \_def \_currfamily{} % default current family is empty
184 \_def \_modlist{} % list of currently used modifiers
185
186 \_def \_addtomodlist#1{\_addto\_modlist#1}
187 \_let \_addtomodlistb=\_addtomodlist
188
189 \_def\_famdepend#1#2{\_ifcsname#2\_endcsname \_csname#2\_ea\_endcsname \_else
190 \_ifx\_addtomodlist\_addtomodlistb
191 \_opwarning{\_string#1 is undeclared in family "\_currfamily", ignored}\_fi\_fi
192 }
193 \_def\_setnewmeaning #1=\_tmpa#2{%
194 \_ifx #1\_undefined \_else \_ifx #1\_tmpa \_else
195 \_opwarning{\_string#1 is redefined by \_string#2}%
196 \_fi\_fi
197 \_let#1=\_tmpa
198 }
199 \_public \_moddef ;

```

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

- `\def \_tryload:<currfam>:<XX> {\fontdef \_ten<XX> {<data>}}` loads font `\_ten<XX>`,
- `\protected\def \<XX> {\famdepend \<XX> {\_f:<currfam>:<XX>}}`,
- `\def \_f:<currfam>:<XX> {\_tryload:<currfam>:<XX>\_ten<XX>}` keeps family dependent definition,
- `\def \_currvar:\_ten<XX> {\<XX>}` in order to the `\currvar` macro work correctly.

`\famvardef\tt` behaves somewhat differently: it doesn't re-define the `\tt` macro which is defined as `\_tryloadtt \_tentt` in sections 2.14 and 2.16.2. It only re-defines the internal `\_tryloadtt` macro.

fonts-select.opm

```

218 \_def\_famvardef#1{\_edef\_tmp{\_csstring#1}%
219 \_ea\_famvardefA \_ea#1\_csname \_ten\_tmp\_ea\_endcsname
220 \_csname \_tryload:\_currfamily:\_tmp\_endcsname
221 }
222 \_def\_famvardefA #1#2#3#4{% #1=\_XX #2=\_tenXX #3=\_tryload:currfam:XX #4=data
223 \_isinlist{\_rm\_bf\_it\_bi\_currvar\_currvar}#1\_iftrue

```

```

224     \_opwarning{\_string\famvardef:
225         You cannot re-declare standard variant selector \_string#1}%
226     \_else
227         \_def#3{\_fontdef#2{#4}}%
228         \_protected\_edef\_tmpa{\_noexpand\_famdepend\_noexpand#1{\_f:\_noexpand\_currfamily:\_tmp}}%
229         \_ifx #1\_tt \_let\_tryloadtt=#3\_else \_setnewmeaning #1=\_tmpa \famvardef \_fi
230         \_sdef{\_f:\_currfamily:\_tmp}{#3#2}%
231         \_sdef{\_currvar:\_csstring#2}{#1}%
232     \_fi
233 }
234 \_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 load this file twice. The `\_listfamnames` macro prints registered font families to the terminal and to the log file.

```

                                                                    fonts-select.opm
252 \_def\_fontfam[#1]{%
253     \_lowercase{\_edef\_famname{\_ea\_removespaces #1 {} }}%
254     \_isfile {f-\_famname.opm}\_iftrue \_opinput {f-\_famname.opm}%
255     \_else
256         \_tryloadfamslocal
257         \_edef\_famfile{\_trycs{\_famf:\_famname}{}}%
258         \_ifx\_famfile\_empty \_listfamnames
259         \_else \_opinput {\_famfile.opm}%
260     \_fi\_fi
261 }
262 \_def\_tryloadfamslocal{%
263     \_isfile {fams-local.opm}\_iftrue
264         \_opinput {fams-local.opm}\_famfrom={}%
265     \_fi
266     \_let \_tryloadfamslocal=\_relax % need not to load fams-local.opm twice
267 }
268 \_def\_listfamnames {%
269     \_wterm{==== List of font families =====}
270     \_begingroup
271         \_let\_famtext=\_wterm
272         \_def\_faminfo [##1]##2##3##4{%
273             \_wterm{ \_space\_noexpand\fontfam [##1] -- ##2}%
274             \_let\_famalias=\_famaliasA}%
275             \_opinput {fams-ini.opm}%
276             \_isfile {fams-local.opm}\_iftrue \_opinput {fams-local.opm}\_fi
277             \_message{^^J}%
278         \_endgroup
279     }
280     \_def\_famaliasA{\_message{ \_space\_space\_space\_space -- alias:}
281         \_def\_famalias[##1]{\_message{[##1]}}\_famalias
282     }
283     \_public \fontfam ;

```

When the `fams-ini.opm` or `fams-local.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.

`\_famfrom` declares type foundry or owner or designer of the font family. It can be used in `fams-ini.opm` or `fams-local.opm` and it is printed in the font catalog.

fonts-select.opm

```

306 \_def\_famtext #1{}
307 \_def\_faminfo [#1]#2#3#4{%
308   \_lowercase{\_edef\_tmp{\_ea\_removespaces #1 {} }}%
309   \_sdef{\_famf:\_tmp}{#3}%
310   \_def\_famfile{#3}%
311 }
312 \_def\_famalias [#1]{%
313   \_lowercase{\_edef\_tmpa{\_ea\_removespaces #1 {} }}%
314   \_sdef{\_famf:\_tmpa\_ea}\_ea{\_famfile}%
315 }
316 \_newtoks\_famfrom
317 \_input fams-ini.opm
318 \_let\_famfile=\_undefined
319 \_famfrom={}

```

When the `\fontfam[catalog]` is used then the file `fonts-catalog.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

```

332 \_newtoks \_catalogsample
333 \_newtoks \_catalogmathsample
334 \_newtoks \_catalogonly
335 \_newtoks \_catalogexclude
336 \_catalogsample={ABCDabcd Qsty fi fl áéíóúüű řžč ÁÉÍÓÚ ŘŽČ 0123456789}
337
338 \_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

```

348 \_def \_defaultfontfeatures {+tlig;}
349 \_def \_setff #1{%
350   \_ifx~#1~\_let \_fontfeatures=\_defaultfontfeatures
351   \_else \_edef\_fontfeatures{\_fontfeatures #1;}\_fi
352   \_reloading
353 }
354 \_setff {} % default font features: +tlig;
355 \_def\_removefeature #1{%
356   \_isinlist\_fontfeatures{#1}\_iftrue
357     \_def\_tmp ##1#1##2;##3\_relax{\_def\_fontfeatures{##1##3}}%
358     \_ea \_tmp \_fontfeatures \_relax
359   \_fi
360 }
361 \_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

```

373 \_def\_savedfontcolor{}
374 \_def\_savedletterspace{}
375 \_def\_savedwsp{}
376
377 \_def \_setfontcolor #1{\_removefeature{color=}%
378   \_edef\_tmp{\_calculatefontcolor{#1}}%
379   \_ifx\_tmp\_empty \_else \_edef\_fontfeatures{\_fontfeatures color=\_tmp;}\_fi
380   \_reloading
381 }
382 \_def \_setletterspace #1{\_removefeature{letterspace=}%
383   \_if~#1~\_else \_edef\_fontfeatures{\_fontfeatures letterspace=#1;}\_fi

```



```

384 \_reloading
385 }
386 \_def \_setwordspace #1{%
387 \_if^#1^\_def\_setwsp##1{\\_removefeature{+Ws}%
388 \_else \_def\_setwsp{\_setwspA#1/}\_setff{+Ws}\_fi
389 \_reloading
390 }
391 \_def\_setwsp #1{%
392 \_def\_setwspA #1{\_ifx/#1\_ea\_setwspB \_else\_afterfi{\_setwspC#1}\_fi}
393 \_def\_setwspB #1/#2/#3/#4{\_fontdimen2#4=#1\_fontdimen2#4%
394 \_fontdimen3#4=#2\_fontdimen3#4\_fontdimen4#4=#3\_fontdimen4#4}
395 \_def\_setwspC #1/{\_setwspB #1/#1/#1/}
396 }
397 \_def\_calculatefontcolor#1{\_trycs{fc:#1}{#1}} % you can define more smart macro ...
398 \_sdef{fc:red}{FF0000FF} \_sdef{fc:green}{00FF00FF} \_sdef{fc:blue}{0000FFFF}
399 \_sdef{fc:yellow}{FFFF00FF} \_sdef{fc:cyan}{00FFFFFF} \_sdef{fc:magenta}{FF00FFFF}
400 \_sdef{fc:white}{FFFFFFFF} \_sdef{fc:grey}{00000080} \_sdef{fc:lgrey}{00000025}
401 \_sdef{fc:black}{} % ... you can declare more colors...
402
403 \_public \setfontcolor \setletterspace \setwordspace ;

```

## 2.14 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 Unicode fonts, i.e. Unicode-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 the normal shape of all math symbols and `\boldmath` for the 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 cmli % 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 cmbsty % 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}

```

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 2.16.2.

`math-preload.opm`

```

57 \chardef\_bffam = 8
58 \chardef\_bifam = 9
59 %\_chardef\_tffam = 10
60 %\_chardef\_itfam = 11
61
62 \protected\_def \_rm {\_tryloadrm \_tenrm \_fam0 }
63 \protected\_def \_bf {\_tryloadbf \_tenbf \_fam\_bffam}
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 cmte 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 cmb 0 cmb5 5.5 cmb6 6.5 cmb7 7.5 cmb8 8.5 cmb9 9.5 cmb10 *
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 cmtt8 8.5 cmtt9 9.5 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 6 eufm7 8.5 eufm10 *
107 \regtfm eufb 0 eufb5 6 eufb7 8.5 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 bbsfams 0 ecxs0500 5.5 ecxs0600 6.5 ecxs0700 7.5 ecxs0800
117      8.5 ecxs0900 9.5 ecxs1000 11.1 ecxs1200 *
118 \_regtfm bbsfams 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 1pt by default.

`\_corrmsize`  $\langle factor \rangle$  $\langle space \rangle$  can be used just before `\_loadmathfamily` or `\_setmathfamily`. The  $\langle factor \rangle$  is decimal number, it denotes scale-factor “size of loaded math font in `\textstyle`: size of text font”. You can use it in `\_normalmath` or `\_boldmath` macros if you want to do a corrections (for example due to different ex-height in text and math font). The `\_corrmsize` is applied only to one following `\_loadmathfamily` or `\_setmathfamily`. If it is missing then the  $\langle factor \rangle$  is 1 for such math family (i.e. no size corrections).

math-preload.opm

```

148 \_def\_corrmsize#1 {\_ptmunit=#1\_ptunit} % for corrections of sizes in diferent fonts
149
150 \_def\_loadmathfamily #1 #2 {%
151   \_edef\_optsize\the\_optsize}%
152   \_optsize=\_sizemtext   \_font\_mF=\_whichfm{#2} at\_optsize \_textfont#1=\_mF
153   \_optsize=\_sizemscript \_font\_mF=\_whichfm{#2} at\_optsize \_scriptfont#1=\_mF
154   \_optsize=\_sizemsscript \_font\_mF=\_whichfm{#2} at\_optsize \_scriptscriptfont#1=\_mF
155   \_optsize=\_optsize\the\_ptmunit=\_ptunit
156 }
157 \_def\_setmathfamily #1 #2{\_let\_mF=#2%
158   \_edef\_optsize\the\_optsize}%
159   \_optsize=\_sizemtext   \_fontlet#2=#2 at\_optsize \_textfont#1=#2%
160   \_optsize=\_sizemscript \_fontlet#2=#2 at\_optsize \_scriptfont#1=#2%
161   \_optsize=\_sizemsscript \_fontlet#2=#2 at\_optsize \_scriptscriptfont#1=#2%
162   \_optsize=\_optsize\the\_ptmunit=\_ptunit \_let#2=\_mF
163 }
164 \_def\_setmathsizes[#1/#2/#3]{\_ptmunit=\_ptunit
165   \_def\_sizemtext{#1\_ptmunit}\_def\_sizemscript{#2\_ptmunit}%
166   \_def\_sizemsscript{#3\_ptmunit}%
167 }
168 \_newdimen\_ptunit   \_ptunit=1pt
169 \_newdimen\_ptmunit  \_ptmunit=1\_ptunit
170
171 \_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

```

180 \_def\_setmathdimens{% PlainTeX sets these dimens for 10pt size only:
181   \_delimitershortfall=0.5\_fontdimen6\_textfont3
182   \_nulldelimiterspace=0.12\_fontdimen6\_textfont3
183   \_scriptspace=0.05\_fontdimen6\_textfont3
184   \_skewchar\_textfont1=127 \_skewchar\_scriptfont1=127
185   \_skewchar\_scriptscriptfont1=127
186   \_skewchar\_textfont2=48  \_skewchar\_scriptfont2=48
187   \_skewchar\_scriptscriptfont2=48
188   \_skewchar\_textfont6=127 \_skewchar\_scriptfont6=127
189   \_skewchar\_scriptscriptfont6=127
190 }

```

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

math-preload.opm

```

200 \_suppressfontnotfounderror=1
201 \_setmathsizes[10/7/5]\_normalmath
202 \_suppressfontnotfounderror=0

```

## 2.15 Math macros

math-macros.opm

```
3 \_codedecl \sin {Math macros plus mathchardefs <2021-04-25>} % preloaded in format
```

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

There is a problem: The `x_n` is tokenized as `x`, `_`, `n` and it works without problems. But `\int_a^b` is tokenized as `\int_a`, `^`, `b`. The control sequence `\int_a` isn't defined. We must write `\int _a^b`.

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

The `\mathsb` macro activates the rewriting rule `\<word>_<nonleter>` to `\<word> _<nonletter>` and `\<word>_<letter><nonletter>` to `\<word> _<letter><nonletter>` at input processor level. The `\mathsboff` deactivates it. You can ask by `\_ifmathsb` if this feature is activated or deactivated. By default, it is activated in the `\everyjob`, see section 2.1. Note, that the `\everyjob` is processed after the first line of the document is read, so the `\mathsb` is activated from the second line of the document.

math-macros.opm

```
29 \catcode\_ = 8 \let\sb = _
30 \catcode\_ = 13 \let _ = \sb
31 \catcode\_ = 11
32 \private \sb ;
33
34 \newif\\_ifmathsb \_mathsbfalse
35 \def \_mathsb {%
36 \directlua{
37 callback.add_to_callback("process_input_buffer",
38 function (str)
39 return string.gsub(str.." ", "(\_nbb[a-zA-Z]+)_[a-zA-Z]?[^\_a-zA-Z]", "\_pcent 1 \_pcent 2")
40 end, "\_mathsb") }%
41 \global\_mathsbtrue
42 }
43 \def \_mathsboff {%
44 \directlua{ callback.remove_from_callback("process_input_buffer", "\_mathsb") }%
45 \global \_mathsbfalse
46 }
47 \public \mathsboff \mathsb ;
```

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

math-macros.opm

```
55 \_mathcode\^{}="2201 % \cdot
56 \_mathcode\^{}A="3223 % \downarrow
57 \_mathcode\^{}B="010B % \alpha
58 \_mathcode\^{}C="010C % \beta
59 \_mathcode\^{}D="225E % \land
60 \_mathcode\^{}E="023A % \lnot
61 \_mathcode\^{}F="3232 % \in
62 \_mathcode\^{}G="0119 % \pi
63 \_mathcode\^{}H="0115 % \lambda
64 \_mathcode\^{}I="010D % \gamma
65 \_mathcode\^{}J="010E % \delta
66 \_mathcode\^{}K="3222 % \uparrow
67 \_mathcode\^{}L="2206 % \pm
68 \_mathcode\^{}M="2208 % \oplus
69 \_mathcode\^{}N="0231 % \infty
70 \_mathcode\^{}O="0140 % \partial
71 \_mathcode\^{}P="321A % \subset
72 \_mathcode\^{}Q="321B % \supset
73 \_mathcode\^{}R="225C % \cap
74 \_mathcode\^{}S="225B % \cup
75 \_mathcode\^{}T="0238 % \forall
76 \_mathcode\^{}U="0239 % \exists
77 \_mathcode\^{}V="220A % \otimes
78 \_mathcode\^{}W="3224 % \leftrightarrow
79 \_mathcode\^{}X="3220 % \leftarrow
80 \_mathcode\^{}Y="3221 % \rightarrow
81 \_mathcode\^{}Z="8000 % \ne
```

```

82 \_mathcode\`^^["2205 % \diamond
83 \_mathcode\`^^\="3214 % \le
84 \_mathcode\`^^\="3215 % \ge
85 \_mathcode\`^^\="3211 % \equiv
86 \_mathcode\`^^\_="225F % \lor
87 \_mathcode\`\_="8000 % \space
88 \_mathcode\`!\="5021
89 \_mathcode\`'\="8000 % \prime
90 \_mathcode\`(\="4028
91 \_mathcode\`)="5029
92 \_mathcode\`*="2203 % \ast
93 \_mathcode\`+="202B
94 \_mathcode\`,`="613B
95 \_mathcode\`-="2200
96 \_mathcode\`\_="013A
97 \_mathcode\`/= "013D
98 \_mathcode\`:= "303A
99 \_mathcode\`;="603B
100 \_mathcode\`<="313C
101 \_mathcode\`== "303D
102 \_mathcode\`>="313E
103 \_mathcode\`?="503F
104 \_mathcode\`["405B
105 \_mathcode\`\\="026E % \backslash
106 \_mathcode\`]="505D
107 \_mathcode\`\_="8000 % math-active subscript
108 \_mathcode\`{"4266
109 \_mathcode\`|="026A
110 \_mathcode\`}="5267
111 \_mathcode\`^^?="1273 % \smallint
112
113 \_delcode\`(\="028300
114 \_delcode\`)="029301
115 \_delcode\`["05B302
116 \_delcode\`]="05D303
117 \_delcode\`<="26830A
118 \_delcode\`>="26930B
119 \_delcode\`/= "02F30E
120 \_delcode\`|="26A30C
121 \_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`

```

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

```

306 \_protected\_def\log {\_mathop{\_rm log}\_nolimits}

```

```

307 \protected\def\lg {\mathop{\rm lg}\nolimits}
308 \protected\def\ln {\mathop{\rm ln}\nolimits}
309 \protected\def\lim {\mathop{\rm lim}\nolimits}
310 \protected\def\limsup {\mathop{\rm lim}\nolimits\thinsup}
311 \protected\def\liminf {\mathop{\rm lim}\nolimits\thinsub}
312 \protected\def\sin {\mathop{\rm sin}\nolimits}
313 \protected\def\arcsin {\mathop{\rm arcsin}\nolimits}
314 \protected\def\sinh {\mathop{\rm sinh}\nolimits}
315 \protected\def\cos {\mathop{\rm cos}\nolimits}
316 \protected\def\arccos {\mathop{\rm arccos}\nolimits}
317 \protected\def\cosh {\mathop{\rm cosh}\nolimits}
318 \protected\def\tan {\mathop{\rm tan}\nolimits}
319 \protected\def\arctan {\mathop{\rm arctan}\nolimits}
320 \protected\def\tanh {\mathop{\rm tanh}\nolimits}
321 \protected\def\cot {\mathop{\rm cot}\nolimits}
322 \protected\def\coth {\mathop{\rm coth}\nolimits}
323 %\protected\def\sec {\mathop{\rm sec}\nolimits} % \sec is section
324 \protected\def\csc {\mathop{\rm csc}\nolimits}
325 \protected\def\max {\mathop{\rm max}\nolimits}
326 \protected\def\min {\mathop{\rm min}\nolimits}
327 \protected\def\sup {\mathop{\rm sup}\nolimits}
328 \protected\def\inf {\mathop{\rm inf}\nolimits}
329 \protected\def\arg {\mathop{\rm arg}\nolimits}
330 \protected\def\ker {\mathop{\rm ker}\nolimits}
331 \protected\def\dim {\mathop{\rm dim}\nolimits}
332 \protected\def\hom {\mathop{\rm hom}\nolimits}
333 \protected\def\det {\mathop{\rm det}\nolimits}
334 \protected\def\exp {\mathop{\rm exp}\nolimits}
335 \protected\def\Pr {\mathop{\rm Pr}\nolimits}
336 \protected\def\gcd {\mathop{\rm gcd}\nolimits}
337 \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 a more readable form.

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

```

347 \let\sp=\public \sp ;
348 % \sb=, defined at beginning of this file
349
350 \def\thinskip {\mskip\thinmuskip}
351 \protected\def\relax_ifmode {\thinskip \else \thinspace \fi}
352 \protected\def\medskip {\mskip\medmuskip} \let\medsk = \>
353 \protected\def\thickskip {\mskip\thickmuskip} \let\thicksk = \;
354 \protected\def\thinneg {\mskip-\thinmuskip} \let\thinneg = \!
355 %\def*{\discretionary{\thinspace\the\textfont2\char2}{}} % obsolete

```

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

```

361 {\catcode\active \gdef{\_bgroup\_primes}} % primes dance
362 \def\_primes{\_prime\_isnextchar{\_primesA}%
363 \_isnextchar{\_primesB}{\_egroup}}
364 \def\_primesA #1{\_primes}
365 \def\_primesB #1#2{\_egroup}
366 \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.

```

375 %{\catcode\active \gdef{\_Z{\not=}} % \_Z is like \ne in math %obsolete
376
377 \def\_scalebig#1#2{{\_left#1\_vbox to#2\_fontdimen6\_textfont1}{%
378 \_kern-\_nulldelimiterspace\_right.}}
379 \protected\def\_big#1{\_scalebig{#1}{.85}}
380 \protected\def\_Big#1{\_scalebig{#1}{1.15}}
381 \protected\def\_bigg#1{\_scalebig{#1}{1.45}}
382 \protected\def\_Bigg#1{\_scalebig{#1}{1.75}}
383 \public \big \Big \bigg \Bigg ;
384

```



```

385 \protected\def\bigl{\mathopen\big}
386 \protected\def\bigm{\mathrel\big}
387 \protected\def\bigr{\mathclose\big}
388 \protected\def\Bigl{\mathopen\Big}
389 \protected\def\Bigm{\mathrel\Big}
390 \protected\def\Bigr{\mathclose\Big}
391 \protected\def\biggl{\mathopen\bigg}
392 \protected\def\biggm{\mathrel\bigg}
393 \protected\def\biggr{\mathclose\bigg}
394 \protected\def\Biggl{\mathopen\Bigg}
395 \protected\def\Biggm{\mathrel\Bigg}
396 \protected\def\Biggr{\mathclose\Bigg}
397 \public \bigl \bigm \bigr \Bigl \Bigm \Bigr \biggl \biggm \biggr \Biggl \Biggm \Biggr ;

```

Math relations defined by the `\jointrel` plain TeX macro:

math-macros.opm

```

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

```

`\ldots`, `\cdots`, `\vdots`, `\ddots` from plain TeX

math-macros.opm

```

429 \mathchardef\ldotp="613A % ldot as a punctuation mark
430 \mathchardef\cdotp="6201 % cdot as a punctuation mark
431 \mathchardef\colon="603A % colon as a punctuation mark
432 \public \ldotp \cdotp \colon ;
433
434 \protected\def\ldots{\mathinner{\ldotp\ldotp\ldotp}}
435 \protected\def\cdots{\mathinner{\cdotp\cdotp\cdotp}}
436 \protected\def\vdots{\vbox{\baselineskip=.4em \lineskiplimit=\zo
437 \kern.6em \hbox{.}\hbox{.}\hbox{.}}}
438 \protected\def\ddots{\mathinner{%
439 \mkern1mu\raise.7em\vbox{\kern.7em\hbox{.}}\mkern2mu
440 \raise.4em\hbox{.}\mkern2mu\raise.1em\hbox{.}\mkern1mu}}
441
442 \public \ldots \cdots \vdots \ddots ;

```

`\adots` inspired by plain TeX

math-macros.opm

```

448 \protected\def\adots{\mathinner{%
449 \mkern1mu\raise.1em\hbox{.}\mkern2mu
450 \raise.4em\hbox{.}\mkern2mu\raise.7em\vbox{\kern.7em\hbox{.}}\mkern1mu}}
451
452 \public \adots ;

```

Math accents (encoding dependent declarations).

math-macros.opm

```

458 \protected\def\acute{\mathaccent"7013 }
459 \protected\def\grave{\mathaccent"7012 }
460 \protected\def\ddot{\mathaccent"707F }
461 \protected\def\tilde{\mathaccent"707E }
462 \protected\def\bar{\mathaccent"7016 }

```

```

463 \protected\def\breve{\mathaccent"7015 }
464 \protected\def\check{\mathaccent"7014 }
465 \protected\def\hat{\mathaccent"705E }
466 \protected\def\vec{\mathaccent"017E }
467 \protected\def\dot{\mathaccent"705F }
468 \protected\def\widetilde{\mathaccent"0365 }
469 \protected\def\widehat{\mathaccent"0362 }

```

`\_math`, `\skew`, `\overrightarrow`, `\overleftarrow`, `\overbrace`, `\underbrace` macros. The last four are redefined when Unicode math is loaded.

math-macros.opm

```

477 \def\_math{\mathsurround\_zo}
478 \protected\def\_skew #1#2#3{{\_muskip0=#1mu\_divide\_muskip0=by2 \_mkern\_muskip0
479   #2{\_mkern-\_muskip0{#3}\_mkern\_muskip0}\_mkern-\_muskip0}{}}
480 \protected\def\_overrightarrow #1{\_vbox{\_math\_ialign{##\_crrc
481   \_rightarrowfill\_crrc\_noalign{\_kern-.1em \_nointerlineskip}
482   $\_hfil\_displaystyle{#1}\_hfil$\_crrc}}}}
483 \protected\def\_overleftarrow #1{\_vbox{\_math\_ialign{##\_crrc
484   \_leftarrowfill\_crrc\_noalign{\_kern-.1em \_nointerlineskip}
485   $\_hfil\_displaystyle{#1}\_hfil$\_crrc}}}}
486 \protected\def\_overbrace #1{\_mathop{%
487   \_vbox{\_math\_ialign{##\_crrc\_noalign{\_kern.3em}
488   \_downbracefill\_crrc\_noalign{\_kern.3em \_nointerlineskip}
489   $\_hfil\_displaystyle{#1}\_hfil$\_crrc}}}\_limits}
490 \protected\def\_underbrace #1{\_mathop{\_vtop{\_math\_ialign{##\_crrc
491   $\_hfil\_displaystyle{#1}\_hfil$\_crrc\_noalign{\_kern.3em \_nointerlineskip}
492   \_upbracefill\_crrc\_noalign{\_kern.3em}}}\_limits}
493
494 \_public \overrightarrow \overleftarrow \overbrace \underbrace \skew ;

```

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

math-macros.opm

```

500 \protected\def\lmoustache{\delimiter"437A340 } % top from (, bottom from )
501 \protected\def\rmoustache{\delimiter"537B341 } % top from ), bottom from (
502 \protected\def\lgroup{\delimiter"462833A } % extensible ( with sharper tips
503 \protected\def\rgroup{\delimiter"562933B } % extensible ) with sharper tips
504 \protected\def\arrowvert{\delimiter"26A33C } % arrow without arrowheads
505 \protected\def\Arrowvert{\delimiter"26B33D } % double arrow without arrowheads
506 \protected\def\bracevert{\delimiter"77C33E } % the vertical bar that extends braces
507 \protected\def\Vert{\delimiter"26B30D } \let\|= \Vert
508 \protected\def\vert{\delimiter"26A30C }
509 \protected\def\uparrow{\delimiter"3222378 }
510 \protected\def\downarrow{\delimiter"3223379 }
511 \protected\def\updownarrow{\delimiter"326C33F }
512 \protected\def\Uparrow{\delimiter"322A37E }
513 \protected\def\Downarrow{\delimiter"322B37F }
514 \protected\def\Updownarrow{\delimiter"326D377 }
515 \protected\def\backslash{\delimiter"26E30F } % for double coset G\backslash H
516 \protected\def\langle{\delimiter"426830A }
517 \protected\def\rangle{\delimiter"526930B }
518 \protected\def\lbrace{\delimiter"4266308 } \let\_lbrace=\lbrace
519 \protected\def\rbrace{\delimiter"5267309 } \let\_rbrace=\rbrace
520 \protected\def\{{{\_ifmmode \_lbrace\_else\_char`\_fi}
521 \protected\def\}{\_ifmmode \_rbrace\_else\_char`\_fi}
522
523 \protected\def\rceil{\delimiter"5265307 }
524 \protected\def\lceil{\delimiter"4264306 }
525 \protected\def\rfloor{\delimiter"5263305 }
526 \protected\def\lfloor{\delimiter"4262304 }
527
528 \protected\def\choose{\_atopwithdelims()}
529 \protected\def\brack{\_atopwithdelims[]}
530 \protected\def\brace{\_atopwithdelims\_lbrace\_rbrace}
531
532 \protected\def\sqrt{\radical"270370 } \_public \sqrt ;

```

`\mathpalette`, `\vphantom`, `\hphantom`, `\phantom`, `\mathstrut`, and `\smash` macros from plain T<sub>E</sub>X.

math-macros.opm

```

539 \def\_mathpalette#1#2{\_mathchoice{#1\_displaystyle{#2}}%
540   {#1\_textstyle{#2}}{#1\_scriptstyle{#2}}{#1\_scriptscriptstyle{#2}}}

```

```

541 \newbox\rootbox
542 \protected\def\root#1\of{\setbox\rootbox
543 \hbox{$\mathscriptscriptstyle{#1}$}\mathpalette\rootA}
544 \def\rootA#1#2{\setbox0=\hbox{$\math#1\sqrt{#2}$}\dimen0=\ht0
545 \advance\dimen0by-\dp0
546 \mkern5mu\raise.6\dimen0\copy\rootbox \mkern-10mu\box0 }
547 \newif\ifvp \newif\ifhp
548 \protected\def\vphantom{\vptrue\hpfalse\phant}
549 \protected\def\hphantom{\vpfalse\hptrue\phant}
550 \protected\def\phantom{\vptrue\hptrue\phant}
551 \def\phant{\ifmode\def\next{\mathpalette\mathphant}%
552 \else\let\next=\makephant\fi\next}
553 \def\makephant#1{\setbox0\hbox{#1}\finphant}
554 \def\mathphant#1#2{\setbox0=\hbox{$\math#1{#2}$}\finphant}
555 \def\finphant{\setbox2=\null
556 \ifvp \ht2=\ht0 \dp2=\dp0 \fi
557 \ifhp \wd2=\wd0 \fi \hbox{\box2}}
558 \def\mathstrut{\vphantom{}}
559 \protected\def\smash{\relax % \relax, in case this comes first in \halign
560 \ifmode\def\next{\mathpalette\mathsmash}\else\let\next\makesmash
561 \fi\next}
562 \def\makesmash#1{\setbox0=\hbox{#1}\finsmash}
563 \def\mathsmash#1#2{\setbox0=\hbox{$\math#1{#2}$}\finsmash}
564 \def\finsmash{\ht0=\zo \dp0=\zo \hbox{\box0}}
565 \public \mathpalette \vphantom \hphantom \phantom \mathstrut \smash ;

```

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

math-macros.opm

```

572 \protected\def\cong{\mathrel{\mathpalette\overeq\sim}} % congruence sign
573 \def\overeq#1#2{\lower.05em\vbox{\lineskiplimit\maxdimen\lineskip=-.05em
574 \ialign{$\math#1\hfil#\hfil$\crrc#2\crrc=\crrc}}
575 \protected\def\notin{\mathrel{\mathpalette\cancel\in}}
576 \def\cancel#1#2{\math\oalign{$\hfil#1\mkern1mu\hfil$\crrc#1#2$}}
577 \protected\def\rightleftharpoons{\mathrel{\mathpalette\rlhp{}}}
578 \def\rlhp#1{\vcenter{\math\hbox{\oalign{\raise.2em
579 \hbox{$#1\rightarrow$}\crrc
580 $#1\leftarrow$}}}}
581 \protected\def\buildrel#1\over#2{\mathrel{\mathop{\kern\zo #2}\limits^{#1}}}
582 \protected\def\doteq{\buildrel\textstyle.\over=}
583 \private \in \sim ;
584 \public \cong \notin \rightleftharpoons \buildrel \doteq ;
585
586 \protected\def\bmod{\nonscript\mskip-\medmuskip\mkern5mu
587 \mathbin{\rm mod}\penalty900\mkern5mu\nonscript\mskip-\medmuskip}
588 \protected\def\pmod#1{\allowbreak\mkern18mu({\rm mod}\_thinsk\_thinsk#1)}
589 \public \bmod \pmod ;

```

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

math-macros.opm

```

599 \protected\def\matrix#1{\null\_thinsk
600 \edef\tmpa{\the\_numexpr \mathstyle/4\relax}% 0 0 1 1 1 1 2 2
601 \vcenter{\matrixbaselines\math
602 \ialign{\the\_lmfil$\_matrixstyle##$\hfil&\quad\the\_lmfil$\_matrixstyle##$\hfil\crrc
603 \mathstrut\crrc\noalign{\kern-\_baselineskip}
604 #1\crrc\mathstrut\crrc\noalign{\kern-\_baselineskip}}\_thinsk}
605
606 \def\matrixbaselines{\normalbaselines \def\_matrixstyle{}}%
607 \let\_matrixbaselines=\relax % \matrix inside matrix does not change size again
608 \ifcase\tmpa \or
609 \_baselineskip=.7\_baselineskip \def\_quad {\_hskip.7em\relax}%
610 \let\_matrixstyle=\scriptstyle
611 \or
612 \_baselineskip=.5\_baselineskip \def\_quad {\_hskip.5em\relax}%
613 \let\_matrixstyle=\scriptscriptstyle
614 \fi
615 }
616 \protected\def\pmatrix#1{\_left(\_matrix{#1}\_right)}

```

```

617
618 \public \matrix \pmatrix ;

```

The `\cases` and `\bordermatrix` macros are almost identical as in plain  $\TeX$ . You can simply re-define `\bordermatrix` with other delimiters using the common `\_bordermatrixwithdelims` macro.

math-macros.opm

```

626 \protected\long\def\_cases#1{\_left{\\_thinsk\_vcenter{\_normalbaselines\_math
627 \_ialign{###\_hfil$&\_quad{##\_unsskip}\_hfil\_crrcr#1\_crrcr}}\_right.}
628
629 \_newdimen\_ptrenwd
630 \_ptrenwd=8.75pt % width of the big left (
631 \protected\_def\_bordermatrix{\_bordermatrixwithdelims()}
632 \_def\_bordermatrixwithdelims#1#2#3{\_begingroup \_math
633 \_setbox0=\_vbox{\_bordermatrixA #3\_stopbmatrix}%
634 \_setbox2=\_vbox{\_unvcopy0 \_global\_setbox1=\_lastbox}%
635 \_setbox2=\_hbox{\_unhbox1 \_unskip\_global\_setbox1=\_lastbox}%
636 \_setbox2=\_hbox{$\_kern\_wd1 \_kern-\_ptrenwd\_left#1\_kern-\_wd1
637 \_global\_setbox1=\_vbox{\_box1 \_kern.2em}%
638 \_vcenter{\_kern-\_ht1 \_unvbox0 \_kern-\_baselineskip}\_thinsk\_right#2$}%
639 \_null\_thicksk\_vbox{\_kern\_ht1 \_box2}\_endgroup}
640 \_def\_bordermatrixA #1\_crrcr#2\_stopbmatrix{%
641 \_ialign{###\_hfil\_kern.2em\_kern\_ptrenwd&\_thinspace\_hfil###\_hfil
642 &\_quad\_hfil###\_hfil\_crrcr
643 \_omit\_strut\_hfil\_crrcr\_noalign{\_kern-\_baselineskip}%
644 #1\_crrcr\_noalign{\_kern.2em}#2\_crrcr\_omit\_strut\_crrcr}}
645
646 \public \cases \bordermatrix ;

```

The `\eqalign` macro behaves like in Plain  $\TeX$  by default. It creates the `\vcenter` in the math mode. The content 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. See section 4.4 in the [Typesetting Math with  \$\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}\$](#) . The `\baselineskip` value is set by the `\eqlines` parameter and math style by the `\eqstyle` parameter.

There are more possible columns than two (used in classical Plain  $\text{TeX}$ ): `rlcr``lcr``lc` etc. where `r` and `l` columns are without spaces and `c` column (if used) has space `\eqspace/2` at its both sides.

math-macros.opm

```

667 \_long\_def\_eqalign#1{\_null\_thinsk\_vcenter{\_the\_eqlines\_math
668 \_ialign{&\_hfil\_the\_eqstyle{##}$&\_the\_eqstyle{\_}\_}\_hfil
669 &\_hskip.5\_eqspace\_hfil\_the\_eqstyle{##}$\_hskip.5\_eqspace\_hfil
670 \_crrcr#1\_crrcr}}\_thinsk}
671
672 \public \eqalign ;

```

The `\displaylines{<formula>\cr<formula>\cr...<formula>}` creates horizontally centered formulae. It behaves exactly as in Plain  $\TeX$ . The `\halign` is applied directly in the outer display environment with lines of type `\hbox to\displaywidth`. This enables to break lines inside such display to more pages but it is impossible to use `\eqno` or `\leqno` or `\eqmark`.

$\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$  offers `\dislaylines to<dimen>{\_displaylines{<formula>\cr<formula>\cr...<formula>}}` as an alternative case of usage `\displaylines`. See section 4.3 in the [Typesetting Math with  \$\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}\$](#) . The centered formulas are in `\vcenter` in this case, so lines cannot be broken into more pages, but this case enables to use `\eqno` or `\leqno` or `\eqmark`.

math-macros.opm

```

692 \_def\_displaylines #1{\_ifx&#1&\_ea\_displaylinesD
693 \_else \_def\_tmp to##1\_end{\_def\_tmp{\_dimexpr ##1}}\_tmp #1\_end
694 \_ea\_displaylinesto \_fi}
695 \_long\_def\_displaylinesD #1{\_display \_tabskip=\_zoskip
696 \_halign{\_hbox to\_displaywidth{$\_elign\_hfil\_displaystyle##\_hfil$}\_crrcr
697 #1\_crrcr}}
698 \_long\_def\_displaylinesto #1{\_vcenter{\_openup\_jot \_math \_tabskip=\_zoskip
699 \_halign{\_strut\_hbox to\_span\_tmp{$\_hss\_displaystyle##\_hss$}\_crrcr
700 #1\_crrcr}}
701
702 \public \displaylines ;

```

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

```

709 \def\openup{\afterassignment\openupA\dimen0=}
710 \def\openupA{\advance\lineskip by\dimen0
711 \advance\baselineskip by\dimen0
712 \advance\lineskiplimit by\dimen0 }
713 \newifi\ifdtop
714 \def\display{\_global\_dtoptrue\openup\_jot\_math
715 \everycr{\noalign{\_ifdtop \_global\_dtopfalse \_ifdim\_prevdepth>-1000pt
716 \vskip-\lineskiplimit \vskip\normallineskiplimit \_fi
717 \_else \_penalty\_interdisplaylinepenalty \_fi}}}
718 \def\_elign{\_tabskip=\_zoskip\_everycr{}} % restore inside \_display
719 \long\_def\_eqalignno#1{\_display \_tabskip=\_centering
720 \_halign to\_displaywidth{\\_hfil$\_elign\_displaystyle{##}$\_tabskip=\_zoskip
721 &$\_elign\_displaystyle{ }\_##$}\_hfil\_tabskip\_centering
722 &\_hbox to\_zo{\_hss$\_elign##$}\_tabskip\_zoskip\_crrc
723 #1\_crrc}}
724 \long\_def\_leqalignno#1{\_display \_tabskip=\_centering
725 \_halign to\_displaywidth{\\_hfil$\_elign\_displaystyle{##}$\_tabskip=\_zoskip
726 &$\_elign\_displaystyle{ }\_##$}\_hfil\_tabskip=\_centering
727 &\_kern-\_displaywidth\_hbox to\_zo{$\_elign##$\_hss}\_tabskip\_displaywidth\_crrc
728 #1\_crrc}}
729 \_public \openup \eqalignno \leqalignno ;

```

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

```

736 \def\amsafam{4} \def\amsbfam{5}
737
738 \mathchardef \boxdot "2\_amsafam 00
739 \mathchardef \boxplus "2\_amsafam 01
740 \mathchardef \boxtimes "2\_amsafam 02
741 \mathchardef \square "0\_amsafam 03
742 \mathchardef \blacksquare "0\_amsafam 04
743 \mathchardef \centerdot "2\_amsafam 05
744 \mathchardef \lozenge "0\_amsafam 06
745 \mathchardef \blacklozenge "0\_amsafam 07
746 \mathchardef \circlearrowright "3\_amsafam 08
747 \mathchardef \circlearrowleft "3\_amsafam 09
748 \mathchardef \rightleftharpoons "3\_amsafam 0A
749 \mathchardef \leftrightharpoons "3\_amsafam 0B
750 \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, \notXXX becomes \_notXXX;
if \_nXXX is defined, \notXXX becomes \_nXXX;
otherwise, \notXXX is done in the usual way.

```

```

985 \mathchardef \_notchar "3236
986
987 \_protected\_def \_not#1{%
988 \_ifx #1<\_nless \_else
989 \_ifx #1>\_ngtr \_else
990 \_edef\_tmpn{\_csstring#1}%
991 \_ifcsname \_not\_tmpn\_endcsname \_csname \_not\_tmpn\_endcsname
992 \_else \_ifcsname \_n\_tmpn\_endcsname \_csname \_n\_tmpn\_endcsname
993 \_else \_mathrel{\_mathord{\\_notchar}\_mathord{#1}}}%
994 \_fi \_fi \_fi \_fi}
995 \_private
996 \nleq \ngeq \_nless \_ngtr \_nprec \_nsucc \_nleqslant \ngeqslant \_npreceq
997 \_nsucceq \_nleqq \_ngeqq \_nsim \_ncong \_nsubseteqq \_nsupseteqq \_nsubseteq
998 \_nsupseteq \_nparallel \_nmid \_nshortmid \_nshortparallel \_nvDash \_VDash
999 \_nvDash \_VDash \_ntrianglerighteq \_ntrianglelefteq \_ntriangleleft
1000 \_ntriangleright \_nleftarrow \_nrightarrow \_nLeftarrow \_nRightarrow
1001 \_nLeftrightarrow \_nleftrightharrow \_nexists ;
1002 \_public \not ;

```

`\mathstyles{<math list>}` behaves like `{<math list>}`, but you can use the 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`

```
1022 \_newcount\_stylenum
1023 \_def\mathstyles#1{\_mathchoice{\_stylenum0 #1}{\_stylenum1 #1}%
1024                               {\_stylenum2 #1}{\_stylenum3 #1}}
1025 \_def\dobystyle#1#2#3#4{\_ifcase\_stylenum#1\_or#2\_or#3\_or#4\_fi}
1026 \_def\currstyle{\dobystyle\displaystyle\textstyle\scriptstyle\scriptscriptstyle}
1027 \_public \mathstyles \dobystyle \currstyle \stylenum ;
```

The `\cramped` macro sets the cramped variant of the current style. Note that `\currstyle` initializes non-cramped variants. The example `\mathframe` above should be:

`\def\mathframe#1{\mathstyles{\frame{$\currstyle\cramped #1$}}}`.

Second note: `\cramped` macro reads the current math style from the `\mathstyle` LuaTeX primitive, so it does not work in numerators of generalized fractions but you can use it before the fraction is opened: `$\cramped {x^2\over y^2}$`.

`math-macros.opm`

```
1041 \_def\cramped{\_ifcase\_numexpr(\_mathstyle+1)/2\_relax\_or
1042   \_crampeddisplaystyle \_or \_crampedtextstyle \_or
1043   \_crampedscriptstyle \_or \_crampedscriptscriptstyle \_fi
1044 }
1045 \_public \cramped ;
```

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

The `\_textmff` and `\_scriptmff` are redefined in order to respect optical sizes. If we are in script style then the math mode starts in text style, but optical size is given to script style. The `\mathbox` in non-Unicode math respects optical sizes using different principle.

`math-macros.opm`

```
1058 \_def\mathbox#1{\_mathstyles{\_hbox{%
1059   \_ifnum\_stylenum<2 \_everymath{\_currstyle}%
1060   \_else
1061     \_ifnum\_stylenum=2 \_def\_textmff{+ssty=0;}\_fi
1062     \_ifnum\_stylenum=3 \_def\_textmff{+ssty=1;}\_def\_scriptmff{+ssty=1;}\_fi
1063     \_typoscale[\_dobystyle{}{}{700}{500}]/\_fi #1}}}%
1064 }
1065 \_public \mathbox ;
```

## 2.16 Unicode-math fonts

The `\loadmath {⟨Unicode-math font⟩}` 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 the new Unicode-math font isn't loaded until `\doloadmath` is used.

`\loadboldmath {⟨bold-font⟩} \to {⟨normal-font⟩}` loads bold variant only if `⟨normal-font⟩` was successfully loaded by the previous `\loadmath`. For example:

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

There are very few Unicode-math fonts with full `\boldmath` support. I know only XITSMath-Bold and KpMath-Bold. If `\loadboldmath` is not used then “faked bold” created from `\normalmath` is used by default.

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{[texgyrebonum-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
\loadmath{[KpMath-Regular]} ... KP fonts Math

```

## 2.16.1 Unicode-math macros preloaded in the format

```

3 \_codedecl \loadmath {Unicode Math fonts <2021-04-05>} % preloaded in format

```

math-unicode.opm

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

- define `\_unimathfont` as  $\langle Unicode\text{-}math\ font \rangle$ ,
- redefine `\normalmath` and `\boldmath` macros to their Unicode counterparts,
- load the `\_unimathfont` by `\normalmath`,
- print information about the 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     \_ifx\_ncharmA\_undefined \_opinput {unimath-codes.opm}\_fi
35     \_mathloadingfalse
36   \_fi\_fi}
37
38 \_public \loadmath \noloadmath \doloadmath ;

```

`\loadboldmath`  $\langle \boldsymbol{font} \rangle$   $\rightarrow$   $\langle normal\text{-}font \rangle$  defines `\_unimathboldfont` as  $\langle \boldsymbol{font} \rangle$  only if `\_unimathfont` is defined as  $\langle normal\text{-}font \rangle$ . 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 a similar sense as `\_setmathdimens`. 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 `\normalunimath` shows a combination of base Unicode-math font with 8bit Math font at family 4. See definition of `\script` macro where `\fam4` is used.

math-unicode.opm

```

73 \def\normalunimath{%
74   \loadumathfamily 1 {\unimathfont}{} % Base font
75   \loadmathfamily 4 rsfs % script
76   \setunimathdimens
77 }%
78 \def\boldunimath{%
79   \ifx\unimathboldfont \undefined
80     \loadumathfamily 1 {\unimathfont}{embolden=1.7;} % Base faked bold
81   \else
82     \loadumathfamily 1 {\unimathboldfont}{} % Base real bold font
83   \fi
84   \loadmathfamily 4 rsfs % script
85   \setunimathdimens
86 }%
87 \def\setunimathdimens{% PlainTeX sets these dimens for 10pt size only:
88   \delimitershortfall=0.5\fontdimen6\textfont3
89   \nulldelimiterspace=0.12\fontdimen6\textfont3
90   \scriptspace=0.05\fontdimen6\textfont3
91   \setbox0=\hbox{\everymath}\$ \fam1\displaystyle{0\atop0}$%
92   \Umathfractiondelsize\displaystyle = \dimexpr(\ht0-\Umathaxis\displaystyle)*2\relax
93   \setbox0=\box\voidbox
94 }

```

If you try the example above about `\loadboldmath{[xitsmath-bold]} \to {[xitsmath-regular]}` then you can find a bug in XITSMath-Bold font: the symbols for norm  $\|x\|$  are missing. So, we have to define `\boldmath` macro manually. The missing symbol is loaded from family 5 as no-bold variant in our example:

```

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

```

`\loadumathfamily` *number* {*font*} {*font features*} loads the given Unicode-math fonts in three sizes given by the `\setmathsizes` macro and sets it as the math family *number*. The *font features* 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 math family 1 is loaded then the family 2 and 3 are set by the same font because T<sub>E</sub>X needs to read dimension information about generating math formulae from these three math families. All information needed by T<sub>E</sub>X is collected in single Unicode-math font. The `\corrmsize` *factor* *space* can be used just before `\loadumathfamily`, see section 2.14 for more information.

The `\textmff`, `\scriptmff` and `\sscriptmff` are additional font features for text, script and sscript sizes respectively. They are locally re-defined in `\mathbox` macro.

math-unicode.opm

```

130 \def\_umathname#1#2{"#1:\mfontfeatures#2}
131 \def\_mfontfeatures{mode=base;script=math;}
132
133 \def\_loadumathfamily #1 #2#3 {%
134   \edef\_optsize\the\_optsize%
135   \_optsize=\_sizemtext \_font\_mF=\_umathname{#2}{\_textmff #3} at\_optsize
136   \_textfont#1=\_mF \_ifnum#1=1 \_textfont2=\_mF \_textfont3=\_mF \_fi
137   \_optsize=\_sizemscript \_font\_mF=\_umathname{#2}{\_scriptmff #3} at\_optsize
138   \_scriptfont#1=\_mF \_ifnum#1=1 \_scriptfont2=\_mF \_scriptfont3=\_mF \_fi
139   \_optsize=\_sizemsscript \_font\_mF=\_umathname{#2}{\_sscriptmff #3} at\_optsize
140   \_scriptscriptfont#1=\_mF \_ifnum#1=1 \_scriptscriptfont2=\_mF \_scriptscriptfont3=\_mF \_fi
141   \_optsize=\_optsize\save \_ptmunit=\_ptunit
142 }
143 \def\_textmff{} \def\_scriptmff{+ssty=0;} \def\_sscriptmff{+ssty=1;}

```

Unicode math font includes all typical math alphabets together, user needs not to load more T<sub>E</sub>X 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`  $\{\langle from \rangle - \langle to \rangle\} \langle class \rangle \langle family \rangle \backslash \langle first \rangle$  sets `\Umathcodes` of the characters in the interval  $\langle from \rangle - \langle to \rangle$  to  $\backslash \langle first \rangle$ ,  $\backslash \langle first \rangle + 1$ ,  $\backslash \langle first \rangle + 2$  etc., but `\_umathcharholes` are skipped (`\_umathcharholes` are parts of the Unicode table not designed for math alphabets but they cause 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.

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

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

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

math-unicode.opm

```

174 \_newcount\_umathnumA \_newcount\_umathnumB
175
176 \_def\_umathcorr#1#2{\_ea#1\_ea{\_the#2}}
177 \_def\_umathprepare#1{\_def\_umathscanholes##1[#1]##2##3\_relax{##2}}
178 \_def\_umathvalue#1f{\_ea\_umathscanholes\_umathcharholes[#1]{#1}\_relax}
179
180 \_def\_umathcharholes{% holes in math alphabets:
181   [119893]{ "210E}[119965]{ "212C}[119968]{ "2130}[119969]{ "2131}%
182   [119971]{ "210B}[119972]{ "2110}[119975]{ "2112}[119976]{ "2133}[119981]{ "211B}%
183   [119994]{ "212F}[119996]{ "210A}[120004]{ "2134}%
184   [120070]{ "212D}[120075]{ "210C}[120076]{ "2111}[120085]{ "211C}[120093]{ "2128}%
185   [120122]{ "2102}[120127]{ "210D}[120133]{ "2115}[120135]{ "2119}%
186   [120136]{ "211A}[120137]{ "211D}[120145]{ "2124}%
187 }
188 \_def\_umathrange#1#2#3#4{\_umathnumB=#4\_def\_tmp{#2 #3 }\_umathrangeA#1}
189 \_def\_umathrangeA#1-#2{\_umathnumA=#1\_relax
190   \_loop
191     \_umathcorr\_umathprepare\_umathnumB
192     \_Umathcode \_umathnumA = \_tmp \_umathcorr\_umathvalue{\_umathnumB}
193     \_ifnum\_umathnumA<`#2\_relax
194       \_advance\_umathnumA by1 \_advance\_umathnumB by1
195   \_repeat
196 }
197 \_def\_umathrangeGREEK{\_umathrange{~~~~0391-~~~~03a9}}
198 \_def\_umathrangegreek{\_umathrange{~~~~03b1-~~~~03d6}}
199 \_def\_greekdef#1{\_ifx#1\_relax \_else
200   \_begingroup \_lccode`X=\_umathnumB \_lowercase{\_endgroup \_def#1{X}}%
201   \_advance\_umathnumB by 1
202   \_ea\_greekdef \_fi
203 }
```

## 2.16.2 Macros and codes set when `\loadmatfont` is processed

The file `unimath-codes.opm` is loaded when the `\loadmath` is used. The macros here redefines globally all encoding dependent settings declared in the section 2.15.

unimath-codes.opm

```

3 \_codedecl \_ncharrmA {Uni math codes <2021-04-25>} % preloaded on demand by \loadmath
```

The control sequences for `\alpha`, `\beta` etc are redefined here. The `\alpha` expands to the character with Unicode "03B1, this is a 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 \phi
20   \varphi \chi \psi \omega \vardelta \epsilon \vartheta \varkappa \phi
21   \varrho \varpi \_relax
```

The math alphabets are declared here using the `\_umathrange{<range>}{<class>}{<family>}{<starting-code>}` macro.

unimath-codes.opm

```

28 \_chardef\_ncharrmA="A" \_chardef\_ncharrrma="a
29 \_chardef\_ncharbfa="1D400 \_chardef\_ncharbfa="1D41A
30 \_chardef\_ncharita="1D434 \_chardef\_ncharita="1D44E
31 \_chardef\_ncharbia="1D468 \_chardef\_ncharbia="1D482
32 \_chardef\_ncharcla="1D49C \_chardef\_ncharcla="1D4B6
33 \_chardef\_ncharbca="1D4D0 \_chardef\_ncharbca="1D4EA
34 \_chardef\_ncharfra="1D504 \_chardef\_ncharfra="1D51E
35 \_chardef\_ncharbra="1D56C \_chardef\_ncharbra="1D586
36 \_chardef\_ncharbba="1D538 \_chardef\_ncharbba="1D552
37 \_chardef\_ncharsna="1D5A0 \_chardef\_ncharsna="1D5BA
38 \_chardef\_ncharbsa="1D5D4 \_chardef\_ncharbsa="1D5EE
39 \_chardef\_ncharsia="1D608 \_chardef\_ncharsia="1D622
40 \_chardef\_ncharsxa="1D63C \_chardef\_ncharsxa="1D656
41 \_chardef\_nchartta="1D670 \_chardef\_nchartta="1D68A
42
43 \_protected\_def\_rmvariables {\_umathrange{A-Z}71\_ncharrmA \_umathrange{a-z}71\_ncharrrma}
44 \_protected\_def\_bfvariables {\_umathrange{A-Z}71\_ncharbfa \_umathrange{a-z}71\_ncharbfa}
45 \_protected\_def\_itvariables {\_umathrange{A-Z}71\_ncharita \_umathrange{a-z}71\_ncharita}
46 \_protected\_def\_bivvariables {\_umathrange{A-Z}71\_ncharbia \_umathrange{a-z}71\_ncharbia}
47 \_protected\_def\_calvariables {\_umathrange{A-Z}71\_ncharcla \_umathrange{a-z}71\_ncharcla}
48 \_protected\_def\_bcalvariables {\_umathrange{A-Z}71\_ncharbca \_umathrange{a-z}71\_ncharbca}
49 \_protected\_def\_frakvariables {\_umathrange{A-Z}71\_ncharfra \_umathrange{a-z}71\_ncharfra}
50 \_protected\_def\_bfrakvariables {\_umathrange{A-Z}71\_ncharbra \_umathrange{a-z}71\_ncharbra}
51 \_protected\_def\_bbvariables {\_umathrange{A-Z}71\_ncharbba \_umathrange{a-z}71\_ncharbba}
52 \_protected\_def\_sansvariables {\_umathrange{A-Z}71\_ncharsna \_umathrange{a-z}71\_ncharsna}
53 \_protected\_def\_bsansvariables {\_umathrange{A-Z}71\_ncharbsa \_umathrange{a-z}71\_ncharbsa}
54 \_protected\_def\_isansvariables {\_umathrange{A-Z}71\_ncharsia \_umathrange{a-z}71\_ncharsia}
55 \_protected\_def\_bisansvariables {\_umathrange{A-Z}71\_ncharsxa \_umathrange{a-z}71\_ncharsxa}
56 \_protected\_def\_ttvariables {\_umathrange{A-Z}71\_nchartta \_umathrange{a-z}71\_nchartta}
57
58 \_chardef\_greekrmA="0391 \_chardef\_greekrma="03B1
59 \_chardef\_greekbfa="1D6A8 \_chardef\_greekbfa="1D6C2
60 \_chardef\_greekita="1D6E2 \_chardef\_greekita="1D6FC
61 \_chardef\_greekbia="1D71C \_chardef\_greekbia="1D736
62 \_chardef\_greekсна="1D756 \_chardef\_greekсна="1D770
63 \_chardef\_greekсия="1D790 \_chardef\_greekсия="1D7AA
64
65 \_protected\_def\_itgreek {\_umathrangegreek71\_greekita}
66 \_protected\_def\_rmgreek {\_umathrangegreek71\_greekrma}
67 \_protected\_def\_bfgreek {\_umathrangegreek71\_greekbfa}
68 \_protected\_def\_bigreek {\_umathrangegreek71\_greekbia}
69 \_protected\_def\_bsansgreek {\_umathrangegreek71\_greekсна}
70 \_protected\_def\_bisansgreek {\_umathrangegreek71\_greekсия}
71 \_protected\_def\_itGreeK {\_umathrangeGREEK71\_greekita \_setnablait}
72 \_protected\_def\_rmGreeK {\_umathrangeGREEK71\_greekrma \_setnablarm}
73 \_protected\_def\_bfGreeK {\_umathrangeGREEK71\_greekbfa \_setnablafb}
74 \_protected\_def\_biGreeK {\_umathrangeGREEK71\_greekbia \_setnablabi}
75 \_protected\_def\_bsansGreeK {\_umathrangeGREEK71\_greekсна \_setnablabsans}
76 \_protected\_def\_bisansGreeK {\_umathrangeGREEK71\_greekсия \_setnablabisans}

```

`\_setnabla` is used in order to `\nabla` behaves like uppercase Greek letter, similar like `\Delta`. It depends on `\bf`, `\it` etc. selectors. If you want to deactivate this behavior, use `\def\_setnabla#1 {}`.

unimath-codes.opm

```

84 \_def \_setnabla {\_Umathcode"2207 = 7 1}
85 \_def \_setnablarm {\_setnabla"02207 }
86 \_def \_setnablafb {\_setnabla"1D6C1 }
87 \_def \_setnablait {\_setnabla"1D6FB }
88 \_def \_setnablabi {\_setnabla"1D735 }
89 \_def \_setnablabsans {\_setnabla"1D76F }
90 \_def \_setnablabisans {\_setnabla"1D7A9 }

```

Digits are configured like math alphabets.

unimath-codes.opm

```

96 \_chardef\_digitrm0="0
97 \_chardef\_digitbfo="1D7CE
98 \_chardef\_digitbb0="1D7D8

```

```

99 \_chardef\_digitsn0="1D7E2
100 \_chardef\_digitbs0="1D7EC
101 \_chardef\_digittt0="1D7F6
102
103 \_protected\_def\_rmdigits {\_umathrange{0-9}71\_digitrm0}
104 \_protected\_def\_bfdigits {\_umathrange{0-9}71\_digitbf0}
105 \_protected\_def\_bbdigits {\_umathrange{0-9}71\_digitbb0}
106 \_protected\_def\_sansdigits {\_umathrange{0-9}71\_digitsn0}
107 \_protected\_def\_bsansdigits {\_umathrange{0-9}71\_digitbs0}
108 \_protected\_def\_ttdigits {\_umathrange{0-9}71\_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 2.14) 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\_bfGreek\_bfdigits}}
\_protected\_def\_bi
  {\_tryloadbi\_tenbi \_inmath{\_bivvariables\_bigreek\_bfGreek\_bfdigits}}
\_public \bf \bi ;

```

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

```

127 \_protected\_def\_inmath#1{\_relax \_ifmode#1\_fi} % to keep off \loop processing in text mode
128
129 % You can redefine these macros to follow your wishes.
130 % For example, you need upright lowercase greek letters, you don't need
131 % \bf and \bi behave as sans serif in math, ...
132
133 \_protected\_def\_rm {\_tryloadrm \_tenrm \_inmath{\_rmvariables \_rmdigits}}
134 \_protected\_def\_it {\_tryloadit \_tenit \_inmath{\_itvariables \_itGreek}}
135 \_protected\_def\_bf
136   {\_tryloadbf \_tenbf \_inmath{\_bsansvariables \_bsansgreek \_bsansGreek \_bsansdigits}}
137 \_protected\_def\_bi
138   {\_tryloadbi \_tenbi \_inmath{\_bisansvariables \_bisansgreek \_bisansGreek \_bsansdigits}}
139 \_protected\_def\_tt {\_tryloadtt \_tentt \_inmath{\_ttvariables \_ttdigits}}
140 \_protected\_def\_bbchar {\_bbvariables \_bbdigits}
141 \_protected\_def\_cal {\_calvariables}
142 \_protected\_def\_frak {\_frakvariables}
143 \_protected\_def\_misans {\_isansvariables \_sansdigits}
144 \_protected\_def\_mbisans {\_bisansvariables \_bisansgreek \_bsansGreek \_bsansdigits}
145 \_protected\_def\_script {\_rmvariables \_fam4 }
146 \_protected\_def\_mit {\_itvariables \_rmdigits \_itgreek \_rmGreek }
147
148 \_public \rm \it \bf \bi \tt \bbchar \cal \frak \misans \mbisans \script \mit ;

```

Each Unicode slot carries information about math type. This is saved in the file `MathClass-15.txt` which is copied to `mathclass.opm`. The file has the following format:

```

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`.

```

158 \begingroup % \input mathclass.opm (which is a copy of MathClass.txt):
159 \long\def\p#1;#2 {\ifx`#2\else
160   \edef\tmp{\csname _c:#2\endcsname}\if\relax\tmp\else \pA#1...\end#2\fi
161   \ea\p \fi }
162 \def\pA#1..#2..#3\end#4{%
163   \ifx\relax#2\relax \pset{"#1}{#4}\else \fornum "#1..#2\do{\pset{##1}{#4}}\fi
164 }
165 \sdef{c:L}{1}\sdef{c:B}{2}\sdef{c:V}{2}\sdef{c:R}{3}\sdef{c:N}{0}\sdef{c:U}{0}
166 \sdef{c:F}{0}\sdef{c:O}{4}\sdef{c:C}{5}\sdef{c:P}{6}\sdef{c:A}{7}
167 \def\pset#1#2{\_Umathcode#1=\_tmp\_space 1 #1\relax
168   \if#20\_Udelcode#1=1 #1\relax\fi
169   \if#2C\_Udelcode#1=1 #1\relax\fi
170   \if#2F\_Udelcode#1=1 #1\relax\fi
171 }
172 \catcode`=14 \everyeof={;} } \def\par{}
173 \globaldefs=1 \ea \p \input mathclass.opm
174 \endgroup

```

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

```

70 \UnicodeMathSymbol{"00393}{\mupGamma} }{\mathalpha}{capital gamma, greek}%
71 \UnicodeMathSymbol{"00394}{\mupDelta} }{\mathalpha}{capital delta, greek}%
72 \UnicodeMathSymbol{"00395}{\mupEpsilon} }{\mathalpha}{capital epsilon, greek}%
73 \UnicodeMathSymbol{"00396}{\mupZeta} }{\mathalpha}{capital zeta, greek}%
74 \UnicodeMathSymbol{"00397}{\mupEta} }{\mathalpha}{capital eta, greek}%
75 \UnicodeMathSymbol{"00398}{\mupTheta} }{\mathalpha}{capital theta, greek}%
76 \UnicodeMathSymbol{"00399}{\mupIota} }{\mathalpha}{capital iota, greek}%
77 \UnicodeMathSymbol{"0039A}{\mupKappa} }{\mathalpha}{capital kappa, greek}%
78 \UnicodeMathSymbol{"0039B}{\mupLambda} }{\mathalpha}{capital lambda, greek}%
79 \UnicodeMathSymbol{"0039C}{\mupMu} }{\mathalpha}{capital mu, greek}%
80 \UnicodeMathSymbol{"0039D}{\mupNu} }{\mathalpha}{capital nu, greek}%
81 \UnicodeMathSymbol{"0039E}{\mupXi} }{\mathalpha}{capital xi, greek}%
82 \UnicodeMathSymbol{"0039F}{\mupOmicron} }{\mathalpha}{capital omicron, greek}%
83 \UnicodeMathSymbol{"003A0}{\mupPi} }{\mathalpha}{capital pi, greek}%
84 \UnicodeMathSymbol{"003A1}{\mupRho} }{\mathalpha}{capital rho, greek}%
85 \UnicodeMathSymbol{"003A3}{\mupSigma} }{\mathalpha}{capital sigma, greek}%

```

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

```

183 \begingroup % \input unimath-table.opm (it is a copy of unicode-math-table.tex):
184 \def\UnicodeMathSymbol #1#2#3#4{%
185   \ifnum#1=\_Umathcodenum#1 % the code isn't set by mathclass.opm
186     \_Umathchardef#2=0 1 #1 \_Umathcode#1=0 1 #1
187   \else \_Umathcharnumdef#2=\_Umathcodenum#1 \fi
188   \ifx#3\_mathopen \_def#2{\_Udelimiter 4 1 #1 }\fi
189   \ifx#3\_mathclose \_def#2{\_Udelimiter 5 1 #1 }\fi
190   \ifx#3\_mathaccent \_def#2{\_Umathaccent fixed 7 1 #1 }\fi
191 }
192 \globaldefs=1 \input unimath-table.opm
193 \endgroup

```

Many special characters must be declared with care...

```

199 \global\_Udelcode`<=1 "027E8 % these characters have different meaning
200 \global\_Udelcode`>=1 "027E9 % as normal and as delimiter
201
202 \_mit % default math alphabets setting
203
204 % hyphen character is transformed to minus:
205 \_Umathcode `~ = 2 1 "2212
206
207 % mathclass defines : as Punct, plain.tex as Rel, we keep mathclass,
208 % i.e. there is difference from plain.tex, you can use $f:A\to B$.
209
210 % mathclas defines ! as Ord, plain.tex as Close
211 \_Umathcode `! = 5 1 `! % keep plain.tex declaration
212 \_Umathchardef \mathexclam = 5 1 `!
213 % mathclas defines ? as Punct, plain.tex as Close
214 \_Umathcode `? = 5 1 `? % keep plain.tex declaration

```



```

215 \_Umathchardef \mathquestion = 5 1 `?
216
217 \_Umathcode `* = 2 1 "02217 % equivalent to \ast, like in plain TeX
218
219 \_protected\_def \_sqrt {\_Uradical 1 "0221A }
220 \_protected\_def \_cuberoot {\_Uradical 1 "0221B }
221 \_protected\_def \_fourthroot {\_Uradical 1 "0221C }
222
223 \_def \nabla {~~~~2207} % \nabla behaves as uppercase Gereek letter, see \_setnabla
224
225 \_public \sqrt \cuberoot \fourthroot ;
226
227 \_def\_intwithnolimits#1#2 {\_ifx#1\_relax \_else
228 \_ea\_let\_csname\_csstring#1op\_endcsname=#1%
229 \_ea\_def\_ea #1\_ea{\_csname\_csstring#1op\_endcsname \_nolimits}%
230 \_bgroup \_lccode`~=#2 \_lowercase{\_egroup \_mathcode`~="8000 \_let ~=#1}%
231 \_ea \_intwithnolimits \_fi
232 }
233 \_intwithnolimits \int "0222B \iint "0222C \iiint "0222D
234 \oint "0222E \oiint "0222F \oiint "02230
235 \intclockwise "02231 \varointclockwise "02232 \ointctrclockwise "02233
236 \sumint "02A0B \iiint "02A0C \intbar "02A0D \intBar "02A0E \fint "02A0F
237 \pointint "02A15 \sqint "02A16 \intlarhk "02A17 \intx "02A18
238 \intcap "02A19 \intcup "02A1A \upint "02A1B \lowint "02A1C \_relax "0
239
240 \_protected\_def \vert {\_Udelimiter 0 1 "07C }
241 \_protected\_def \Vert {\_Udelimiter 0 1 "02016 }
242 \_protected\_def \Vvert {\_Udelimiter 0 1 "02980 }
243
244 \_protected\_def \_overbrace #1{\mathop {\Umathaccent 7 1 "023DE{#1}}\limits}
245 \_protected\_def \_underbrace #1{\mathop {\Umathaccent bottom 7 1 "023DF{#1}}\limits}
246 \_protected\_def \_overparen #1{\mathop {\Umathaccent 7 1 "023DC{#1}}\limits}
247 \_protected\_def \_underparen #1{\mathop {\Umathaccent bottom 7 1 "023DD{#1}}\limits}
248 \_protected\_def \_overbracket #1{\mathop {\Umathaccent 7 1 "023B4{#1}}\limits}
249 \_protected\_def \_underbracket #1{\mathop {\Umathaccent bottom 7 1 "023B5{#1}}\limits}
250
251 \_public \overbrace \underbrace \overparen \underparen \overbracket \underbracket ;
252
253 \_protected\_def \widehat {\Umathaccent 7 1 "00302 }
254 \_protected\_def \widetilde {\Umathaccent 7 1 "00303 }
255 \_protected\_def \overleftharpoon {\Umathaccent 7 1 "020D0 }
256 \_protected\_def \overrightharpoon {\Umathaccent 7 1 "020D1 }
257 \_protected\_def \overleftarrow {\Umathaccent 7 1 "020D6 }
258 \_protected\_def \overrightarrow {\Umathaccent 7 1 "020D7 }
259 \_protected\_def \overleftrightharpoon {\Umathaccent 7 1 "020E1 }
260
261 \_mathchardef\ldotp="612E
262 \_let\|= \Vert
263 \_mathcode`\_="8000
264
265 \_global\_Umathcode "22EF = 0 1 "22EF % mathclass says that it is Rel
266 \_global\_Umathcode "002E = 0 1 "002E % mathclass says that dot is Punct
267 \_global\_Umathchardef \unicodeddots = 0 1 "22EF
268
269 \_global\_Umathcode `/_ = 0 1 `/_ % mathclass says that / is Bin, Plain TeX says that it is Ord.
270
271 % compressed dots in S and SS styles (usable in \matrix when it is in T, S and SS style)
272 \_protected\_def \vdots {\_relax \_ifnum \_mathstyle>3 \_unicodevdots \_else \_vdots \_fi}
273 \_protected\_def \ddots {\_relax \_ifnum \_mathstyle>3 \_unicodeddots \_else \_ddots \_fi}
274 \_protected\_def \adots {\_relax \_ifnum \_mathstyle>3 \_unicodeadots \_else \_adots \_fi}
275
276 % Unicode superscripts (²) and subscripts as simple macros with \mathcode"8000
277 \_bgroup
278 \_def\_tmp#1#2{\_global\_mathcode#1="8000 \_lccode`\_~=#1 \_lowercase{\_gdef~}{#2}}
279 \_fornum 0..1 \_do {\_tmp{"207#1}{\_#1}}
280 \_tmp{"B2}{\_#2}\_tmp{"B3}{\_#3}
281 \_fornum 4..9 \_do {\_tmp{"207#1}{\_#1}}
282 \_fornum 0..9 \_do {\_tmp{"208#1}{\_#1}}
283 \_egroup

```

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

`unimath-codes.opm`

```

290 \_let \setminus=\smallsetminus
291 \_let \diamond=\smwhtdiamond
292 \_let \colon=\mathcolon
293 \_let \bullet=\smbllkcicle
294 \_let \circ=\vysmwhtcircle
295 \_let \bigcirc=\mdlgwhtcircle
296 \_let \to=\rightarrow
297 \_let \le=\leq
298 \_let \ge=\geq
299 \_let \neq=\neq
300 \_protected\_def \triangle {\mathord{\bigtriangleup}}
301 \_let \emptyset=\varnothing
302 \_let \hbar=\hslash
303 \_let \land=\wedge
304 \_let \lor=\vee
305 \_let \owns=\ni
306 \_let \gets=\leftarrow
307 \_let \mathring=\ocirc
308 \_let \lnot=\neg
309 \_let \longdivisionsign=\longdivision
310 \_let \backepsilon=\upbackepsilon
311 \_let \eth=\matheth
312 \_let \dbkarow=\dbkarrow
313 \_let \drbkarow=\drbkarow
314 \_let \hksearrow=\hksearrow
315 \_let \hkswarrow=\hkswarrow
316
317 \_let \upalpha=\mupalpha
318 \_let \upbeta=\mupbeta
319 \_let \upgamma=\mupgamma
320 \_let \updelta=\mupdelta
321 \_let \upepsilon=\mupvarepsilon
322 \_let \upvarepsilon=\mupvarepsilon
323 \_let \upzeta=\mupzeta
324 \_let \upeta=\mupeta
325 \_let \uptheta=\muptheta
326 \_let \upiota=\mupiota
327 \_let \upkappa=\mupkappa
328 \_let \uplambda=\muplambda
329 \_let \upmu=\mupmu
330 \_let \upnu=\mupnu
331 \_let \upxi=\mupxi
332 \_let \upomicron=\mupomicron
333 \_let \uppi=\muppi
334 \_let \uprho=\muprho
335 \_let \upvarrho=\mupvarrho
336 \_let \upvarsigma=\mupvarsigma
337 \_let \upsigma=\mupsigma
338 \_let \uptau=\muptau
339 \_let \upupsilon=\mupupsilon
340 \_let \upvarphi=\mupvarphi
341 \_let \upchi=\mupchi
342 \_let \uppsi=\muppsi
343 \_let \upomega=\mupomega
344 \_let \upvartheta=\mupvartheta
345 \_let \upphi=\mupphi
346 \_let \upvarpi=\mupvarpi

```

The `\not` macro is redefined here. If the `\not!⟨char⟩` is defined (by `\negationof`) then this macro is used. Else centered / is printed over the `⟨char⟩`.

`unimath-codes.opm`

```

354 \_protected\_def \not#1{%
355   \_trycs{\not!\_csstring#1}{\_mathrel\_mathstyle{%
356     \_setbox0=\_hbox{\_math$\_currstyle#1$}%
357     \_hbox to\_wd0{\_hss$\_currstyle/\_hss}\_kern-\_wd0 \_box0
358   }}

```

```

359 \_def\_negationof #1#2{\_ea\_let \_csname _not!\_csstring#1\_endcsname =#2}
360
361 \_negationof = \neq
362 \_negationof < \nless
363 \_negationof > \ngtr
364 \_negationof \gets \nleftarrow
365 \_negationof \simeq \nsime
366 \_negationof \equal \ne
367 \_negationof \le \nleq
368 \_negationof \ge \ngeq
369 \_negationof \greater \ngtr
370 \_negationof \forksnot \forks
371 \_negationof \in \notin
372 \_negationof \mid \nmid
373 \_negationof \cong \ncong
374 \_negationof \leftarrow \nleftarrow
375 \_negationof \rightarrow \nrightarrow
376 \_negationof \leftrightarrow \nleftrightarrow
377 \_negationof \Leftrightarrow \nLeftrightarrow
378 \_negationof \Leftrightarrow \nLeftrightarrow
379 \_negationof \Rightarrow \nRightarrow
380 \_negationof \exists \nexists
381 \_negationof \ni \nni
382 \_negationof \parallel \nparallel
383 \_negationof \sim \nsim
384 \_negationof \approx \napprox
385 \_negationof \equiv \nequiv
386 \_negationof \asymp \nasymp
387 \_negationof \lesssim \nlesssim
388 \_negationof \ngtrsim \ngtrsim
389 \_negationof \lessgtr \nlessgtr
390 \_negationof \gtrless \ngtrless
391 \_negationof \prec \nprec
392 \_negationof \succ \nsucc
393 \_negationof \subset \nsubset
394 \_negationof \supset \nsupset
395 \_negationof \subseteq \nsubseteq
396 \_negationof \supseteq \nsupseteq
397 \_negationof \vdash \nvdash
398 \_negationof \vdash \nvDash
399 \_negationof \Vdash \nVdash
400 \_negationof \VDash \nVDash
401 \_negationof \preccurlyeq \npreccurlyeq
402 \_negationof \succcurlyeq \nsucccurlyeq
403 \_negationof \sqsubseteq \nsqsubseteq
404 \_negationof \sqsupseteq \nsqsupseteq
405 \_negationof \vartriangleleft \nvartriangleleft
406 \_negationof \vartriangleright \nvartriangleright
407 \_negationof \trianglelefteq \ntrianglelefteq
408 \_negationof \trianglerighteq \ntrianglerighteq
409 \_negationof \vinfty \nvinfty
410
411 \_public \not ;

```

Newly declared public control sequences are used in internal macros by OpTeX. We need to get new meanings for these control sequences in the private namespace.

unimath-codes.opm

```

419 \_private
420 \ldotp \cdotp \bullet \triangleleft \triangleright \mapstochar \rightarrow
421 \prime \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
422 \rbrace \lbrace \relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
423 \longrightarrow \Longleftarrow \unicodevdots \unicodeddots \unicodeadots ;

```

### 2.16.3 More Unicode-math examples

Example of using additional math font is in section 5.3 in the [optex-math.pdf](#) documentation

You can combine more Unicode math fonts in single formula simply by the `\addUmathfont` macro, see [OpTeX trick 0030](#).

See <http://tex.stackexchange.com/questions/308749> for technical details about Unicode-math.

## 2.16.4 Printing all Unicode math slots in used math font

This file can be used for testing your Unicode-math font and/or for printing T<sub>E</sub>X sequences which can be used in math.

Load Unicode math font first (for example by `\fontfam[termes]` or by `\loadmath{\langle math-font \rangle}`) and then you can do `\input print-unimath.opm`. The big table with all math symbols is printed.

`print-unimath.opm`

```

3 \_codedecl \_undefined {Printing Unicode-math table \string<2020-06-08>}
4
5 \_begingroup
6   \_def\UnicodeMathSymbol#1#2#3#4{%
7     \_ifnum#1>'10000 \_endinput \_else \_printmathsymbol{#1}{#2}{#3}{#4}\_fi
8   }
9   \_def\UnicodeMathSymbolA#1#2#3#4{%
10    \_ifnum#1>'10000 \_printmathsymbol{#1}{#2}{#3}{#4}\_fi
11  }
12  \_def\_printmathsymbol#1#2#3#4{%
13    \_hbox{\_hbox to2em{#2}{\_hss}\_hbox to3em
14      {\small\_printop#3\_hss}{\_tt\_string#2\_trycs{\_eq:\_string#2}{}}}
15  }
16  \_def\_eq#1#2{\_sdef{\_eq:\_string#2}{\_string#1}}
17  \_eq \diamond\smwhtdiamond \_eq \bullet\smblkcrcircle \_eq \circ\vysmwhtcircle
18  \_eq \bigcirc\mdlgwhtcircle \_eq \rightarrow \_eq \leq\leq
19  \_eq \geq\geq \_eq \neq\ne \_eq \emptyset\varnothing \_eq \hbar\hslash
20  \_eq \land\wedge \_eq \lor\vee \_eq \owns\ni \_eq \gets\leftarrow
21  \_eq \mathring\circ \_eq \lnot\neg \_eq \backepsilon\upbackepsilon
22  \_eq \eth\matheth \_eq \dbkarow\dbkarow \_eq \drbkarow\drbkarow
23  \_eq \hksearrow\hksearrow \_eq \hkswarrow\hkswarrow
24
25  \_tracinglostchars=0
26  \_fontdef\small{\_setfontsize{at5pt}\_rm}
27  \_def\_printop{\_def\mathop{0p}}
28  \_def\mathalpha{Alph}\_def\mathord{0rd}\_def\mathbin{Bin}\_def\mathrel{Rel}
29  \_def\mathopen{Open}\_def\mathclose{Close}\_def\mathpunct{Punct}\_def\mathfence{Fence}
30  \_def\mathaccent{Acc}\_def\mathaccentwide{Accw}\_def\mathbotaccentwide{AccBw}
31  \_def\mathbotaccent{AccB}\_def\mathaccentoverlay{AccO}
32  \_def\mathover{Over}\_def\mathunder{Under}
33  \_typosize[7.5/9]\_normalmath \_everymath={ }
34
35  Codes U+00000 \_dots\ U+10000
36  \_begmulti 3
37    \_input unimath-table.opm
38  \_endmulti
39
40  \_medskip\_goodbreak
41  Codes U+10001 \_dots\ U+1EEF1 \_let\UnicodeMathSymbol=\UnicodeMathSymbolA
42  \_begmulti 4
43    \_input unimath-table.opm
44  \_endmulti
45 \_endgroup

```

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

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

`fonts-opmac.opm`

```

3 \_codedecl \_typosize {Font managing macros from OPmac <2021-03-10>} % loaded in format

```

`\typosize` [*font-size*]/[*baselineskip*] 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 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   \_ifx$#1$\_def\_tmp{[/]\_else
40     \_settmpdim{#1}\_optsize
41     \_edef\_tmp{[\_ea\_ignorept\_the\_tmpdim/]\_fi
42   \_ifx$#2$\_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** { $\langle baselineskip \rangle$ } sets new **\\_baselineskip** and more values of registers which are dependent on the  $\langle baselineskip \rangle$  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 **\\_mainfontsize** and **\\_mainbaselineskip** registers and loads fonts at given sizes. It redefines itself as **\\_setmainvaluesL** to set the main values only first. The **\\_setmainvaluesL** does only fonts loading.

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

**\\_mfontsrule** gives the rule how math fonts are loaded when **\\_typosize** or **\\_typoscale** are used. The value of **\\_mfontsrule** can be:

- 0: no math fonts are loaded. User must use **\\_normalmath** or **\\_boldmath** explicitly.
- 1: **\\_normalmath** is run if **\\_typosize**/**\\_typoscale** are used first or they are run at outer group level. No **\\_everymath**/**\\_everydisplay** are set in this case. If **\\_typosize**/**\\_typoscale** are run repeatedly in a group then **\\_normalmath** is run only when math formula occurs. This is done using **\\_everymath**/**\\_everydisplay** and **\\_setmathfonts**. **\\_mfontsrule**=1 is default.
- 2: **\\_normalmath** is run whenever **\\_typosize**/**\\_typoscale** are used. **\\_everymath**/**\\_everydisplay** registers are untouched.

fonts-opmac.opm

```

99 \_newskip \_mainbaselineskip \_mainbaselineskip=0pt \_relax
100 \_newdimen \_mainfontsize \_mainfontsize=0pt
101 \_newcount \_mfontsrule \_mfontsrule=1
102
103 \_def\_setmainvalues {%
104   \_mainbaselineskip=\_baselineskip

```

```

105 \_mainfo size=\_optsize
106 \_topskip=\_mainfo size \_splittopskip=\_topskip
107 \_ifmmode \_else \_bf \_it \_bi \_rm \_fi % load all basic variants of the family
108 \_ifnum \_mfontsrule>0 \_normalmath \_fi % load math fonts first
109 \_let \_setmainvalues=\_setmainvaluesL
110 }
111 \_def \_setmainvaluesL {\_relax \_ifmmode \_else \_rm \_fi % load text font
112 \_ifcase \_mfontsrule % load math fonts
113 \_or \_ifnum\_currentgroup level=0 \_normalmath
114 \_else \_everymath={\_setmathfonts}\_everydisplay={\_normalmath}%
115 \_let \_runboldmath=\_relax \_fi
116 \_or \_normalmath \_fi}
117 \_def \_scalemain {%
118 \_ifdim \_mainfo size=\_zo
119 \_mainfo size=10pt \_mainbaselineskip=12pt
120 \_let \_setmainvalues=\_setmainvaluesL
121 \_fi
122 \_optsize=\_mainfo size \_baselineskip=\_mainbaselineskip
123 }
124 \_public \_scalemain \_mainfo size \_mainbaselineskip \_mfontsrule ;

```

Suppose following example: `{\typosize[13/15] Let  $M$  be a subset of  $R$  and  $x$  in  $M$ ...}` If `\mfontsrule=1` then `\typosize` does not load math fonts immediately but at the first math formula. It is done by `\everymath` register, but the contents of this register is processed inside the math group. If we do `\everymath={\_normalmath}` then this complicated macro will be processed three times in your example above. We want only one processing, so we do `\everymath={\_setmathfonts}` and this macro closes math mode first, loads fonts and opens math mode again.

fonts-opmac.opm

```

138 \_def \_setmathfonts{\_normalmath\_everymath{}\_everydisplay{}}

```

`\thefontsize` [*size*] and `\thefontscale` [*factor*] do modification of the size of the current font. They are implemented by the `\newcurrfontsize` macro.

fonts-opmac.opm

```

146 \_protected\_def \_thefontsize[#1]{\_if$#1$\_else
147 \_tmpdim=#1\_ptunit
148 \_newcurrfontsize{at\_tmpdim}%
149 \_fi
150 \_ignorespaces
151 }
152 \_protected\_def \_thefontscale[#1]{\_ifx$#1$\_else
153 \_tmpdim=#1pt \_divide\_tmpdim by1000
154 \_tmpdim=\_ea\_ea\_ea\_ignorept \_pdfontsize\_font \_tmpdim
155 \_newcurrfontsize{at\_tmpdim}%
156 \_fi
157 \_ignorespaces
158 }
159 \_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

```

168 \_protected\_def \_em {%
169 \_ea\_ifx \_the\_font \_tenit \_additcorr \_rm \_else
170 \_ea\_ifx \_the\_font \_tenbf \_bi\_aftergroup\_afteritcorr\_else
171 \_ea\_ifx \_the\_font \_tenbi \_additcorr \_bf \_else
172 \_it \_aftergroup\_afteritcorr\_fi\_fi\_fi
173 }
174 \_def \_additcorr{\_ifdim\_lastskip>\_zo
175 \_skip0=\_lastskip \_unskip\_italcorr \_hskip\_skip0 \_else\_italcorr \_fi}
176 \_def \_afteritcorr{\_futurelet\_next\_afteritcorrA}
177 \_def \_afteritcorrA{\_ifx\_next.\_else\_ifx\_next,\_else \_italcorr \_fi\_fi}
178 \_let \_italcorr=\/

```

The `\boldify` macro does `\let\rm\bf`, `\let\it\bi` and `\let\normalmath=\boldmath`. All following text will be in bold. It should be used after `\typosize` or `\typoscale` macros.

The internal `\_runboldmath` macro runs `\_boldmath` immediately if no delay of the math font loading



is set by `\_setmainvaluesL`.

The `\rm`, `\it` in math mode must keep its original meaning.

fonts-opmac.opm

```
189 \_protected\def \_boldify {%
190   \_let \_setmainvalues=\_setmainvaluesL
191   \_let\it =\_bi \_let\rm =\_bf \_let\_normalmath=\_boldmath \_bf
192   \_runboldmath
193   \_ifx\_ncharmA\_undefined \_protected\_addto\rm{\_fam0 }\_protected\_addto\it{\_fam1 }%
194   \_else \_protected\def\rm {\_tryloadbf \_tenbf \_inmath{\_rmvariables \_rmdigits}}%
195   \_protected\def\it {\_tryloadbi \_tenbi \_inmath{\_itvariables}}%
196   \_fi
197 }
198 \_def\_runboldmath{\_boldmath}
199
200 \_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 the default font size (declared by first `\typsize` command and redefines itself be only the font switch for the next pages.

fonts-opmac.opm

```
210 \_def \_rmfixed {% used in default \footline
211   {\_ifdim\_mainfosize=0pt \_mainfosize=10pt \_fi
212   \_fontdef\_tenrm{\_setfontsize{at\_mainfosize}\_resetmod\_rm}%
213   \_global\_let\_rmfixed=\_tenrm}% next use will be font switch only
214   \_rmfixed
215 }
216 \_let \rmfixed = \_tenrm % user can redefine it
```

## 2.18 Output routine

The output routine `\_optexoutput` is similar as in plain T<sub>E</sub>X. It does:

- `\_begoutput` which does:
  - increments `\pageno`,
  - prints `\_Xpage{\_pageno}{\_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: \_pageno` 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 0pt).
  - footline box by `\_makefootline`, if the `\footline` is nonempty
- `\_endoutput` which does:
  - increments `\pageno` using `\advancepageno`
  - runs output routine repeatedly if `\dosupereject` is activated.

output.opm

```
3 \_codedecl \nopagenumbers {Output routine <2021-02-25>} % preloaded in format
```

`\_optexoutput` is the default output routine. You can create another...

output.opm

```
9 \_output={\_optexoutput}
10 \_def \_optexoutput{\_begoutput \_shipout\_completepage \_endoutput}
```

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 are local in the `\output` group.

The `\_prepoffsets` can set `\hoffset` differently for the left or right page. It is re-defined by the `\margins` macro..

The `\_regmark` tokens list includes accumulated #2 from the `\regmacro`. Logos and other 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 \_setxhsize \_preoffsets \_the\_regmark}
29 \_def \_endoutput{\_advancepageno
30 {\_globaldefs=1 \_the\_nextpages \_nextpages={}}%
31 \_ifnum\_outputpenalty>-20000 \_else\_dosupereject\_fi
32 }
33 \_def \_preoffsets {}
```

output.opm

The `\hsize` value can be changed at various places in the document but we need to have a constant value `\_xhsize` in the output routine (for headlines and footlines, for instance). This value is set from the current value of `\hsize` when `\_setxhsize` macro is called. This macro destroys itself, so the value is set only once. Typically it is done in `\margins` macro or when first `\_optexoutput` routine is called (see `\_begoutput`). Or it is called at the beginning of the `\begtt...\endtt` environment before `\hsize` value is eventually changed by the user in this environment.

```
47 \_newdimen \_xhsize
48 \_def\_setxhsize {\_global\_xhsize=\_hsize \_global\_let\_setxhsize=\_relax}
```

output.opm

`\gpageno` counts pages from one in the whole document

```
54 \_newcount\_gpageno
55 \_public \gpageno ;
```

output.opm

The `\_completepage` is similar to 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`) extended 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, the default value of the `\_ensureblack` macro is empty. But the first usage of color macros in the document re-defines `\_ensureblack`. See the section 2.20 for more details.

```
70 \_def\_completepage{\_vbox{%
71 \_istoksemtty \_pgbackground
72 \_iffalse \_ensureblack{\_backgroundbox{\_the\_pgbackground}}\_nointerlineskip \_fi
73 \_ensureblack{\_makeheadline}%
74 \_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
75 \_ensureblack{\_makefootline}}%
76 }
77 \_def \_ensureblack #1{#1} % will be re-defined by color macros
78 \_let \_opennotestack = \_relax % will be re-defined by color macros
79 \_def \_backgroundbox #1{\_moveleft\_hoffset\_vbox to\_zo{\_kern-\_voffset #1\_vss}}
```

output.opm

`\_makeheadline` creates `\vbox to0pt` with its contents (the `\headline`) shifted by `\headlinedist` up.

```
86 \_def\_makeheadline {\_istoksemtty \_headline \_iffalse
87 \_vbox to\_zo{\_vss
88 \_baselineskip=\_headlinedist \_lineskiplimit=-\_maxdimen
89 \_hbox to\_xhsize{\_the\_headline}\_hbox{}}\_nointerlineskip
90 \_fi
91 }
```

output.opm

The `\_makefootline` appends the `\footline` to the page-body box.

```
97 \_def\_makefootline{\_istoksemtty \_footline \_iffalse
98 \_baselineskip=\_footlinedist
99 \_lineskiplimit=-\_maxdimen \_hbox to\_xhsize{\_the\_footline}
100 \_fi
101 }
```

output.opm

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.

```

110 \def\pagecontents{\pagedest % destination of the page
111 \ifvoid\topins \else \ensureblack{\unvbox\topins}\fi
112 \dimen0=\dp255 \unvbox255 % open up \box255
113 \ifvoid\footins \else % footnote info is present
114 \vskip\skip\footins
115 \ensureblack{\footnoterule \openfnstack \unvbox\footins}\fi
116 \kern-\dimen0 \vskip \pgbottomskip
117 }
118 \def \pagedest {\def\destheight{25pt}\dest[pg:\the\pageno]}
119 \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.

```

130 \countdef\pageno=0 \pageno=1 % first page is number 1
131 \def \folio {\ifnum\pageno<0 \romannumeral-\pageno \else \number\pageno \fi}
132 \def \nopagenumbers {\footline={}}
133 \def \advancepageno {%
134 \ifnum\pageno<0 \decr\pageno \else \incr\pageno \fi
135 } % increase |pageno|
136 \def \raggedbottom {\topskip=\dimexpr\topskip plus60pt \pgbottomskip=0pt plus1fil\relax}
137 \def \normalbottom {\topskip=\dimexpr\topskip \pgbottomskip=0pt\relax}
138
139 \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 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 a 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.

```

152 \newinsert\footins
153 \def \footnote #1{\let\osf=\empty % parameter #2 (the text) is read later
154 \ifhmode \edef\osf{\spacefactor\the\spacefactor}\fi
155 #1\osf\vfootnote{#1}}
156 \def\vfootnote{\opfootnote{}}
157 \def \opfootnote #1#2{\insert\footins\bgroup
158 \interlinepenalty=\interfootnotelinepenalty
159 \leftskip=\zo \rightskip=\zo \spaceskip=\zo \xspaceskip=\zo \relax
160 \let\colorstackcnt=\fnstack % special color stack for footnotes
161 #1\relax % local settings used by \fnote macro
162 \splittopskip=\ht\strutbox % top baseline for broken footnotes
163 \splitmaxdepth=\dp\strutbox \floatingpenalty=20000
164 \textindent{#2}\footstrut
165 \isnextchar \bgroup
166 {\bgroup \aftergroup\vfootA \afterassignment\ignorespaces \let\next={}\vfootB}%
167 }
168 \def\vfootA{\unskip\strut\isnextchar\colorstackpop\closefncolor\vfootF}
169 \def\vfootB #1{#1\unskip\strut\vfootF}
170 \def\vfootF{\egroup} % close \insert\footins\bgroup
171 \def\closefncolor#1{#1\isnextchar\colorstackpop\closefncolor\vfootF}
172 \def \footstrut {\vbox to \splittopskip{}}
173 \skip\footins=\bigskipamount % space added when footnote is present
174 \count\footins=1000 % footnote magnification factor (1 to 1)
175 \dimen\footins=8in % maximum footnotes per page
176 \public
177 \footins \footnote \vfootnote \footstrut ;

```

The `\topins` macros `\topinsert`, `\midinsert`, `\pageinsert`, `\endinsert` are the same as in plain T<sub>E</sub>X.

```

185 \newinsert\topins
186 \newif\ifupage \newif\ifumid
187 \def \topinsert {\umidfalse \upagefalse \oins}
188 \def \midinsert {\umidtrue \oins}
189 \def \pageinsert {\umidfalse \upagetrue \oins}
190 \skip\topins=\zoskip % no space added when a topinsert is present

```

```

191 \count\topins=1000 % magnification factor (1 to 1)
192 \dimen\topins=\maxdimen % no limit per page
193 \def \oins {\par \begingroup\setbox0=\vbox\bgroup} % start a \vbox
194 \def \endinsert {\par\egroup % finish the \vbox
195 \ifumid \dimen0=\ht0 \advance\dimen0 by\dp0 \advance\dimen0 by\baselineskip
196 \advance\dimen0 by\pagetotal \advance\dimen0 by-\pageshrink
197 \ifdim\dimen0>\pagegoal \umidfalse \upagefalse \fi \fi
198 \ifumid \bigskip \box0 \bigbreak
199 \else \insert \topins {\penalty100 % floating insertion
200 \splittopskip=0pt
201 \splitmaxdepth=\maxdimen \floatingpenalty=0
202 \ifupage \dimen0=\dp0
203 \vbox to\vsizel{\unvbox0 \kern-\dimen0}% depth is zero
204 \else \box0 \nobreak \bigskip \fi}\fi\endgroup}
205
206 \public \topins \topinsert \midinsert \pageinsert \endinsert ;

```

The `\draft` macro is an example of usage `\pgbackground` to create watercolor marks.

output.opm

```

213 \def \draft {\pgbackground={\draftbox{\draftfont DRAFT}}}%
214 \fontdef\draftfont{\setfontsize{at10pt}\bf}%
215 \global\let\draftfont=\draftfont
216 }
217 \def \draftbox #1{\setbox0=\hbox{#1}%
218 \kern.5\vsizel\kern\voffset\kern4.5\wd0
219 \hbox toOpt{\kern.5\hsize\kern\hoffset\kern-2\wd0
220 \pdfsave \pdfrotate{55}\pdfscale{10}{10}%
221 \hbox toOpt{\localcolor\setgreycolor{.8}\box0\hss}%
222 \pdfrestore
223 \hss}%
224 }
225 \public \draft ;

```

## 2.19 Margins

The `\margins` macro is documented in the section 1.2.1.

margins.opm

```

3 \codedec1 \margins {Macros for margins setting <2021-03-15>} % preloaded in format

```

`\margins/⟨pg⟩ ⟨fmt⟩ (⟨left⟩,⟨right⟩,⟨top⟩,⟨bot⟩)⟨unit⟩` takes its parameters, does calculation and sets `\hoffset`, `\voffset`, `\hsize` and `\vsizel` registers. Note that OpTeX sets the page origin at the top left corner of the paper, no at the obscure position 1 in, 1 in. It is much more comfortable for macro writers.

margins.opm

```

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 \hsize=\hsize \setxhsize % \xhsize used by \output routine
29 \fi\fi
30 \if$#5$\if$#6$\voffset=\dimexpr (\pgheight-\vsizel)/2 \relax
31 \else \voffset=\dimexpr \pgheight-\vsizel-#6\tmp \relax % only bottom margin
32 \fi
33 \else \if$#6$\voffset=#5\tmp \relax % only top margin
34 \else \vsizel=\dimexpr \pgheight-#5\tmp-#6\tmp \relax % top+bottom margin
35 \voffset=#5\tmp \relax
36 \fi\fi
37 \if 1#1\shiftoffset=0pt \def\preppoffsets{}\else \if 2#1% double-page layout

```

```

38     \shiftoffset = \dimexpr \pgwidth -\hsize -2\hoffset \relax
39     \def\preppoffsets{\ifodd\pageno \else \advance\hoffset \shiftoffset \fi}%
40     \else \opwarning{use \string\margins/1 or \string\margins/2}%
41     \fi\fi\fi
42 }
43 \def\setpagedimens{\_isnextchar{\_setpagedimensB}{\_setpagedimensA}}
44 \def\setpagedimensA#1 {\_ifcsname _pgs:#1\_endcsname
45     \_ea\_ea\_ea\_setpagedimensB \_csname _pgs:#1\_endcsname\_space
46     \else \opwarning{page specification "#1" is undefined}\_fi}
47 \def\setpagedimensB (#1,#2)#3 {\_setpagedimensC\_pgwidth=#1:#3
48     \_setpagedimensC\_pgheight=#2:#3
49     \pdfpagewidth=\pgwidth \pdfpageheight=\pgheight
50 }
51 \def\setpagedimensC #1=#2:#3 {#1=#2\_ifx^#3\_tmp\_else#3\_fi\_relax\_truedimen#1}
52
53 \_public \margins ;

```

The common page dimensions are defined here.

```

59 \sdef{pgs:a3}{(297,420)mm} \sdef{pgs:a4}{(210,297)mm} \sdef{pgs:a5}{(148,210)mm}
60 \sdef{pgs:a3l}{(420,297)mm} \sdef{pgs:a4l}{(297,210)mm} \sdef{pgs:a5l}{(210,148)mm}
61 \sdef{pgs:b5}{(176,250)mm} \sdef{pgs:letter}{(8.5,11)in}

```

margins.opm

`\magscale` [*factor*] does `\mag=`*factor* and recalculates page dimensions to their true values.

margins.opm

```

68 \def\trueunit{}
69 \def\magscale[#1]{\_mag=#1\_def\trueunit{true}%
70     \_ifdim\_pgwidth=0pt \else \truedimen\_pgwidth \truedimen\_pgheight \fi
71     \truedimen\_pdfpagewidth \truedimen\_pdfpageheight
72 }
73 \def\truedimen#1{\_ifx\_trueunit\_empty \else#1=\_ea\_ignorept\_the#1truept \fi}
74
75 \_public \magscale ;

```

## 2.20 Colors

The colors have different behavior than fonts. Marks (whatsits) with color information are stored into PDF output and  $\TeX$  doesn't interpret them. The PDF viewer (or PDF interpreter in a printer) reads these marks and switches colors according to them. This is independent of  $\TeX$  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 return to the black color using `\Black` manually.

By default, Op $\TeX$  sets `\localcolor`. It means that the typesetting returns to a previous color at the end of the current group, so you cannot write `\Black` explicitly. This is implemented using the `\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 the previous color is 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 the 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 of 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 the black component from the color. You can use the `-Black` trick after `\useK` command to remove grey components occurred during color mixing.

Finally, you can use `^` immediately preceded before the macro name of the color. Then the complementary color is used here.

```
\colordef\mycolor{\Grey+.6^Blue} % the same as \colordef\mycolor{\Grey+.6Yellow}
```

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 \_newifi \_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 componentns \rangle}` parameter expand to `\_setcolor{\langle pdf-primitive \rangle}` 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 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}`.



```

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 (*pdf-primitive*). 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 similar work, it re-defines `\formatrgb` macro. The Grey color space is unchanged and works in both main settings (RGB or CMYK) without collisions.

```

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 the current color to your macro by `\let\yourmacro=\currentcolor` and you can return to this color by the command `\setcolor\yourmacro`.

```

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\openfnoteack=\openfnoteackA
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 the next pages and their colors are independent of colors used in the main page-body. The `\openfnoteack` is defined as `\openfnoteackA` when the `\setcolor` is used first. The `\fnoteack` is initialized in `\everyjob` because the initialization is not saved to the format.

```

119 %\mathchardef\fnoteack=\pdfcolorstackinit page {0 g 0 G} % must be in \everyjob
120 \def \openfnoteackA {\pdfcolorstack\fnoteack 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 ;',
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\colordefin{\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 the 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 <selector><K><R><G><what-define>{<data>}` does the real work. It initializes the Lua variables for summation. It expands `<data>` in the group where color selectors have special meaning, then it adjusts the resulting string by `\replstring` and runs it. Example shows how the `<data>` 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 <coef>!<mod><type>` expands to `\addcolor:<mod><type> <coef>` for example it expands to `\addcolor:=K <coef>` followed by one or three or four numbers (depending on `<type>`). `<mod>` is = (use as is) or ^ (use complementary color). `<type>` is K for CMYK, R for RGB and G for GREY color space. Uppercase `<type>` informs that `\cmykcolordef` is processed and lowercase `<type>` informs that `\rgbcolordef` is processed. All variants of commands `\addcolor:<mod><type>` are defined. All of them expand to `\addcolorA <v1> <v2> <v3> <v4>` which adds the values of Lua variables. The `\rgbcolordef` uses `\addcolorA <R> <G> <B> 0` and `\cmykcolordef` uses `\addcolorA <C> <M> <Y> <K>`. 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  $\langle \textit{what-define} \rangle$  is defined as  $\langle \textit{selector} \rangle \{ \langle \textit{expanded} \_tmpb \rangle \}$ , for example `\_setcmkclor{1 0 .5 .3}`.

colors.opm

```

218 \_def\_\rgbcolordef {\_commoncolordef \_setrgbcolor krg}
219 \_def\_\cmkcolordef {\_commoncolordef \_setcmkcolor 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\_\setcmkcolor##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{-!}{-!}%
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 \_cmktoRGB #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=\_cmkcolordef % default \_colordef is \_cmkcolordef

```

Public versions of `\colordef` and `\useK` macros are declared using `\_def`, because the internal versions `\_colordef` and `\_useK` are changed during processing.

colors.opm

```

272 \_def \useK{\_useK}
273 \_def \colordef {\_colordef}
274 \_public \cmkcolordef \rgbcolordef ;

```

The `\_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

```

```

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     \_ifcename \_tmpb\_endcename \_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 ;

```

## 2.21 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  $\TeX$  run are saved here.  $\text{Op}\TeX$  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 `\gpageno` is internal page number globally numbered from one and `\pageno` is a page number (`\the\pageno`) used in pagination (they may differ). Each page begins with `\_Xpage`. The `\label` is a label used by user in `\label[\label]` and `\text` is a text which should be referenced (the number of section or table, for example 2.3.14). The `\title` is the title of the chapter (`\level=1`, `\type=chap`), section (`\level=2`, `\type=sec`), subsection (`\level=3`, `\type=secc`). The `\_Xpage` is written at the 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 an empty `test.ref` file in your directory.

```

3 \_codedecl \openref {File for references <2021-04-13>} % preloaded in format

```

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     \_gfnoteum=0 \_lfnoteum=0 \_mnoteum=0
17     \_openrefA{\_string\_inputref}%
18   \_fi
19 }

```

If the file does not exist 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 a dummy in this case.

```

28 \_def\_wrefrelax#1#2{}
29 \_let\_wref=\_wrefrelax

```

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 \langle macro \rangle \langle data \rangle` is redefined in order to save the line `\langle macro \rangle \langle data \rangle` to the .ref file using asynchronous `\write` primitive. Finally, the `\_openref` destroys itself, because we need not open the file again.

The `\_wref \langle csname \rangle \langle params \rangle` does exactly `\write\_reffile{\string \langle csname \rangle \langle params \rangle}` in this case and `\_ewref \langle csname \rangle \langle params \rangle` does `\write\_reffile{\string \langle csname \rangle \langle expanded-params \rangle}`.

ref-file.opm

```

43 \_def\_openref {%
44   \_ifx \_wref\_wrefrelax \_openrefA{\_string\openref}\_fi
45   \_gdef\_openref{}%
46 }
47 \_def\_openrefA #1{%
48   \_immediate\_openout\_reffile="\_jobname.ref"\_relax
49   \_gdef\_wref ##1##2{\_write\_reffile{\_bslash\_csstring##1##2}}%
50   \_immediate\_write\_reffile {\_pcent\_pcent\_space OPTeX <\_optexversion> - REF file (#1)}%
51   \_immediate\_wref \Xrefversion{\\_REFversion}}%
52 }
53 \_def\_ewref #1#2{\_edef\_ewrefA{#2}\_ea\_wref\_ea#1\_ea{\_ewrefA}}
54 \_def\openref{\\_openref}

```

We are using the convention that the macros used in .ref file are named `\_X\langle foo \rangle`. If there is a new version of OpTeX with a 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 the first line of .ref file is in the form

`\Xrefversion{\langle version \rangle}`

We can check the version compatibility by this macro. Because OPmac does not understand `\_Xrefversion` we use `\Xrefversion` (with a different number of `\langle version \rangle` form OPmac) here. The result: OPmac skips the .ref files produced by OpTeX and vice versa.

ref-file.opm

```

72 \_def\_REFversion{6} % actual version of .ref files in OpTeX
73 \_def\_Xrefversion#1{\_ifnum #1=\_REFversion\_relax \_else \_endinput \_fi}
74 \_public \Xrefversion ; % we want to ignore .ref files generated by OPmac

```

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{\langle definitions of your ref macros \rangle}`. This command sends to .ref file your `\langle definitions of your ref macros \rangle` 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 `\langle definition of your ref macros \rangle` when the catcode of # is 12 because we needn't duplicate each # in the .ref file.

ref-file.opm

```

94 \_def\_refdecl{\_bgroup \_catcode`\#=12 \_refdeclA}
95 \_def\_refdeclA #1{\_egroup\_openref
96   \_immediate\_write\_reffile {\_pcent\_pcent\_space \_string \refdecl:}%
97   \_immediate\_write\_reffile {\_detokenize{#1}}%
98 }
99 \_public \refdecl ;

```

## 2.22 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 2.21 for more information about .ref file concept.

references.opm

```

3 \_codedecl \ref {References <2021-04-13>} % preloaded in format

```

`\_Xpage \langle gpageno \rangle \langle pageno \rangle` saves the parameter pair into `\_currpage`. Resets `\_lfnotenum`; it is used if footnotes are numbered from one at each page.

```
10 \_def\_Xpage#1#2{\_def\_currpage{#1}{#2}}\_lfnotennum=0 }
```

Counter for the number of unresolved references `\_unresolvedrefs`.

```
16 \_newcount\_unresolvedrefs
17 \_unresolvedrefs=0
```

`\_Xlabel {<label>}{<text>}` saves the `<text>` to `\_lab:<label>` and saves `[pg:<pageno>]{<pageno>}` to `\_pgref:<label>`.

```
24 \_def\_Xlabel#1#2{\_sdef{\_lab:#1}{#2}}\_sxdef{\_pgref:#1}{\_ea\_bracketspg\_currpage}}
25 \_def\_bracketspg#1#2{[pg:#1]{#2}}
```

`\_label[<label>]` saves the declared label to `\_lastlabel` and `\_wlabel{<text>}` uses the `\_lastlabel` and activates `\_wref\_Xlabel{<label>}{<text>}`.

```
33 \_def\_label[#1]{\_isempty{#1}\_iftrue \_global\_let \_lastlabel=\_undefined
34 \_else \_isdefined{10:#1}%
35 \_iftrue \_slideshowhook\_opwarning{Duplicated label [#1], ignored}\_else \_xdef\_lastlabel{#1}\_fi
36 \_fi \_ignorespaces
37 }
38 \_let \_slideshowhook=\_relax % redefined if \_slides + \_slideshow.
39 \_def\_wlabel#1{%
40 \_ifx\_lastlabel\_undefined \_else
41 \_dest[ref:\_lastlabel]%
42 \_printlabel\_lastlabel
43 \_ewref \_Xlabel {\_lastlabel}{#1}}%
44 \_sxdef{\_lab:\_lastlabel}{#1}\_sxdef{10:\_lastlabel}{}%
45 \_global\_let\_lastlabel=\_undefined
46 \_fi
47 }
48 \_public \_label \_wlabel ;
```

`\_ref[<label>]` uses saved `\_lab:<label>` and prints (linked) `<text>`. If the reference is backward 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 the first run of T<sub>E</sub>X and we print a warning and do `\_openref`.

`\_pgref[<label>]` uses `{<pageno>}{<pageno>}` from `\_pgref:<label>` and prints (linked) `<pageno>` using `\_ilink` macro.

```
61 \_def\_ref[#1]{\_isdefined{\_lab:#1}%
62 \_iftrue \_ilink[ref:#1]{\_csname \_lab:#1\_endcsname}%
63 \_else ??\_opwarning[label [#1] unknown. Try to TeX me again]%
64 \_incr\_unresolvedrefs \_openref
65 \_fi
66 }
67 \_def\_pgref[#1]{\_isdefined{\_pgref:#1}%
68 \_iftrue \_ea\_ea\_ea\_ilink \_csname \_pgref:#1\_endcsname
69 \_else ??\_opwarning[pg-label [#1] unknown. Try to TeX me again]%
70 \_incr\_unresolvedrefs \_openref
71 \_fi
72 }
73 \_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.

```
81 \_def\_printlabel#1{}
82 \_def\_showlabels {%
83 \_def\_printlabel##1{\_vbox to\_zo{\_vss\_llap{\_labelfont[#1]}\_kern1.7ex}}%
84 \_fontdef\_labelfont{\_setfontsize{at10pt}\_setfontcolor{blue}\_tt}
85 }
86 \_public \_showlabels ;
```



## 2.23 Hyperlinks

There are four types of internal links and one type of external link:

- `ref:<label>` – the destination is created when `\label[<label>]` is used, see also the section 2.22.
- `toc:<tocrefnum>` – the destination is created at chap/sec/secc titles, see also the section 2.24.
- `pg:<gpageno>` – the destination is created at beginning of each page, see also the section 2.18.
- `cite:<bibpart>/<bibnum>` – the destination is created in bibliography reference, see section 2.32.1.
- `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 the 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 \ulink {Hyperlinks <2021-04-17>} % 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 `\_desttheight`. The reason is that the destination is exactly at the top border of the PDF viewer but we want to see the line where the destination is. The destination box is positioned by a different way dependent on the current vertical or horizontal mode.

hyperlinks.opm

```
16 \_def\_desttheight{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 to\_zo{\_kern-\_desttheight \_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 its meaning is set to `\_linkactive`. The `\_linkactive` is implemented by `\_pdfstartlink..._pdfendlink` primitives. The `<color>` is the color of the link text generated by the `\_linkactive` macro. The `<color>` parameter can be overwritten by definition of `\_<type>linkcolor`. For example `\def\_toclinkcolor{\Red}` means that links from table of contents are in red. This is similar concept as `\_def\_tocborder`.

`\ilink[<type>:<spec>]{<text>}` is equivalent to `\link` but the `<color>` is used from `\hyperlinks` declaration (or it is overwritten by `\def\_<type>linkcolor`).

`\_linkdims` are default dimensions of the link area.

hyperlinks.opm

```
48 \_protected\_def\_linkactive[#1:#2]#3#4{\_leavevmode\_pdfstartlink \_linkdims
49   attr{\_pdfborder{#1}} goto name{#1:#2}\_relax
50   {\_localcolor\_trycs{#1linkcolor}{#3}#4}\_pdfendlink
51 }
52 \_protected\_def\_link[#1]#2#3{\_leavevmode{#3}}
53 \_def\_ilink[#1]#2{\_link[#1]{}{#2}}
54 \_def\_linkdims{height.9em depth.3em}
55 \_public \ilink \link ;
```

`\ulink[<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

```
65 \_protected\_def\_urlactive[#1:#2]#3#4{\_leavevmode{\_escapechar=-1
66   \_pdfstartlink \_linkdims attr{\_pdfborder{#1}}%
67   user{Subtype/Link/A <</Type/Action/S/URI/URI(\_detokenize{#2})>>}\_relax
68   {\_localcolor\_trycs{#1linkcolor}{#3}#4}\_pdfendlink}%
69 }
70 \_def\_ulink[#1]#2{\_leavevmode{#2}}
71 \_public \ulink ;
```

The `\pdfstartlink` primitive uses `attr{\pdfborder{<type>}}` in its parameter (see `\linkactive` or `\urlactive` macros). The `\pdfborder{<type>}` macro expands to `/C[? ? ?] /Border[0 0 .6]` if the `\_border{<type>}` macro (i.e. `\_refborder`, `\_citeborder`, `\_tocborder`, `\_pgborder`, `\_urlborder`, `\_fntborder` or `\_fnfborder`) is defined. Users 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

```
85 \def\_pdfborder#1{\_ifcsize\_#1border\_endcsize
86 /C [\_csname\_#1border\_endcsize] /Border [0 0 .6]\_else /Border [0 0 0]\_fi
87 }
```

`\hyperlinks{<ilink_color>}{<ulink_color>}` activates `\dest`, `\link`, `\ilink`, `\ulink` in order they create links. These macros are redefined here to their “active” version.

hyperlinks.opm

```
95 \def\_hyperlinks#1#2{%
96 \_let\_dest=\_destactive \_let\_link=\_linkactive
97 \_def\_ilink[#1]##2{\_link[#1]{#1}{#2}}%
98 \_def\_ulink[#1]##2{\_urlactive[url:#1]{#2}{#2}}%
99 \_public \dest \ilink \ulink \link ;%
100 }
101 \_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 nonbreakable 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

```
115 \def\_url#1{%
116 \_def\_tmpa{#1}\_replstring\_tmpa {\|}{}%
117 {\_escapechar=-1 \_ea}\_ea\_edef\_ea\_tmpa\_ea{\_detokenize\_ea{\_tmpa}}%
118 \_def\_tmpb{#1}\_replstring\_tmpb {\|}{\_urlbskip}%
119 \_replstring\_tmpb {//} {\_urlskip\_urlslashtslasht\_urlbskip}%
120 \_replstring\_tmpb {/} {\_urlskip/\_urlbskip}%
121 \_replstring\_tmpb {.} {\_urlskip.\_urlbskip}%
122 \_replstring\_tmpb {?} {\_urlskip?\_urlbskip}%
123 \_replstring\_tmpb {=} {\_urlskip=\_urlbskip}%
124 \_ea\_replstring\_ea\_tmpb \_ea{\_string &} {\_urlbskip\_char`& \_urlskip}%
125 \_ea\_replstring\_ea\_tmpb \_ea{\_bslash} {\_penalty0}%
126 \_ea\_ulink \_ea{\_tmpa} {\_urlfont\_tmpb\_null}%
127 }
128 \_def\_urlfont{\_tt}
129 \_def\_urlskip{\_null\_nobreak\_hskip0pt plus0.05em\_relax}
130 \_def\_urlbskip{\_penalty100 \_hskip0pt plus0.05em\_relax}
131 \_def\_urlslashtslasht{/\_urlskip/}
132
133 \_public \url ;
```

## 2.24 Making table of contents

maketoc.opm

```
3 \_codedecl \maketoc {Macros for maketoc <2020-02-09>} % preloaded in format
```

`\_Xtoc {<level>}{<type>}{<number>}{<o-title>}{<title>}` (in `.ref` file) reads given data and appends them to the `\_toclist` as `\_tocline{<level>}{<type>}{<number>}{<o-title>}{<title>}{<gpageno>}{<pageno>}` where:

- `<level>`: 0 reserved, 1: chapter, 2: section, 3: subsection
- `<type>`: the type of the level, i.e. chap, sec, secc
- `<number>`: the number of the chapter/section/subsection in the format 1.2.3
- `<o-title>`: outlines title, if differs from `<title>`.
- `<title>`: the title text
- `<gpageno>`: the page number numbered from 1 independently of pagination
- `<pageno>`: the page number used in the pagination

The last two parameters are restored from previous `\_Xpage{<pageno>}{<gpageno>}`, data were saved in the `\_currpage` macro.

We read the `<title>` parameter by `\scantoeol` from `.ref` file because the `<title>` can include something like ``{``.

```

26 \_def\_toclist{}
27 \_newif \_ifischap \_ischapfalse
28
29 \_def\_Xtoc#1#2#3#4{\_ifnum#1=0 \_ischaptrue\_fi
30   \_addto\_toclist{\_tocline{#1}{#2}{#3}{#4}}\_scantoeol\_XtocA}
31 \_def\_XtocA#1{\_addto\_toclist{#1}}\_ea\_addto\_ea\_toclist\_ea{\_currpage}}

```

maketoc.opm

`\_tocline{<level>}{<type>}{<number>}{<o-title>}{<title>}{<gpageno>}{<pageno>}` prints the record to the table of contents. It opens group, reduces `\_leftskip`, `\_rightskip`, runs the `\everytocline` (user can customise the design of TOC here) and runs `\_tocl: <level> {<number>}{<title>}{<pageno>}` macro. This macro starts with vertical mode, inserts one record with given `<level>` and it should end by `\_tocpar` which returns to horizontal mode. The `\_tocpar` appends `\_nobreak \_hskip-2\_iindent\_null \_par`. This causes that the last line of the record is shifted outside the margin given by `\_rightskip`. A typical record (with long `<title>`) looks like this:

```

      |                               |
\llap{<number>} text text text text text
                text text text text text
                text text ..... <pageno>

```

Margins given by `\_leftskip` and `\_rightskip` are denoted by `|` in the example above.

`\tocrefnum` is the global counter of all TOC records (used by hyperlinks).

```

56 \_newcount \_tocrefnum
57 \_def\_tocline#1#2#3#4#5#6#7{%
58   \_advance\_tocrefnum by1
59   \_bgroup
60     \_leftskip=\_iindent \_rightskip=2\_iindent
61     \_ifischap \_advance\_leftskip by \_iindent \_fi
62     \_def\_pgn{\_ilink[pg:#6]}%
63     \_the\_everytocline
64     \_ifcsname \_tocl:#1\_endcsname
65     \_cs{\_tocl:#1}{#3}{\_scantextokens{#5}}{#7}\_par
66     \_fi
67   \_egroup
68 }
69 \_public \_tocrefnum ;

```

maketoc.opm

You can re-define default macros for each level of tocline if you want.

Parameters are `{<number>}{<title>}{<pageno>}`.

```

76 \_sdef{\_tocl:1}#1#2#3{\_nofirst\_bigskip \_bf\_llaptoclink{#1}{#2}\_hfill \_pgn{#3}\_tocpar}
77 \_sdef{\_tocl:2}#1#2#3{\_llaptoclink{#1}{#2}\_tocdotfill \_pgn{#3}\_tocpar}
78 \_sdef{\_tocl:3}#1#2#3{\_advance\_leftskip by\_iindent \_cs{\_tocl:2}{#1}{#2}{#3}}

```

maketoc.opm

The auxiliary macros are:

- `\_llaptoclink<text>` does `\_noindent\_llap{<linked text>}`.
- `\_tocdotfill` creates dots in the TOC.
- `\_nofirst` macro applies the `\_macro` only if we don't print the first record of the TOC.
- `\_tocpar` finalizes one TOC record with `\_llap{<pageno>}`.
- `\_pgn{<pageno>}` creates `<pageno>` as link to real `<gpageno>` saved in #6 of `\_tocline`. This is temporarily defined in the `\_tocline`.

maketoc.opm

```

93 \_def\_llaptoclink#1{\_noindent
94   \_llap{\_ilink[toc:\_the\_tocrefnum]{\_enspace#1\_kern.4em}\_kern.1em}}
95 \_def\_tocdotfill{\_nobreak\_leaders\_hbox to.8em{\_hss.\_hss}\_hskip 1em plus1fill\_relax}
96 \_def\_nofirst #1{\_ifnum \_lastpenalty=11333 \_else #1\_fi}
97 \_def\_tocpar{\_nobreak \_hskip-2\_iindent\_null \_par}

```

If you want a special formatting of TOC with adding more special lines (not generated as titles from `\chap`, `\sec`, `\secc`), you can define `\addtotoc{<level>}{<type>}{<number>}{<o-title>}{<title>}` macro:

```

\def\addtotoc#1#2#3#4#5{%
  \incr\_tocrefnum
  \_dest[toc:\_the\_tocrefnum]%
  \_ewref\_Xtoc{#1}{#2}{#3}{#4}{#5}%
}

```

and you can declare special lines (or something else) as an unused level (10 in the following example):

```
\sdef\_tocl:10}#1#2#3{\medskip\hbox{Blue #2}\medskip}
```

Now, users can add a blue line into TOC by

```
\addtotoc{10}{blue-line}{}{\relax}{\textit{blue text to be added in the TOC}}
```

anywhere in the document. Note that `\relax` in the fourth parameter means that outline will be not generated. And second parameter `blue-line` is only a comment (unused in macros).

`\maketoc` prints warning if TOC data is empty, else it creates TOC by running `\_toclist`

maketoc.opm

```

127 \_def\_maketoc{\_par \_ifx\_toclist\_empty
128   \_opwarning{\_noexpand\_maketoc -- data unavailable, TeX me again}\_openref
129   \_incr\_unresolvedrefs
130 \_else \_begingroup
131   \_tocrefnum=0 \_penalty11333
132   \_the\_regtoc \_toclist
133 \_endgroup \_fi
134 }

```

`\regmacro` appends its parameters to `\_regtoc`, `\_regmark` and `\_regoul`. These token lists are used in `\maketoc`, `\_begoutput` and `\pdfunidef`.

maketoc.opm

```

142 \_newtoks \_regtoc \_newtoks \_regmark \_newtoks \_regoul
143
144 \_def\_regmacro #1#2#3{%
145   \_toksapp\_regtoc{#1}\_toksapp\_regmark{#2}\_toksapp\_regoul{#3}%
146 }
147 \_public \maketoc \regmacro ;

```

## 2.25 PDF outlines

### 2.25.1 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`.

The first step, the `\outlines` macro sets `\_tocline` to `\_outlinesA` and calculates the number of direct descendants of each record. The second step, the `\outlines` macro sets `\_tocline` to `\_outlinesB` and it uses prepared data and creates 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), the third one for level 2 (sections) etc. The number of direct descendants will be stored in these macros after the first step is finished. Each new outline of a given level increases the `<num>` at the 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 the 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`  $\{\langle text \rangle\}$  inserts one outline with zero direct descendants. It creates a link destination of the type `oul:\langle num \rangle` into the document (where `\insertoutline` is used) and the link itself is created too in the outline.

outlines.opm

```

3 \_codedecl \outlines {PDF outlines <2021-02-09>} % 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     \_incr\_unresolvedrefs
9   \_else
10    \_ifx\_dest\_destactive \_else
11      \_opwarning{\_noexpand\outlines doesn't work when \_noexpand\hyperlinks isn't declared}\_fi
12    {\_let\_tocline=\_outlinesA
13     \_count0=0 \_count1=0 \_count2=0 \_count3=0 \_toclist % calculate numbers o childs
14     \_def\_outlinelevel{#1}\_let\_tocline=\_outlinesB
15     \_tocrefnum=0 \_count0=0 \_count1=0 \_count2=0 \_count3=0
16     \_toclist}% create outlines
17   \_fi
18 }
19 \_def\_outlinesA#1#2#3#4#5#6#7{%
20   \_isequal{\relax}{#4}\_iffalse
21     \_advance\_count#1 by1
22     \_ifcase#1\_or
23       \_addoneol{\_ol:\_the\_count0}\_or
24       \_addoneol{\_ol:\_the\_count0:\_the\_count1}\_or
25       \_addoneol{\_ol:\_the\_count0:\_the\_count1:\_the\_count2}\_or
26       \_addoneol{\_ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}\_fi
27   \_fi
28 }
29 \_def\_addoneol#1{%
30   \_ifcsname #1\_endcsname
31     \_tmpnum=\_csname#1\_endcsname\_relax
32     \_advance\_tmpnum by1 \_sxddef{#1}{\_the\_tmpnum}%
33   \_else \_sxddef{#1}{1}%
34   \_fi
35 }
36 \_def\_outlinesB#1#2#3#4#5#6#7{%
37   \_advance\_tocrefnum by1
38   \_isequal{\relax}{#4}\_iffalse
39     \_advance\_count#1 by1
40     \_ifcase#1%
41       \_tmpnum=\_trycs{\_ol:\_the\_count0}{0}\_or
42       \_tmpnum=\_trycs{\_ol:\_the\_count0:\_the\_count1}{0}\_relax\_or
43       \_tmpnum=\_trycs{\_ol:\_the\_count0:\_the\_count1:\_the\_count2}{0}\_relax\_or
44       \_tmpnum=\_trycs{\_ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}{0}\_relax\_or
45       \_tmpnum = 0\_relax\_fi
46     \_isempty{#4}\_iftrue \_pdfunidef\_tmp{#5}\_else \_pdfunidef\_tmp{#4}\_fi
47     \_outlinesC{toc:\_the\_tocrefnum}{\_ifnum#1<\_outlinelevel\_space\_else-\_fi}{\_tmpnum}{\_tmp}%
48   \_fi
49 }
50 \_def\_outlinesC#1#2#3#4{\_pdfoutline goto name{#1} count #2#3{#4}\_relax}
51
52 \_newcount\_oulnum
53 \_def\_insertoutline#1{\_incr\_oulnum
54   \_pdfdest name{oul:\_the\_oulnum} xyz\_relax
55   \_pdfunidef\_tmp{#1}%
56   \_pdfoutline goto name{oul:\_the\_oulnum} count0 {\_tmp}\_relax
57 }
58 \_public \outlines \insertoutline ;

```

## 2.25.2 Strings in PDF outlines

There are only two encodings for PDF strings (used in PDFoutlines, PDFinfo, etc.). The first one is PDFDocEncoding which is single-byte encoding, but it misses most international characters.

The second encoding is Big Endian UTF-16 which is implemented in this file. It encodes a single character in either two or four bytes. This encoding is T<sub>E</sub>X-discomfortable because it looks like

```
<FEFF 0043 0076 0069 010D 0065 006E 00ED 0020 006A 0065 0020 007A 00E1 0074
011B 017E 0020 0061 0020 0078 2208 D835DD44>
```

This example shows a hexadecimal PDF string (enclosed in <> as opposed to the literal PDF string enclosed in ( )). In these strings each byte is represented by two hexadecimal characters (0–9, A–F). You can tell the encoding is UTF-16BE, because it starts with “Byte order mark” FEFF. Each unicode character is then encoded in one or two byte pairs. The example string corresponds to the text “Cvičení je zátěž a  $x \in \mathbb{M}$ ”. Notice the 4 bytes for the last character,  $\mathbb{M}$ . (Even the whitespace would be OK in a PDF file, because it should be ignored by PDF viewers, but LuaTeX doesn’t allow it.)

pdfuni-string.opm

```
3 \_codedecl \pdfunidef {PDFUnicode strings for outlines <2021-02-08>} % preloaded in format
```

`\_hexprint` is a command defined in Lua, that scans a number and expands to its UTF-16 Big Endian encoded form for use in PDF hexadecimal strings.

pdfuni-string.opm

```
10 \bgroup
11 \_catcode`\%=12
12 \_gdef\_hexprint{\_directlua{
13   local num = token.scan_int()
14   if num < 0x10000 then
15     tex.print(string.format("%04X", num))
16   else
17     num = num - 0x10000
18     local high = bit32.rshift(num, 10) + 0xD800
19     local low = bit32.band(num, 0x3FF) + 0xDC00
20     tex.print(string.format("%04X%04X", high, low))
21   end
22 }}
23 \egroup
```

`\pdfunidef\macro{<text>}` does more things than only converting to hexadecimal PDF string. The `<text>` can be scanned in verbatim mode (it is true because `\_Xtoc` reads the `<text>` in verbatim mode). First `\edef` do `\_scantextokens\unexpanded` and second `\edef` expands the parameter according to current values on selected macros from `\_regoul`. Then `\_removeoutmath` converts `.. $x^2$ ..` to `..x2..`, i.e. removes dollars. Then `\_removeoutbraces` converts `..{x}..` to `..x..`. Finally, the `<text>` is detokenized, spaces are preprocessed using `\_replstring` and then the `\_pdfunidef` is repeated on each character. It calls the `\_directlua` chunk to print hexadecimal numbers in the macro `\_hexprint`.

Characters for quotes (and separators for quotes) are activated by first `\_scantextokens` and they are defined as the same non-active characters. But `\_regoul` can change this definition.

pdfuni-string.opm

```
41 \_def\_pdfunidef#1#2{%
42   \_begingroup
43     \_catcodetable\_optexcatcodes \_adef{"}\_adef{'}%
44     \_the\_regoul \_relax % \_regmacro alternatives of logos etc.
45     \_ifx\_savedttchar\_undefined \_def#1{\_scantextokens{\_unexpanded{#2}}}%
46     \_else \_lccode\;=\_savedttchar \_lowercase{\_prepinverb#1;}{#2}\fi
47     \_edef#1{#1}%
48     \_escapechar=-1
49     \_edef#1{#1\_empty}%
50     \_escapechar=-\\
51     \_ea\_edef \_ea#1\_ea{\_ea\_removeoutmath #1$\_end$}% $x$ -> x
52     \_ea\_edef \_ea#1\_ea{\_ea\_removeoutbraces #1{\_end}}% {x} -> x
53     \_edef#1{\_detokenize\_ea{#1}}%
54     \_replstring#1{ }{ }% text text -> text{ }text
55     \_catcode`\%=12 \_let\;=\_bslash
56     \_edef\_out{<FEFF}
57     \_ea\_pdfunidefB#1~% text -> \_out in octal
58     \_ea
59   \_endgroup
60   \_ea\_def\_ea#1\_ea{\_out>}
61 }
62 \_def\_pdfunidefB#1{%
63   \_ifx~#1\_else
64     \_edef\_out{\_out \_hexprint `#1}
65   \_ea\_pdfunidefB \_fi
66 }
67
```



```

68 \def\removeoutbraces #1#{#1\removeoutbracesA}
69 \def\removeoutbracesA #1{\ifx\end#1\else #1\ea\removeoutbraces\fi}
70 \def\removeoutmath #1$#2$#{1\ifx\end#2\else #2\ea\removeoutmath\fi}

```

The `\_prepinverb` macro `\_prepinverb{separator}{text}`, e.g. `\_prepinverb\tmpb|{aaa|bbb|ccc|ddd|eee}` does `\def\tmpb{su}{aaa}bbb{su}{ccc}ddd{su}{eee}` where `su` is `\scantextokens\unexpanded`. It means that in-line verbatim are not argument of `\scantextoken`. First `\edef\tmpb` tokenizes again the `text` but not the parts which were in the the in-line verbatim.

pdfuni-string.opm

```

81 \def\_prepinverb#1#2#3{\def#1}%
82 \def\_dotmpb ##1#2##2{\addto#1{\scantextokens{\unexpanded{##1}}}%
83 \ifx\end##2\else\ea\_dotmpbA\ea##2\fi}%
84 \def\_dotmpbA ##1#2{\addto#1{##1}\dotmpb}%
85 \dotmpb#3#2\end
86 }

```

The `\regmacro` is used in order to set the values of macros `\em`, `\rm`, `\bf`, `\it`, `\bi`, `\tt`, `\/` and `\sim` to values usable in PDF outlines.

pdfuni-string.opm

```

94 \regmacro {}{\let\em=\empty \let\rm=\empty \let\bf=\empty
95 \let\it=\empty \let\bi=\empty \let\tt=\empty \let\/=\empty
96 \let\sim=\space
97 }
98 \public \pdfunidef ;

```

## 2.26 Chapters, sections, subsections

sections.opm

```

3 \codedec \chap {Titles, chapters, sections, subsections <2021-03-03>} % 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.

sections.opm

```

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 using `\scantoel` and `\prnttit`. It means that the parameter is separated by end of line and inline verbatim is allowed. The same principle is used in the `\chap`, `\sec`, and `\secc` macros.

sections.opm

```

25 \def\_prnttit #1{\vglue\_titskip
26 {\leftskip=0pt plus1fill \rightskip=\leftskip % centering
27 \_titfont \noindent \scantextokens{#1}\par}%
28 \nobreak\_bigskip
29 }
30 \def\_tit{\scantoel\_prnttit}
31
32 \public \tit ;

```

You can re-define `\_printchap`, `\_printsec` or `\_printsecc` macros if another design of section titles is needed. These macros get the `title` text in its parameter. The common recommendations for these macros are:

- Use `\_abovetitle{penaltyA}{skipA}` and `\_belowtitle{skipB}` for inserting vertical material above and below the section title. The arguments of these macros are normally used, i.e. `\_abovetitle` inserts `penaltyA skipA` and `\_belowtitle` inserts `skipB`. But there is an exception: if `\_belowtitle{skipB}` is immediately followed by `\_abovetitle{penaltyA}{skipA}` (for example section title is immediately followed by subsection title), then only `skipA` is generated, i.e. `skipB penaltyA skipA` is reduced only to `skipA`. The reason for such behavior: we don't want to duplicate vertical skip and we don't want to use the negative penalty in such cases. Moreover, `\_abovetitle{penaltyA}{skipA}` takes previous whatever vertical skip (other than from `\_belowtitle`) and generates only greater from this pair of skips. It means that `whatever-skip penaltyA skipA` is transformed to `penaltyA max(whatever-skip skipA)`. The

reason for such behavior: we don't want to duplicate vertical skips (from `\belowlistskip`, for example) above the title.

- Use `\_printrefnum[⟨pre⟩@⟨post⟩]` 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 the label (if `\label[⟨label⟩]` precedes) and saves references for the 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 into more pages.
- You can use `\_firstnoindent` in order to the first paragraph after the title is not indented.

```

72 \def\_printchap #1{\_vfill\_supereject
73   \vglue\_medskipamount % shifted by topkip+\medskipamount
74   {\_chapfont \noindent \mtext{chap} \_printrefnum[@]\_par
75     \nobreak\_smallskip
76     \noindent \raggedright #1\_nbp}\_mark{}}%
77   \nobreak \belowtitle{\_bigskip}%
78   \_firstnoindent
79 }
80 \def\_printsec#1{\_par
81   \abovetitle{\_penalty-400}\_bigskip
82   {\_secfont \noindent \raggedright \_printrefnum[@\_quad]#1\_nbp}\_insertmark{#1}%
83   \nobreak \belowtitle{\_medskip}%
84   \_firstnoindent
85 }
86 \def\_printsecc#1{\_par
87   \abovetitle{\_penalty-200}{\_medskip\_smallskip}
88   {\_seccfont \noindent \raggedright \_printrefnum[@\_quad]#1\_nbp}%
89   \nobreak \belowtitle{\_medskip}%
90   \_firstnoindent
91 }

```

sections.opm

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, see the [OpTeX trick 0033](#))

```

105 \_newcount\_sectionlevel
106 \def \_secinfo {\_ifcase \_sectionlevel
107   part\_or chap\_or sec\_or secc\_or seccc\_fi
108 }

```

sections.opm

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 don'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  $\langle num \rangle.\langle num \rangle.\langle num \rangle$ .

```

136 \_newcount \_chapnum % chapters
137 \_newcount \_secnum % sections
138 \_newcount \_seccnum % subsections

```

sections.opm

```

139 \_newcount \_tnum      % table numbers
140 \_newcount \_fnum      % figure numbers
141 \_newcount \_dnum      % numbered display maths
142
143 \_def \_chapx {\_secx \_secnum=0 \_lfnotenumber=0 }
144 \_def \_secx {\_seccx \_seccnum=0 \_tnum=0 \_fnum=0 \_dnum=0 \_resetABCDE }
145 \_def \_seccx {}
146
147 \_def \_thechapnum {\_the\_chapnum}
148 \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum}
149 \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum.\_the\_seccnum}
150 \_def \_thetnum {\_othe\_chapnum.\_othe\_secnum.\_the\_tnum}
151 \_def \_thefnum {\_othe\_chapnum.\_othe\_secnum.\_the\_fnum}
152 \_def \_thednum {\_the\_dnum}
153
154 \_def \_othe #1.{\_ifnum#1>0 \_the#1.\_fi}

```

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

```

162 \_newif \_ifnotoc \_notocfalse \_def \_notoc {\_global\_notoctrue}
163 \_newif \_ifnonum \_nonumfalse \_def \_nonum {\_global\_nonumtrue}
164 \_def \_resetnonumnotoc {\_global\_notocfalse \_global\_nonumfalse}
165 \_public \_notoc \_nonum ;

```

The `\chap`, `\sec`, and `\secc` macros are implemented here. The `\_inchap`, `\_insec` and `\_insecc` macros do the real work. First, we read the optional parameter [*label*], if it exists. The `\chap`, `\sec` and `\secc` macro reads its parameter using `\scantoeol`. This causes that they cannot be used inside other macros. Use `\_inchap`, `\_insec`, and `\_insecc` macros directly in such case.

sections.opm

```

176 \_optdef \_chap [] {\_trylabel \_scantoeol \_inchap}
177 \_optdef \_sec [] {\_trylabel \_scantoeol \_insec}
178 \_optdef \_secc [] {\_trylabel \_scantoeol \_insecc}
179 \_def \_trylabel {\_istokempty \_opt \_iffalse \_label {\_the\_opt} \_fi}
180
181 \_def \_inchap #1 {\_par \_sectionlevel=1
182   \_def \_savedtitle {#1}% saved to .ref file
183   \_ifnonum \_else {\_globaldefs=1 \_incr \_chapnum \_chapx} \_fi
184   \_edef \_therefnum {\_ifnonum \_space \_else \_thechapnum \_fi}%
185   \_printchap {\_scantextokens{#1}}%
186   \_resetnonumnotoc
187 }
188 \_def \_insec #1 {\_par \_sectionlevel=2
189   \_def \_savedtitle {#1}% saved to .ref file
190   \_ifnonum \_else {\_globaldefs=1 \_incr \_secnum \_secx} \_fi
191   \_edef \_therefnum {\_ifnonum \_space \_else \_theseccnum \_fi}%
192   \_printsec {\_scantextokens{#1}}%
193   \_resetnonumnotoc
194 }
195 \_def \_insecc #1 {\_par \_sectionlevel=3
196   \_def \_savedtitle {#1}% saved to .ref file
197   \_ifnonum \_else {\_globaldefs=1 \_incr \_seccnum \_seccx} \_fi
198   \_edef \_therefnum {\_ifnonum \_space \_else \_theseccnum \_fi}%
199   \_printsecc {\_scantextokens{#1}}%
200   \_resetnonumnotoc
201 }
202 \_public \_chap \_sec \_secc ;

```

The `\_printrefnum[pre]@[post]` macro is used in `\_print*` macros.

Note that the *<tite-text>* is `\detokenized` before `\_wref`, so the problem of “fragile macros” from old L<sup>A</sup>T<sub>E</sub>X never occurs. This fourth parameter is not delimited by `{...}` but by end of line. This gives possibility to have unbalanced braces in inline verbatim in titles.

sections.opm

```

213 \_def \_printrefnum [#1@#2] {\_leavevmode % we must be in horizontal mode
214   \_ifnonum \_else #1 \_therefnum #2 \_fi
215   \_wlabel \_therefnum % references, if \label[<label>]` is declared
216   \_ifnotoc \_else \_incr \_tocrefnum
217   \_dest[toc:\_the\_tocrefnum]%

```

```

218 \ewref\Xtoc{\_the\_sectionlevel}{\_secinfo}%
219 {\_therefnun}{\_theoutline}\detokenize\_ea{\_savedtitle}}%
220 \_fi
221 \_gdef\_theoutline{}%
222 }

```

`\thisoutline{<text>}` saves text to the `\_theoutline` macro. `\_printrefnum` uses it and removes it.

sections.opm

```

229 \_def\_theoutline{}
230 \_def\_thisoutline#1{\_gdef\_theoutline{#1}}
231 \_public \thisoutline ;

```

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 doesn'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 precede) and create `<whatever-skip>` or `<skipA>` depending on what is greater. The amount of `<skipA>` is measured using `\setbox0=\vbox`.

sections.opm

```

247 \_newskip \_savedtitleskip
248 \_newskip \_savedlastskip
249 \_def\_abovetitle #1#2{\_savedlastskip=\_lastskip % <whatever-skip>
250 \_ifdim\_lastskip>\_zo \_vskip-\_lastskip \_fi
251 \_ifnum\_lastpenalty=11333 \_vskip-\_savedtitleskip \_else #1\_fi
252 \_ifdim\_savedlastskip>\_zo \_setbox0=\_vbox{#2\_global\_tmpdim=\_lastskip}%
253 \_else \_tmpdim=\_maxdimen \_fi
254 \_ifdim\_savedlastskip>\_tmpdim \_vskip\_savedlastskip \_else #2\_fi
255 }
256 \_def\_belowtitle #1{#1\_global\_savedtitleskip=\_lastskip \_penalty11333 }

```

`\nbpar` sets `\interlinepenalty` value. `\nl` is “new line” in the text (or titles), but space in toc or headlines or outlines.

sections.opm

```

263 \_def\_nbpar{\_interlinepenalty=10000\_endgraf}}
264
265 \_protected\_def\_nl{\_unskip\_hfil\_break}
266 \_regmacro {\_def\_nl{\_unskip\_space}} {\_def\_nl{\_unskip\_space}} {\_def\_nl{ }}
267 \_regmacro {\_def\_nl{\_unskip\_space}} {\_def\_nl{\_unskip\_space}} {\_def\_nl{ }}
268
269 \_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 `\_wipeepar` removes the material from `\everypar`.

sections.opm

```

278 \_def \_firstnoindent {\_global\_everypar={\_wipeepar \_setbox7=\_lastbox}}
279 \_def \_wipeepar {\_global\_everypar={}}

```

The `\mark` (for running heads) is used in `\_printsection` only. We suppose that chapters will be printed after `\vfil\break`, so users 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

```

294 \_def\_insertmark#1{\_mark{\_ifnonum\_else\_therefnun\_fi} {\_unexpanded{#1}}}

```

OpTeX 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 {\_thechapnum: \_unexpanded\_ea{\_savedtitle}}}
\_def \_formathead #1#2{\_isempty{#1}\_iffalse #1: #2\_fi}
\_headline = {%
  \ifodd \pageno
    \hfil \ea\_formathead\_firstmark{}{ }%

```

```

\else
  Chapter: \runningchap \hfil
\fi
}

```

The `\secl<number> <title-text><eol>` should be used for various levels of sections (for example, when converting from Markdown to OpTeX). `\secl1` is `\chap`, `\secl2` is `\sec`, `\secl3` is `\secc` and all more levels (for `<number>> 3`) are printed by the common `\seclp` macro. It declares only a simple design. If there is a requirement to use such more levels then the book designer can define something different here.

sections.opm

```

320 \def\secl{\afterassignment\secla \sectionlevel=}
321 \def\secla{\ifcase\sectionlevel
322   \or\ea\chap\or\ea\sec\or\ea\secc\else\ea\seclp\fi}
323 \eoldef\seclp#1{\par \ifnum\lastpenalty=0 \removelastskip\medskip\fi
324   \noindent{\bf #1}\vadjust{\nobreak}\nl\ignorepars}
325 \def\ignorepars{\isnextchar\par{\ignoresecond\ignorepars}{}}
326
327 \public \secl ;

```

The `\caption<letter>` increases `\_<letter>num` counter, defines `\_thecapnum` as `\_the<letter>num` and defines `\_thecaptitle` as language-dependent word using `\_mtext`, runs the `\_everycaption<letter>` tokens register. The group opened by `\caption` is finalized by first `\par` from an empty line or from `\vskip` or from `\endinsert`. The `\_printcaption<letter>` is called, it starts with printing of the caption. The `\cskip` macro inserts nonbreakable vertical space between the caption and the object.

sections.opm

```

342 \def\caption/#1{\def\tmpa{#1}\nospaceafter \capA}
343 \optdef\capA []{\trylabel \incaption}
344 \def\incaption {\_bgroup
345   \ifcsname\_tmpa num\endcsname \ea\incr \csname\_tmpa num\endcsname
346   \else \opwarning{Unknown caption /\tmpa}\fi
347   \edef\_thecapnum {\csname\_the\tmpa num\endcsname}%
348   \edef\_thecaptitle{\_mtext{\tmpa}}%
349   \ea\_the \csname\_everycaption\tmpa\endcsname
350   \def\par{\_nbpar\_egroup}\let\par=\par
351   \cs{printcaption\tmpa}%
352 }
353 \def \cskip {\_par\nobreak\medskip} % space between caption and the object
354
355 \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).

The `\_captionsep` inserts a separator between auto-generated caption number and the following caption text. Default separator is `\_enspace` but if the caption text starts with dot or colon, then the space is not inserted. A user can write `\caption/t: My table` and “**Table 1.1:** My table” is printed. You can re-define the `\_captionsep` macro if you want to use another separator.

sections.opm

```

374 \def \_printcaptiont {%
375   \noindent \_wlabel\_thecapnum {\bf\_thecaptitle~\_thecapnum}%
376   \narrowlastlinecentered\_iindent \_futurelet\_next\_captionsep
377 }
378 \def\_captionsep{\_ifx\_next.\_ea\_bfnext \else\_ifx\_next:\_ea\_ea\_ea\_bfnext
379   \else \_enspace \_fi\fi}
380 \def\_bfnext#1{{\bf#1}}
381 \let \_printcaptionf = \_printcaptiont % caption of figures = caption of tables

```

If you want to declare a new type of `\caption` with independent counter, you can use following lines, where `\caption/a` for Algorithms are declared:

```

\let\_printcaptiona = \_printcaptionf \let\_everycaptiona = \_everycaptionf
\newcount\_anum \addto\_secx {\_anum=0}
\def\_theanum {\_othe\_chapnum.\_the\_secnum.\_the\_anum}
\sdef{mt:a:en}{Algorithm} \sdef{mt:a:cs}{Algoritmus} % + your language...

```

The default format of `\caption` text is a paragraph in block narrower by `\_iindent` and with the last line is centered. This setting is done by the `\_narrowlastlinecentered` macro.

sections.opm

```
398 \_def\_narrowlastlinecentered#1{%
399   \_leftskip=#1plus1fil
400   \_rightskip=#1plus-1fil
401   \_parfillskip=0pt plus2fil\_relax
402 }
```

`\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

```
409 \_optdef\_eqmark []{\_trylabel \_ineqmark}
410 \_def\_ineqmark{\_incr\_dnum
411   \_ifinner\_else\_eqno \_fi
412   \_wlabel\_thednum \_hbox{\_thednum}%
413 }
414 \_public \eqmark ;
```

The `\numberedpar <letter>{\<name>}` is implemented here.

sections.opm

```
420 \_newcount\_counterA \_newcount\_counterB \_newcount\_counterC
421 \_newcount\_counterD \_newcount\_counterE
422
423 \_def\_resetABCDE {\_counterA=0 \_counterB=0 \_counterC=0 \_counterD=0 \_counterE=0 }
424
425 \_def \_theAnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterA}
426 \_def \_theBnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterB}
427 \_def \_theCnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterC}
428 \_def \_theDnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterD}
429 \_def \_theEnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterE}
430
431 \_def\_numberedpar#1#2{\_ea \_incr \_csname \_counter#1\_endcsname
432   \_def\_tmpa{#1}\_def\_tmpb{#2}\_numberedparparam}
433 \_optdef\_numberedparparam[]{%
434   \_ea \_printnumberedpar \_csname \_the\_tmpa num\_ea\_endcsname\_ea{\_tmpb}}
435
436 \_public \numberedpar ;
```

The `\_printnumberedpar \theXnum {\<name>}` 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

```
454 \_def \_printnumberedpar #1#2{\_par
455   \_noindent\_wlabel #1%
456   {\_bf #2 #1\_istoksepty\_opt\_iffalse \_space \_the\_opt \_fi.}\_space
457   \_ignorespaces
458 }
```

## 2.27 Lists, items

lists.opm

```
3 \_codedecl \begitems {Lists: begitems, enditems <2021-03-10>} % 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



```

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 `\beginitems`. So nested lists are numbered independently. Users can set initial value of `\itemnum` to another value after `\beginitems` if they want. Each level of nested lists is indented by the new `\iindent` from left. The default item mark is `\_printitem`.

The `\beginitems` 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).

```

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 \aboveliskip \fi
50   \itemnum=0 \edef*\_relax\ifmmode*\_else\_ea\_startitem\_fi}
51   \advance\leftskip by\iindent
52   \_printitem=\defaultitem
53   \the\_everylist \relax
54 }
55 \def\enditems{\_par\_belowliskip\_egroup \isnextchar\_par{}\_noindent}}
56
57 \def\_startitem{\_par \ifnum\itemnum>0 \interliskip \fi
58   \advance\itemnum by1
59   \the\_everyitem \noindent\_llap{\the\_printitem}\_ignorespaces
60 }
61 \_public \beginitems \enditems \itemnum ;

```

`\novspaces` sets `\listskipamount` to 0pt.

```

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`.

```

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 ;

```

The `\begblock` macro selects fonts from footnotes `\_fnset` and opens new indentation in a group. `\endblock` closes the group. This is implemented as an counterpart of Markdown’s Blockquotes. Redefine these macros if you want to declare different design. The [OpTeX trick 0031](#) shows how to create blocks with grey background splittable to more pages.

lists.opm

```

118 \_def\_begblock{\_bgroup\_fnset \_medskip \_advance\_leftskip by\_iindent \_firstnoindent}
119 \_def\_endblock{\_par\_medskip\_egroup\_isnextchar\_par{}{\_noindent}}
120
121 \_public \_begblock \_endblock ;

```

## 2.28 Verbatim, listings

### 2.28.1 Inline and “display” verbatim

verbatim.opm

```

3 \_codedecl \_begtt {Verbatim <2021-04-18>} % 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 }%

```

`\verbchar` $\langle char \rangle$  saves original catcode of previously declared  $\langle char \rangle$  (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  $\langle text \rangle$  by `\_printinverbatim` and closes group. Suppose that `\verbchar` 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\_verbchar#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 }{\_dsp}\_ttfont \_the\_everyintt\_relax \_readverb}%
83 \_def\_readverb ##1#1{\_printinverbatim{##1}\_endgroup}%
84 }
85 \_let \_activettchar=\_verbchar % for backward compatibility
86 \_public \verbchar \activettchar ;
```

`\begtt` is defined only as public. We don't need a private `\_begtt` variant. This macro opens a group and sets `%` as an active character (temporary). This will allow it to be used as the comment character at the same line after `\begtt`. Then `\_begtti` is run. It is defined by `\eoldef`, so users can put a parameter at the same line where `\begtt` is. This `#1` parameter is used after `\everytt` parameters settings, so users can change them locally.

The `\_begtti` macro 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 the end of the same line and it is simply ignored.

The `\_startverb` puts the scanned data to `\_prepareverbdata`. 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 `\_prepareverbdata` 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.

verbatim.opm

```
121 \_def\_begtt{\_par \_begingroup \_adeft##1\_relax{\_relax}\_begtti}
122 \_eoldef \_begtti#1{\_wipepar \_setxsize
123 \_vskip\_parskip \_ttskip
124 \_setverb
125 \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
126 \_adeft }{\_dsp}\_adeft^^I{\t}\_parindent=\_ttindent \_parskip=0pt
127 \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
128 \_protrudechars=0 % disable protrusion
129 \_the\_everytt \_relax #1\_relax \_ttfont
130 \_def\_testcommentchars##1\_iftrue{\_iffalse}\_let\_hicomments=\_relax
131 \_savemathsb \_endlinechar=^^J
132 \_startverb
133 }
134 \_ea\_def\_ea\_startverb \_ea#\_ea1\_csstring\\endtt#2^^J{%
135 \_prepareverbdata\_tmpb{#1^^J}%
136 \_ea\_printverb \_tmpb\_end
137 \_par \_restoremathsb
138 \_endgroup \_ttskip
139 \_isnextchar\_par{}{\_noindent}%
140 }
141 \_def\_prepareverbdata#1#2{\_def#1{#2}}
```

The `\_printverb` macro calls `\_printverblin` $\{\langle line \rangle\}$  repeatedly to each scanned line of verbatim text. The `\_printverb` is used from `\begtt... \endtt` and from `\verbatim` too.

The `\_testcommentchars` replaces the following `\_iftrue` to `\_iffalse` by default unless the `\commentchars` are set. So, the main body of the loop is written in the `\_else` part of the `\_iftrue` condition. The `\_printverblines{<line>}` is called here.

The `\_printverblines{<line>}` expects that it starts in vertical mode and it must do `\par` to return the vertical mode. The `\_printverblinenum` is used here: it does nothing when `\_ttline<0` else it prints the line number using `\_llap`.

`\_puttptpenalty` puts `\_tptpenalty` before second and next lines, but not before first line in each `\begtt...\endtt` environment.

verbatim.opm

```

162 \_def\_printverb #1^^J#2{%
163   \_ifx\_printverblinenum\_relax \_else \_incr\_ttline \_fi
164   \_testcommentchars #1\_relax\_relax\_relax
165   \_iftrue
166     \_ifx\_end#2 \_printcomments\_fi
167   \_else
168     \_ifx\_vcomments\_empty\_else \_printcomments \_def\_vcomments{}\\_fi
169     \_ifx\_end#2
170       \_bgroup \_adef{ }{}\\_def\t{}% if the last line is empty, we don't print it
171       \_ifcat&#1&\_egroup \_else\_egroup \_printverblines{#1}\_fi
172     \_else
173       \_printverblines{#1}%
174     \_fi
175   \_fi
176   \_ifx\_end#2 \_let\_next=\_relax \_else \_def\_next{\_printverb#2}\_fi
177   \_next
178 }
179 \_def\_printverblines#1{\_puttptpenalty \_indent \_printverblinenum \_kern\_ttshift #1\par}
180 \_def\_initverblinenum{\_tenrm \_thefontscale[700]\_ea\_let\_ea\_sevenrm\_the\_font}
181 \_def\_printverblinenum{\_llap{\_sevenrm \_the\_ttline\_kern.9em}}
182 \_def\_puttptpenalty{\_def\_puttptpenalty{\_penalty\_tptpenalty}}

```

Macro `\verbinp` uses a file read previously or opens the given file. Then it runs the parameter scanning by `\_viscanparameter` and `\_viscanminus`. Finally the `\_doverbinp` is run. At the beginning of `\_doverbinp`, we have `\_viline`= number of lines already read using previous `\verbinp`, `\_vinolines`= the number of lines we need to skip and `\_vidolines`= the number of lines we need to print. A 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 `\_prepareverbdata` and printed via `\_printverb` loop.

verbatim.opm

```

198 \_def\_verbinp #1(#2) #3 {\_par \_def\_tmpa{#3}%
199   \_def\_tmpb{#1}% cmds used in local group
200   \_ifx\_vifilename\_tmpa \_else
201     \_openin\_vifile={#3}%
202     \_global\_viline=0 \_global\_let\_vifilename=\_tmpa
203     \_ifeof\_vifile
204       \_opwarning{\_string\verbinp: file "#3" unable to read}
205       \_ea\_ea\_ea\_skiptorelax
206     \_fi
207   \_fi
208   \_viscanparameter #2+\_relax
209 }
210 \_def\_skiptorelax#1\_relax{}
211
212 \_def \_viscanparameter #1+#2\_relax{%
213   \_if$#2$\_viscanminus(#1)\_else \_viscanplus(#1+#2)\_fi
214 }
215 \_def\_viscanplus(#1+#2+){%
216   \_if$#1$\_tmpnum=\_viline
217   \_else \_ifnum#1<0 \_tmpnum=\_viline \_advance\_tmpnum by-#1
218     \_else \_tmpnum=#1
219     \_advance\_tmpnum by-1
220     \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi % (0+13) = (1+13)
221   \_fi \_fi
222   \_edef\_vinolines{\_the\_tmpnum}%
223   \_if$#2$\_def\_vidolines{0}\_else\_edef\_vidolines{#2}\_fi
224   \_doverbinp

```

```

225 }
226 \def\viscanminus(#1-#2){%
227   \if$#1$\_tmpnum=0
228     \else \_tmpnum=#1 \_advance\_tmpnum by-1 \_fi
229   \ifnum\_tmpnum<0 \_tmpnum=0 \_fi % (0-13) = (1-13)
230   \edef\_vinolines{\_the\_tmpnum}%
231   \if$#2$\_tmpnum=0
232     \else \_tmpnum=#2 \_advance\_tmpnum by-\_vinolines \_fi
233   \edef\_vidolines{\_the\_tmpnum}%
234   \doverbinput
235 }
236 \def\doverbinput{%
237   \_tmpnum=\_vinolines
238   \_advance\_tmpnum by-\_viline
239   \ifnum\_tmpnum<0
240     \openin\_vifile={\_vifilename}%
241     \global\_viline=0
242   \else
243     \edef\_vinolines{\_the\_tmpnum}%
244     \_fi
245     \vskip\_parskip \_ttskip \_wipeepar \_setxhsize
246     \begingroup
247     \ifnum\_ttline<-1 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
248     \_setverb \_adef{ }{\_dsp}\_adef{^I\t}\_parindent=\_ttindent \_parskip=0pt
249     \def\t{\_hskip \_dimexpr\_tabspace em/2\_relax}%
250     \_protrudechars=0 % disable protrusion
251     \_the\_everytt\_relax \_tmpb\_relax \_ttfont
252     \_savemathsb \_endlinechar=^^J \_tmpnum=0
253     \_loop \_ifeof\_vifile \_tmpnum=\_vinolines\_space \_fi
254       \ifnum\_tmpnum<\_vinolines\_space
255         \_vireadline \_advance\_tmpnum by1 \_repeat %% skip lines
256       \edef\_ttlinesave{\_ttline=\_the\_ttline}%
257       \ifnum\_ttline=-1 \_ttline=\_viline \_fi
258       \_tmpnum=0 \_def\_tmpb{}%
259       \ifnum\_vidolines=0 \_tmpnum=-1 \_fi
260       \_ifeof\_vifile \_tmpnum=\_vidolines\_space \_fi
261       \_loop \_ifnum\_tmpnum<\_vidolines\_space
262         \_vireadline
263         \ifnum\_vidolines=0 \_else\_advance\_tmpnum by1 \_fi
264         \_ifeof\_vifile \_tmpnum=\_vidolines\_space \_else \_visaveline \_fi %% save line
265       \_repeat
266       \_ea\_prepareverbdata \_ea \_tmpb\_ea{\_tmpb^^J}%
267       \_catcode`\_ =10 \_catcode`\_ =9 % used in \commentchars comments
268       \_ea\_printverb \_tmpb\_end
269       \global\_ttlinesave
270       \_par \_restoremathsb
271       \_endgroup
272       \_ttskip
273       \_isnextchar\_par{\}\_noindent}%
274 }
275 \def\_vireadline{\_read\_vifile to \_tmp \_incr\_viline }
276 \def\_visaveline{\_ea\_addto\_ea\_tmpb\_ea{\_tmp}}
277
278 \_public \_verbinput ;

```

`\_savemathsb`, `\_restoremathsb` pair is used in `\begtt...\endtt` or in `\verbinput` to temporary suppress the `\mathsb` because we don't need to print `\int _a` in verbatim mode if `\int_a` is really written. The `\_restoremathsb` is defined locally as `\mathsb` only if it is needed.

verbatim.opm

```

288 \def\_savemathsb{\_ifmathsb \_mathsboff \_def\_restoremathsb{\_mathsb}\_fi}
289 \def\_restoremathsb{}

```

If the language of your code printed by `\verbinput` supports the format of comments started by two characters from the beginning of the line then you can set these characters by `\commentchars⟨first⟩⟨second⟩`. Such comments are printed in the non-verbatim mode without these two characters and they look like the verbatim printing is interrupted at the places where such comments are. See the section 2.39 for good illustration. The file `optex.lua` is read by a single command `\verbinput (4-) optex.lua` here and the `\commentchars --` was set before it.

If you need to set a special character by `\commentchars` then you must set the catcode to 12 (and space to 13). Examples:

```
\commentchars //          % C++ comments
\commentchars --          % Lua comments
{\catcode`\%=12 \_ea}\commentchars %%          % TeX comments
{\catcode`\#=12 \catcode`\ =13 \_ea}\commentchars#{ } % bash comments
```

There is one limitation when T<sub>E</sub>X interprets the comments declared by `\commentchars`. Each block of comments is accumulated to one line and then it is re-interpreted by T<sub>E</sub>X. So, the ends of lines in the comments block are lost. You cannot use macros which need to scan end of lines, for example `\begtt... \endtt` inside the comments block does not work. The character % is ignored in comments but you can use \% for printing or % alone for de-activating `\_endpar` from empty comment lines.

Implementation: The `\commentchars⟨first⟩⟨second⟩` redefines the `\_testcommentchars` used in `\_printverb` in order to it removes the following `\_iftrue` and returns `\_iftrue` or `\_iffalse` depending on the fact that the comment characters are or aren't present at the beginning of tested line. If it is true (`\ifnum` expands to `\ifnum 10>0`) then the rest of the line is added to the `\_vcomments` macro.

The `\_hicomments` is `\relax` by default but it is redefined by `\commentchars` in order to keep no-colored comments if we need to use feature from `\commentchars`.

The accumulated comments are printed whenever the non-comment line occurs. This is done by `\_printcomments` macro. You can re-define it, but the main idea must be kept: it is printed in the group, `\_reloading \_rm` initializes normal font, `\catcodetable0` returns to normal catcode table used before `\verbininput` is started, and the text accumulated in `\_vcomments` must be printed by `\_scantextokens` primitive.

verbatim.opm

```
341 \_def\_vcomments{}
342 \_let\_hicomments=\_relax
343
344 \_def\_commentchars#1#2{%
345   \_def\_testcommentchars ##1##2##3\_relax ##4\_iftrue{\_ifnum % not closed in this macro
346     \_ifx #1##1\_ifx#2##21\_fi\_fi 0>0
347     \_ifx\_relax##3\_relax \_addto\_vcomments{\_endgraf}% empty comment=\_enfggraf
348     \_else \_addto\_vcomments{##3 }\_fi}%
349   \_def\_hicomments{\_replfromto{\b\n#1#2}{^~J}{\w{#1#2####1}^~J}}% used in \_hisyntax
350 }
351 \_def\_testcommentchars #1\_iftrue{\_iffalse} % default value of \_testcommentchar
352 \_def\_printcomments{\_ttskip
353   {\_catcodetable0 \_reloading \_rm \_everypar={}%
354     \_noindent \_ignorespaces \_scantextokens\_ea{\_vcomments}\_par}%
355   \_ttskip
356 }
357 \_public \commentchars ;
```

The `\visiblesp` sets spaces as visible characters `□`. It redefines the `\_dsp`, so it is useful for verbatim modes only.

The `\_dsp` is equivalent to `\_□` primitive. It is used in all verbatim environments: spaces are active and defined as `\_dsp` here.

verbatim.opm

```
368 \_def \_visiblesp{\_ifx\_initunifonts\_relax \_def\_dsp{\_char9251 }%
369   \_else \_def\_dsp{\_char32 }\_fi}
370 \_let\_dsp=\_ % primitive "direct space"
371
372 \_public \visiblesp ;
```

## 2.28.2 Listings with syntax highlighting

The user can write

```
\begtt \_hisyntax{C}
...
\_endtt
```

and the code is colored by C syntax. The user can write `\everytt={\_hisyntax{C}}` and all verbatim listings are colored.



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 this:

`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{/#1}^~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\{e{\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 the `\_end` control sequence is appended to the end of the verbatim text. These control sequences are removed in the post-processing phase.

Special macros are 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 the 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 {<pattern>}{<what>}` replaces each `<pattern>` by `<what>`. 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  $\backslash x C\{...\}$ . The patterns inside  $\{...\}$  are not used by next usage of  $\backslash replfromto$  or  $\backslash replthis$  macros.

The  $\backslash xscan$  macro does replacing  $\backslash x$  by  $\backslash z$  in the post-processing phase. The  $\backslash x \langle letter \rangle \langle text \rangle$  expands to  $\backslash xscan \{ \langle letter \rangle \} \langle text \rangle \wedge J$ . If #3 is  $\backslash end$  then it signals that something wrong happens, the  $\langle from \rangle$  was not terminated by legal  $\langle to \rangle$  when  $\backslash replfromto$  did work. We must to fix it by the  $\backslash xscanR$  macro.

hi-syntax.opm

```
63 \def\backslashxscan#1#2\wedge J#3{\ifx\end#3 \ea\backslashxscanR\fi
64 \z{#1}{#2}%
65 \ifx\end#3\else \wedge J\afterfi{\backslashxscan{#1}{#3}\fi}
66 \def\backslashxscanR#1\fi#2\wedge J
```

The  $\backslash hicolor \langle letter \rangle \langle color \rangle$  defines  $\backslash z \langle letter \rangle \langle text \rangle$  as  $\{ \langle color \rangle \langle text \rangle \}$ . It should be used in the context of  $\backslash x \langle letter \rangle \langle text \rangle$  macros.

hi-syntax.opm

```
74 \def\backslashhicolor #1#2{\sdef\z:#1}{#2##1}}
```

The  $\backslash hisyntax \langle name \rangle$  re-defines default  $\backslash prepareverbdata \langle macro \rangle \langle verbtex \rangle$  in order to it does more things: It saves  $\langle verbtex \rangle$  to  $\backslash tmpb$ , appends  $\backslash n$  around spaces and  $\wedge J$  characters in pre-processing phase, it opens  $hisyntax-\langle name \rangle.opm$  file if  $\backslash hisyntax \langle name \rangle$  is not defined. Then  $\backslash the \backslash isyntax \langle name \rangle$  is processed. Finally, the post-processing phase is realized by setting appropriate values to  $\backslash x$  and  $\backslash y$  macros and doing  $\backslash edef \backslash tmpb \{ \backslash tmpb \}$ .

hi-syntax.opm

```
87 \def\backslashhisyntax#1{\def\backslashprepareverbdata##1##2{%
88 \let\n=\relax \let\b=\relax \def\t{\n\noexpand\t\n}\let\start=\relax
89 \adef{ }{\n\noexpand\ } \edef\tmpb{\start\wedge J##2\end}%
90 \replthis{\wedge J}{\n\wedge J\b\n}\replthis{\b\n\end}{\end}%
91 \let\x=\relax \let\y=\relax \let\z=\relax \let\t=\relax
92 \hicomments % keeps comments declared by \commentchars
93 \endlinechar=\wedge M
94 \lowercase{\def\tmpa{#1}}%
95 \ifcsname _hialias:\tmpa\endcsname \edef\tmpa{\cs{hialias:\tmpa}}\fi
96 \ifx\tmpa\empty \else
97 \unless \ifcsname _hisyntax\tmpa\endcsname
98 \isfile{hisyntax-\tmpa.opm} \iftrue \opinput {hisyntax-\tmpa.opm} \fi\fi
99 \ifcsname _hisyntax\tmpa\endcsname
100 \ifcsname hicolors\tmpa\endcsname
101 \cs{hicolors\tmpa}=\cs{hicolors\tmpa}%
102 \else
103 \if\the\hicolors\else
104 \ifcsname hicolors\tmpa\endcsname
105 \global\cs{hicolors\tmpa}=\hicolors \global\hicolors={}%
106 \fi\fi\fi
107 \ea\the \csname _hisyntax\tmpa\endcsname % \the\backslashhisyntax<name>
108 \else\opwarning{Syntax "\tmpa" undeclared (no file hisyntax-\tmpa.opm)}
109 \fi\fi
110 \replthis{\start\n\wedge J}{\replthis{\wedge J\end}{\wedge J}%
111 \def\n{\def\b{\adef{ }{\dsp}}%
112 \bgroup \lccode\-=\ \lowercase{\egroup\def{\noexpand~}}%
113 \def\w#####1{\def\x#####2{\backslashxscan{#####1}#####2\wedge J}%
114 \def\y#####1{\ea \noexpand \csname #####1\endcsname}%
115 \edef\tmpb{\tmpb}%
116 \def\z#####1{\cs{z:#####1}}%
117 \def\t{\hskip \dimexpr\tabspace em/2\relax}%
118 \localcolor
119 }}
120 \public \hisyntax \hicolor ;
```

Aliases for languages can be declared like this. When  $\backslash hisyntax\{xml\}$  is used then this is the same as  $\backslash hisyntax\{html\}$ .

hi-syntax.opm

```
127 \sdef{hialias:xml}{html}
128 \sdef{hialias:json}{c}
```

## 2.29 Graphics

The `\inspic` is defined by `\pdfximage` and `\pdfrefximage` primitives. If you want to use one picture more than once in your document, then the following code is recommended:

```
\newbox\mypic
\setbox\mypic = \hbox{\picw=3cm \inspic{<picture>}}
```

My picture: `\copy\mypic`, again my picture: `\copy\mypic`, etc.

This code downloads the picture data to the PFD output only once (when `\setbox` is processed). Each usage of `\copy\mypic` puts only a pointer to the picture data in the PDF.

If you want to copy the same picture in different sizes, then choose a “basic size” used in `\setbox` and all different sizes can be realized by the `\transformbox{<transformation>}{\copy\mypic}`.

```
3 \_codedecl \inspic {Graphics <2021-03-19>} % preloaded in format
```

graphics.opm

`\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.

```
17 \_def\_inspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inspicB\_inspicA}
18 \_def\_inspicA #1 {\_inspicB {#1}}
19 \_def\_inspicB #1{%
20   \_pdfximage \_ifdim\_picwidth=\_zo \_else width\_picwidth\_fi
21   \_ifdim\_picheight=\_zo \_else height\_picheight\_fi
22   \_picparams {\_the\_picdir#1}%
23   \_pdfrefximage\_pdflastximage\_egroup}
24
25 \_def\_picparams{}
26
27 \_public \inspic ;
```

graphics.opm

Inkscape can 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 after `*.pdf` is included in order to place labels of this picture in the same font as the 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.

```
39 \_def\_inkinspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}
40 \_def\_inkinspicA #1 {\_inkinspicB {#1}}
41 \_def\_inkinspicB #1{%
42   \_ifdim\_picwidth=Opt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
43   \_tmptoks={#1}%
44   \_the\_inkdefs
45   \_opinput {\_the\_picdir #1\_tex}% file with labels
46   \_egroup}
47
48 \_newtoks\_inkdefs \_inkdefs={%
49   \_def\makeatletter#1\makeatother{%
50     \_def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{\_the\_tmptoks}\_hss}%
51     \_def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
52     \_def\put(#1,#2)#3{\_nointerlineskip\_vbox to\_zo{\_vss\_hbox to\_zo{\_kern#1\_picwidth
53       \_pdfsave\_hbox to\_zo{#3}\_pdfrestore\_hss}\_kern#2\_picwidth}}%
54     \_def\begin#1{\_csname \_begin#1\_endcsname}%
55     \_def\_beginpicture(#1,#2){\_vbox\_bgroup
56       \_hbox to\_picwidth{\_kern#2\_picwidth \_def\end##1{\_egroup}}%
57     \_def\_beginatop[#1]#2#3\_end#4{%
58       \_vtop{\_def{\_cr}\_tabiteml{\_tabitemr}\_table{#2}{#3}}%
59     \_def\color[#1]#2{\_scancolor #2,%
60     \_def\_scancolor#1,#2,#3{\_pdfliteral{#1 #2 #3 rg}}%
61     \_def\makebox(#1)[#2]#3{\_hbox to\_zo{\_csname \_mbx:#2\_endcsname{#3}}}%
62     \_sdef{\_mbx:lb}#1#1\_hss{\_sdef{\_mbx:rb}#1#1\_hss}\_sdef{\_mbx:b}#1#1\_hss}\_hss}%
63     \_sdef{\_mbx:lt}#1#1\_hss{\_sdef{\_mbx:rt}#1#1\_hss}\_sdef{\_mbx:t}#1#1\_hss}\_hss}%
64     \_def\rotatebox#1#2{\_pdfrotate{#1}#2}%
```

graphics.opm

```

65 \def\lineheight#1{%
66 \def\setlength#1#2{%
67 }
68 \public \inkinspic ;

```

`\pdfscale{<math>x-scale</math>}{<math>y-scale</math>}` and `\pdfrotate{<math>degrees</math>}` macros are implemented by `\pdfsetmatrix` primitive. We need to know the values of sin, cos function in the `\pdfrotate`. We use Lua code for this.

graphics.opm

```

77 \def\pdfscale#1#2{\pdfsetmatrix{#1 0 0 #2}}
78
79 \def\gonfunc#1#2{%
80 \directlua{tex.print(string.format('\pcent.4f',math.#1(3.14159265*(#2)/180)))}%
81 }
82 \def\sin{\gonfunc{sin}}
83 \def\cos{\gonfunc{cos}}
84
85 \def\pdfrotate#1{\pdfsetmatrix{\cos{#1} \sin{#1} \sin{(#1)-180} \cos{#1}}%
86
87 \public \pdfscale \pdfrotate ;

```

The `\transformbox{<math>transformation</math>}{<math>text</math>}` is copied from OPmac trick 0046.

The `\rotbox{<math>degrees</math>}{<math>text</math>}` 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

```

101 \def\multiplyMxV #1 #2 #3 #4 {% matrix * (vvalX, vvalY)
102 \tmpdim = #1\vvalX \advance\tmpdim by #3\vvalY
103 \vvalY = #4\vvalY \advance\vvalY by #2\vvalX
104 \vvalX = \tmpdim
105 }
106 \def\multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
107 \vvalX=#1pt \vvalY=#2pt \ea\multiplyMxV \currmatrix
108 \edef\tmpb{\ea\ignorept\the\vvalX\space \ea\ignorept\the\vvalY}%
109 \vvalX=#3pt \vvalY=#4pt \ea\multiplyMxV \currmatrix
110 \edef\currmatrix{\tmpb\space
111 \ea\ignorept\the\vvalX\space \ea\ignorept\the\vvalY\space}%
112 }
113 \def\transformbox#1#2{\hbox{\setbox0=\hbox{#2}}%
114 \dimendef\vvalX 11 \dimendef\vvalY 12 % we use these variables
115 \dimendef\newHt 13 \dimendef\newDp 14 % only in this group
116 \dimendef\newLt 15 \dimendef\newRt 16
117 \pretransform{#1}%
118 \kern-\newLt \vrule height\newHt depth\newDp width\zo
119 \setbox0=\hbox{\box0}\ht0=\zo \dp0=\zo
120 \pdfsave#1\rlap{\box0}\pdfrestore \kern\newRt}%
121 }
122 \def\pretransform #1{\def\currmatrix{1 0 0 1}%
123 \def\pdfsetmatrix##1{\edef\tmpb{##1 }\ea\multiplyMxM \tmpb\unskip}%
124 \let\pdfsetmatrix=\pdfsetmatrix #1%
125 \setnewHtDp Opt \ht0 \setnewHtDp Opt -\dp0
126 \setnewHtDp \wd0 \ht0 \setnewHtDp \wd0 -\dp0
127 \protected\def \pdfsetmatrix {\pdfextension setmatrix}%
128 \let\pdfsetmatrix=\pdfsetmatrix
129 }
130 \def\setnewHtDp #1 #2 {%
131 \vvalX=#1\relax \vvalY=#2\relax \ea\multiplyMxV \currmatrix
132 \ifdim\vvalX<\newLt \newLt=\vvalX \fi \ifdim\vvalX>\newRt \newRt=\vvalX \fi
133 \ifdim\vvalY>\newHt \newHt=\vvalY \fi \ifdim-\vvalY>\newDp \newDp=-\vvalY \fi
134 }
135
136 \def\rotbox#1#2{%
137 \ifequal{90}{#1}\iftrue \rotboxA{#1}{\kern\ht0 \tmpdim=\dp0}{\vfill}{#2}%
138 \else \ifequal{-90}{#1}\iftrue \rotboxA{#1}{\kern\dp0 \tmpdim=\ht0}{#2}%
139 \else \transformbox{\pdfrotate{#1}}{#2}%
140 \fi \fi
141 }
142 \def\rotboxA #1#2#3#4{\hbox{\setbox0=\hbox{#4}}#2%

```

```

143 \vbox to\wd0{#3\wd0=\_zo \_dp0=\_zo \_ht0=\_zo
144 \_pdfsave\_pdfrotate{#1}\_box0\_pdfrestore\vfil}%
145 \_kern\_tmpdim
146 }}
147 \_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 a 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

```

166 \_def\_scantwodimens{%
167 \_directlua{tex.print(string.format('\_pcent d){\_pcent d}',
168 token.scan_dimen(),token.scan_dimen()))}%
169 }
170
171 \_def\_puttext{\_ea\_ea\_ea\_puttextA\_scantwodimens}
172 \_def\_puttextA#1#2#3{\_setbox0=\_hbox{#3}\_dimen1=#1sp \_dimen2=#2sp \_puttextB}%
173 \_def\_puttextB{%
174 \_ifvmode
175 \_ifdim\_prevdepth>\_zo \_vskip-\_prevdepth \_relax \_fi
176 \_nointerlineskip
177 \_fi
178 \_wd0=\_zo \_ht0=\_zo \_dp0=\_zo
179 \_vbox to\_zo{\_kern-\_dimen2 \_hbox to\_zo{\_kern\_dimen1 \_box0\_hss}\_vss}}
180
181 \_def\_putpic{\_ea\_ea\_ea\_putpicA\_scantwodimens}
182 \_def\_putpicA#1#2{\_dimen1=#1sp \_dimen2=#2sp \_ea\_ea\_ea\_putpicB\_scantwodimens}
183 \_def\_putpicB#1#2#3{\_setbox0=\_hbox{\_picwidth=#1sp \_picheight=#2sp \_inspic{#3}}\_puttextB}%
184
185 \_newbox\_bgbox
186 \_def\_backgroundpic#1{%
187 \_setbox\_bgbox=\_hbox{\_picwidth=\_pdfpagewidth \_picheight=\_pdfpageheight \_inspic{#1}}%
188 \_pgbackground={\_copy\_bgbox}
189 }
190 \_def\_nospec{0pt}
191 \_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 the 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 the current point back. All these macros are fully expandable and they can be used in the `\pdfliteral` argument.

graphics.opm

```

207 \def\_circle#1#2{\_expr{.5*(#1)} 0 m
208 \_expr{.5*(#1)} \_expr{.276*(#2)} \_expr{.276*(#1)} \_expr{.5*(#2)} 0 \_expr{.5*(#2)} c
209 \_expr{-.276*(#1)} \_expr{.5*(#2)} \_expr{-.5*(#1)} \_expr{.276*(#2)} \_expr{-.5*(#1)} 0 c
210 \_expr{-.5*(#1)} \_expr{-.276*(#2)} \_expr{-.276*(#1)} \_expr{-.5*(#2)} 0 \_expr{-.5*(#2)} c
211 \_expr{.276*(#1)} \_expr{-.5*(#2)} \_expr{.5*(#1)} \_expr{-.276*(#2)} \_expr{.5*(#1)} 0 c h}
212
213 \def\_oval#1#2#3{0 \_expr{-(#3)} m \_expr{#1} \_expr{-(#3)} 1
214 \_expr{(#1)+.552*(#3)} \_expr{-(#3)} \_expr{(#1)+(#3)} \_expr{-.552*(#3)}
215 \_expr{(#1)+(#3)} 0 c
216 \_expr{(#1)+(#3)} \_expr{#2} 1
217 \_expr{(#1)+(#3)} \_expr{(#2)+.552*(#3)} \_expr{(#1)+.552*(#3)} \_expr{(#2)+(#3)}
218 \_expr{#1} \_expr{(#2)+(#3)} c
219 0 \_expr{(#2)+(#3)} 1
220 \_expr{-.552*(#3)} \_expr{(#2)+(#3)} \_expr{-(#3)} \_expr{(#2)+.552*(#3)}
221 \_expr{-(#3)} \_expr{#2} c
222 \_expr{-(#3)} 0 1
223 \_expr{-(#3)} \_expr{-.552*(#3)} \_expr{-.552*(#3)} \_expr{-(#3)} 0 \_expr{-(#3)} c h}
224
225 \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`, `\linewidth`, `\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

```

238 \_newdimen \_linewidth
239 \_def\_fcolor{\_let\_fcolorvalue}
240 \_def\_lcolor{\_let\_lcolorvalue}
241 \_def\_shadow{\_let\_shadowvalue}
242 \_def\_overlapmargins{\_let\_overlapmarginsvalue}
243 \_def\_ratio{\_isnextchar ={\_ratioA}{\_ratioA=}}
244 \_def\_ratioA =#1 {\_def\_ratiovalue{#1}}
245 \_def\_toupvalue#1{\_ifx#1n\_let#1=N\_fi}
246
247 \_def\_setflcolors#1{% use only in a group
248   \_def\_setcolor##1{##1}%
249   \_def\_fillstroke##1##2{##1}%
250   \_edef#1{\_fcolorvalue}%
251   \_def\_fillstroke##1##2{##2}%
252   \_edef#1{#1\_space\_lcolorvalue\_space}%
253 }
254 \_optdef\_inoval[]{\_vbox\_bgroup
255   \_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_linewidth=.5bp
256   \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt
257   \_the\_ovalparams \_relax \_the\_opt \_relax
258   \_toupvalue\_overlapmarginsvalue \_toupvalue\_shadowvalue
259   \_ifx\_overlapmarginsvalue N%
260     \_advance\_hsize by-2\_hhkern \_advance\_hsize by-2\_roundness \_fi
261   \_setbox0=\hbox\_bgroup\_bgroup \_aftergroup\_inovalA \_kern\_hhkern \_let\_next=%
262 }
263 \_def\_inovalA{\_isnextchar\_colorstackpop\_inovalB\_inovalC}
264 \_def\_inovalB#1{#1\_isnextchar\_colorstackpop\_inovalB\_inovalC}
265 \_def\_inovalC{\_egroup % of \setbox0=\hbox\bgroup
266   \_ifdim\_vvkern=\_zo \_else \_ht0=\_dimexpr\_ht0+\_vvkern \_relax
267     \_dp0=\_dimexpr\_dp0+\_vvkern \_relax \_fi
268   \_ifdim\_hhkern=\_zo \_else \_wd0=\_dimexpr\_wd0+\_hhkern \_relax \_fi
269   \_ifx\_overlapmarginsvalue N\_dimen0=\_roundness \_dimen1=\_roundness
270   \_else \_dimen0=-\_hhkern \_dimen1=-\_vvkern \_fi
271   \_setflcolors\_tmp
272   \_hbox{\_kern\_dimen0
273     \_vbox to\_zo{\_kern\_dp0
274       \_ifx\_shadowvalue N\_else
275         \_edef\_tmpb{\_bp{\_wd0+\_linewidth}}{\_bp{\_ht0+\_dp0+\_linewidth}}{\_bp{\_roundness}}}%
276         \_doshadow\_oval
277       \_fi
278       \_pdfliteral{q \_bp{\_linewidth} w \_tmp
279         \_oval{\_bp{\_wd0}}{\_bp{\_ht0+\_dp0}}{\_bp{\_roundness}} B Q}\_vss}%
280       \_ht0=\_dimexpr\_ht0+\_dimen1 \_relax \_dp0=\_dimexpr\_dp0+\_dimen1 \_relax
281       \_box0
282       \_kern\_dimen0}%
283   \_egroup % of \vbox\bgroup
284 }
285 \_optdef\_incircle[]{\_vbox\_bgroup
286   \_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_linewidth=.5bp
287   \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt
288   \_ea\_the \_ea\_circleparams \_space \_relax
289   \_ea\_the \_ea\_opt \_space \_relax
290   \_toupvalue\_overlapmarginsvalue \_toupvalue\_shadowvalue
291   \_setbox0=\hbox\_bgroup\_bgroup \_aftergroup\_incircleA \_kern\_hhkern \_let\_next=%
292 }
293 \_def\_incircleA {\_isnextchar\_colorstackpop\_incircleB\_incircleC}
294 \_def\_incircleB #1{#1\_isnextchar\_colorstackpop\_incircleB\_incircleC}
295 \_def\_incircleC {\_egroup % of \setbox0=\hbox\bgroup
296   \_wd0=\_dimexpr \_wd0+\_hhkern \_relax
297   \_ht0=\_dimexpr \_ht0+\_vvkern \_relax \_dp0=\_dimexpr \_dp0+\_vvkern \_relax
298   \_ifdim \_ratiovalue\_dimexpr \_ht0+\_dp0 > \_wd0

```

```

299     \dimen3=\dimexpr \ht0+\dp0 \relax \dimen2=\ratiovalue\dimen3
300     \else \dimen2=\wd0 \dimen3=\expr{1/\ratiovalue}\dimen2 \fi
301     \setflcolors\tmp
302     \ifx\overlapmarginsvalue N\dimen0=\zo \dimen1=\zo
303     \else \dimen0=-\hhkern \dimen1=-\vvkern \fi
304     \hbox{\kern\dimen0
305         \ifx\shadowvalue N\else
306             \edef\tmpb{\_bp{\dimen2+\lwidth}}{\_bp{\dimen3+\lwidth}}{}}%
307             \doshadow\circlet
308         \fi
309         \pdfliteral{q \_bp{\lwidth} w \tmp \mv{\_bp{.5\wd0}}{\_bp{(\ht0-\dp0)/2}}
310             {\_circle{\_bp{\dimen2}}{\_bp{\dimen3}} B} Q}%
311         \ifdim\dimen1=\zo \else
312             \ht0=\dimexpr \ht0+\dimen1 \relax \dp0=\dimexpr \dp0+\dimen1 \relax \fi
313         \box0
314         \kern\dimen0}
315     \egroup % of \vbox\bgroup
316 }
317 \def\circlet#1#2#3{\_circle{#1}{#2}}
318
319 \public \inoval \incircle \ratio \lwidth \fcolor \lcolor \shadow \overlapmargins ;

```

A shadow effect is implemented here. The shadow is equal to the silhouette of the given path in a 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

```

330 \def\_shadowlevels{9}           % number of layers for blurr effect
331 \def\_shadowdarknessA{0.025}    % transparency of first shadowlevels/2 layers
332 \def\_shadowdarknessB{0.07}    % transparency of second half of layers
333 \def\_shadowmoveto{1.8 -2.5}    % vector defines shifting layer (in bp)
334 \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

```

343 \def\_insertshadowresources{%
344     \global\_addto\_begoutput{\_setshadowresources}%
345     \xdef\_setshadowresources{%
346         \pdfpageresources{ExtGState
347             <<
348             /op1 <</Type /ExtGState /ca \_shadowdarknessA>>
349             /op2 <</Type /ExtGState /ca \_shadowdarknessB>>
350             \morepgresources
351             >>
352         }%
353     }%
354     \global\_let\_insertshadowresources=\_relax
355 }
356 \def\_morepgresources{}

```

The `\_doshadow{<curve>}` does the shadow effect.

graphics.opm

```

362 \def\_doshadow#1{\_vbox{%
363     \_insertshadowresources
364     \tmpnum=\numexpr (\_shadowlevels-1)/2 \relax
365     \edef\tmpfin{\_the\tmpnum}%
366     \ifnum\tmpfin=0 \def\_shadowb{0}\def\_shadowstep{0}%
367     \else \edef\_shadowstep{\_expr{\_shadowb/\tmpfin}}\fi
368     \def\tmpa##1##2##3{\_def\tmpb
369         {#1{##1+2*\_the\tmpnum*\_shadowstep}{##2+2*\_the\tmpnum*\_shadowstep}{##3}}}%
370     \ea \tmpa \tmpb
371     \def\_shadowlayer{%
372         \ifnum\tmpnum=0 /op2 gs \fi
373         \tmpb\_space f
374         \immediateassignment\_advance\tmpnum by-1
375         \ifnum-\tmpfin<\tmpnum
376             \ifx#1\_oval 1 0 0 1 \_shadowstep\_space \_shadowstep\_space cm \fi

```

```

377      \_ea \_shadowlayer \_fi
378  }%
379  \_pdfliteral{q /op1 gs 0 g 1 0 0 1 \_shadowmoveto\_space cm
380    \_ifx#1\_circlet 1 0 0 1 \_expr{\_bp{.5\_wd0}} \_expr{\_bp{(\_ht0-\_dp0)/2}} cm
381    \_else 1 0 0 1 -\_shadowb\_space -\_shadowb\_space cm \_fi
382    \_shadowlayer Q}
383  }}

```

A generic macro `\_clipinpath` $\langle x \rangle \langle y \rangle \langle curve \rangle \langle text \rangle$  declares a clipping path by the  $\langle curve \rangle$  shifted by the  $\langle x \rangle$ ,  $\langle y \rangle$ . The  $\langle text \rangle$  is typeset when such clipping path is active. Dimensions are given by bp without the unit here. The macros `\_clipinoval` $\langle x \rangle \langle y \rangle \langle width \rangle \langle height \rangle \{ \langle text \rangle \}$  and `\_clipincircle` $\langle x \rangle \langle y \rangle \langle width \rangle \langle height \rangle \{ \langle text \rangle \}$  are defined here. These macros read normal  $\text{\TeX}$  dimensions in their parameters.

graphics.opm

```

394 \_def\_clipinpath#1#2#3#4{% #1=x-pos[bp], #2=y-pos[bp], #3=curve, #4=text
395   \_hbox{\_setbox0=\_hbox{\{#4\}}}%
396   \_tmpdim=\_wd0 \_wd0=\_zo
397   \_pdfliteral{q \_mv{#1}{#2}{#3 W n}}%
398   \_box0\_pdfliteral{Q}\_kern\_tmpdim
399 }%
400 }
401
402 \_def\_clipinoval {\_ea\_ea\_ea\_clipinovalA\_scantwodimens}
403 \_def\_clipinovalA #1#2{%
404   \_def\_tmp{\_1/65781.76}{\_2/65781.76}}%
405   \_ea\_ea\_ea\_clipinovalB\_scantwodimens
406 }
407 \_def\_clipinovalB{\_ea\_clipinovalC\_tmp}
408 \_def\_clipinovalC#1#2#3#4{%
409   \_ea\_clipinpath{#1-(#3/131563.52)+(\_bp{\_roundness})}{#2-(#4/131563.52)+(\_bp{\_roundness})}%
410   {\_oval{#3/65781.76-(\_bp{2\_roundness})}{#4/65781.76-(\_bp{2\_roundness})}{\_bp{\_roundness}}}%
411 }
412 \_def\_clipincircle {\_ea\_ea\_ea\_clipincircleA\_scantwodimens}
413 \_def\_clipincircleA #1#2{%
414   \_def\_tmp{\_1/65781.76}{\_2/65781.76}}%
415   \_ea\_ea\_ea\_clipincircleB\_scantwodimens
416 }
417 \_def\_clipincircleB#1#2{%
418   \_ea\_clipinpath\_tmp{\_circle{#1/65781.76}{#2/65781.76}}%
419 }
420 \_public \clipinoval \clipincircle ;

```

## 2.30 The `\table` macro, tables and rules

### 2.30.1 The boundary declarator :

The  $\langle declaration \rangle$  part of `\table{ $\langle declaration \rangle \{ \langle data \rangle \}$`  includes column declarators (letters) and other material: the | or  $\langle cmd \rangle$ . If the boundary declarator : is not used then the boundaries of columns are just before each column declarator with exception of the first one. 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 other places using the boundary declarator : explicitly, for example `{|c:|c(xx):(yy)c}`. The boundary declarator : can be used only once between each pair of column declarators.

Each table item has its group. The  $\langle cmd \rangle$  are parts of the given table item (depending on the boundary declarator position). If you want to apply a special setting for a given column, you can do this by  $\langle setting \rangle$  followed by column declarator. But if the column is not first, you must use :  $\langle setting \rangle$ . 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}{ $\langle data \rangle$ }`

### 2.30.2 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 2.30.3.

By default, `\tabskip` is 0pt. It means that only `\tabiteml`, `\tabitemr` and (`\langle cmd \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 to 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` has individual value. For example `c(\tabskip=5pt):r` gives `\tabskip` value between `c` and `r` columns. You need not use boundary character explicitly, so `c(\tabskip=5pt)r` gives the same result.

Space before the first column is given by the `\tabskip1` and space after the 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.

### 2.30.3 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 to\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 the 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`. The first column starts at the left boundary of this table and it is justified left (to the boundary). The second column ends at the right boundary of the table and it is justified right (to the boundary). The space between them is stretchable and shrinkable to reach the given width `\hsize`.

Example of `\table pxt to\langle size \rangle` (means “paragraphs expanded to”):

```
\table pxt to\hsize { |c|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 the 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 `pxt to` keyword is used. The total sum of (`\langle coefficient \rangle`) must be exactly one. For example,

```
\table pxt to13cm {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.

### 2.30.4 \eqbox: boxes with equal width across the whole document

The `\eqbox [\langle label \rangle]{\langle text \rangle}` behaves like `\hbox{\langle text \rangle}` in the first run of T<sub>E</sub>X. But the widths of all boxes with the same label are saved to `.ref` file and the maximum box width for each label is calculated at the beginning of the next T<sub>E</sub>X run. Then `\eqbox [\langle label \rangle]{\langle text \rangle}` behaves like `\hbox to \langle dim:label \rangle {\hss \langle text \rangle \hss}`, where (`\langle dim:label \rangle`) is the maximum width of all boxes labeled

by the same [*label*]. The documentation of the L<sup>A</sup>T<sub>E</sub>X package `eqparbox` includes more information and tips.

The `\eqboxsize` [*label*]{*dimen*} expands to *dim:label* if this value is known, else it expands to the given *dimen*.

The optional parameter `r` or `l` can be written before [*label*] (for example `\eqbox r[label]{text}`) if you want to put the text to the right or to the left side of the box width.

Try the following example and watch what happens after first T<sub>E</sub>X run and after the second one.

```
\def\leftitem#1{\par
  \noindent \hangindent=\eqboxsize[items]{2em}\hangafter=1
  \eqbox r[items]{#1 }\ignorespaces}

\leftitem {\bf first}      \lorem[1]
\leftitem {\bf second one} \lorem[2]
\leftitem {\bf final}      \lorem[3]
```

## 2.30.5 Implemetation of the `\table` macro and friends

```
3 \_codedecl \table {Basic macros for OpTeX <2021-04-12>} % preloaded in format
```

table.opm

The result of the `\table{<declaration>}{<data>}` 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
```

table.opm

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.

```
24 \_newifi \_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 \_pctottrue}
31 \_public \table ;
```

table.opm

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 processing `{<data>}` by `\_tableB`. The trick `\_tmptoks={<data>}\_edef\_tmpb{\_the\_tmptoks}` is used here in order to keep the hash marks in the *<data>* unchanged.

```
44 \_def\_tableA#1{\_egroup
45   \_the\_thistable \_global\_thistable={}%
46   \_ea\_ifx\_ea\_the\_tabstrut\_setbox\_tstrutbox=\_null
47   \_else \_setbox\_tstrutbox=\_hbox{\_the\_tabstrut}%
48     \_setbox\_tstrutbox=\_hbox{\_vrule width\_zo
49       height\_dimexpr\_ht\_tstrutbox+\_tablinespace
50       depth\_dimexpr\_dp\_tstrutbox+\_tablinespace}%
51   \_offinterlineskip
52   \_lineskip=-2\_tablinespace
53   \_fi
54   \_colnum=0 \_let\_addtabitem=\_addtabitemx
55   \_def\_tmpa{\_tabdata={\_colnum1\_relax}\_scantabdata#1\_relax
56   \_the\_everytable \_bgroup \_catcode`#=#12 \_tableB
57 }
```

table.opm

The `\_tableB` saves *<data>* to `\_tmpb` and does four `\replstrings` to prefix each macro `\crl` (etc.) by `\_crrc`. The reason is: we want to use macros that scan its parameter to the delimiter written in the

right part of the table item declaration. See `\fs` for example. The `\crrc` cannot be hidden in another macro in this 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 `pxto` 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 `pxto` 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 create the 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}
```

The `<data>` are re-tokenized by `\_scantextokens` in order to be more robust to catcode changing inside the `<data>`. But inline verbatim cannot work in special cases here like ``{`` for example.

table.opm

```
98 \_long\_def\_tableB #1{\_egroup \_def\_tmpb{#1}%
99 \_replstring\_tmpb{\crl}{\_crrc\crl}\_replstring\_tmpb{\crl}{\_crrc\crl}}%
100 \_replstring\_tmpb{\crl}{\_crrc\crl}\_replstring\_tmpb{\crl}{\_crrc\crl}}%
101 \_replstring\_tmpb{\crl}{\_crrc\crl}%
102 \_edef\_tabskipmid{\_the\_tabskip}\_tabskip=\_tabskipl
103 \_ifpxto
104 \_tsize=\_hsize \_setbox0 = \_vbox{\_tablepxpreset \_halign \_tableC}%
105 \_tsize=\_dimexpr\_hsize-(\_wd0-\_tmpdim)\_relax
106 \_setbox0=\_null \_halign \_tableC
107 \_else
108 \_halign\_tablew \_tableC
109 \_fi \_egroup
110 }
111 \_def\_tableC{\_ea{\_ea\_bgroup\_the\_tabdata\_egroup\_tabskip=\_tabskipr\_cr
112 \_scantextokens\_ea{\_tmpb\_crrc}}}
113
114 \_def\_tablepxpreset{} % can be used to de-activate references to .ref file
115 \_newbox\_tstrutbox % strut used in table rows
116 \_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||cl|`.

```
tabdata: \_vrule\_the\_tabiteml\_hfil#\_unsskip\_hfil\_the\_tabitemr\_tabstrutA
&\_the\_tabiteml\_hfil#\_unsskip\_the\_tabitemr
\_vrule\_kern\_vbkern\_vrule\_tabstrutA
&\_the\_tabiteml\_hfil#\_unsskip\_hfil\_the\_tabitemr\_tabstrutA
&\_the\_tabiteml#\_unsskip\_hfil\_the\_tabitemr\_vrule\_tabstrutA
ddlinedata: &\_dditem &\_dditem\_vvitem &\_dditem &\_dditem
```

The second result in the `\_ddlinedata` macro is a template of one row of the table used by `\crl` macro.

table.opm

```
136 \_def\_scantabdata#1{\_let\_next=\_scantabdata
137 \_ifx\_relax#1\_let\_next=\_relax
138 \_else\_ifx|#1\_addtabvrule
139 \_else\_ifx(#1\_def\_next{\_scantabdataE}%
140 \_else\_ifx:#1\_def\_next{\_scantabdataF}%
141 \_else\_isinlist{123456789}#1\_iftrue \_def\_next{\_scantabdataC#1}%
142 \_else \_ea\_ifx\_csname\_tabdeclare#1\_endcsname \_relax
143 \_ea\_ifx\_csname\_paramtabdeclare#1\_endcsname \_relax
144 \_opwarning{tab-declarator "#1" unknown, ignored}%
```



```

145         \else
146         \def\next{\ea\scantabdataB\_csname\_paramtabdeclare#1\_endcsname}\_fi
147         \else \def\next{\ea\scantabdataA\_csname\_tabdeclare#1\_endcsname}%
148         \fi\_fi\_fi\_fi\_fi\_fi\_fi \next
149     }
150     \def\scantabdataA#1{\_addtabitem
151         \ea\_addtabdata\ea{#1\_tabstrutA \_tabskip\_tabskipmid\_relax}\_scantabdata}
152     \def\scantabdataB#1#2{\_addtabitem
153         \ea\_addtabdata\ea{#1{#2}\_tabstrutA \_tabskip\_tabskipmid\_relax}\_scantabdata}
154     \def\scantabdataC {\_def\_tmpb{\_afterassignment\_scantabdataD \_tmpnum=}
155     \def\_scantabdataD#1{\_loop \ifnum\_tmpnum>0 \advance\_tmpnum by-1 \_addto\_tmpb{#1}\_repeat
156         \ea\_scantabdata\_tmpb}
157     \def\_scantabdataE#1){\_addtabdata{#1}\_scantabdata}
158     \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.

table.opm

```

178 \_newcount\_colnum      % number of current column in the table
179 \_public \colnum ;
180
181 \_def\_addtabitemx{\_ifnum\_colnum>0
182     \_addtabdata{\_egroup &\_bgroup}\_addto\_ddlinedata{&\_dditem}\_fi
183     \_advance\_colnum by1 \_let\_tmpa=\_relax
184     \_ifnum\_colnum>1 \_ea\_addtabdata\ea{\_ea\_colnum\_the\_colnum\_relax}\_fi}
185 \_def\_addtabdata#1{\_tabdata\ea{\_the\_tabdata#1}}

```

This code converts `||` or `|` from *<table declaration>* to the *<converted declaration>*.

table.opm

```

191 \_def\_addtabvrule{%
192     \_ifx\_tmpa\_vrule \_addtabdata{\_kern\_vvkern}%
193     \_ifnum\_colnum=0 \_addto\_vvleft{\_vvitem}\_else\_addto\_ddlinedata{\_vvitem}\_fi
194     \_else \_ifnum\_colnum=0 \_addto\_vvleft{\_vvitemA}\_else\_addto\_ddlinedata{\_vvitemA}\_fi\_fi
195     \_let\_tmpa=\_vrule \_addtabdata{\_vrule}%
196 }
197 \_def\_tabstrutA{\_copy\_tstrutbox}
198 \_def\_vvleft{}
199 \_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

```

211 \_def\_tabdeclarec{\_the\_tabiteml\_hfil##\_unsskip\_hfil\_the\_tabitemr}
212 \_def\_tabdeclarel{\_the\_tabiteml\_relax##\_unsskip\_hfil\_the\_tabitemr}
213 \_def\_tabdeclarer{\_the\_tabiteml\_hfil##\_unsskip\_the\_tabitemr}
214 \_def\_paramtabdeclarep#1{\_the\_tabiteml
215     \_vtop{\_hsize=#1\_relax \_baselineskip=\_normalbaselineskip
216     \_lineskiplimit=\_zo \_noindent \_hskip\_zo \_relax ##\_unsskip
217     \_ifvmode\_vskip\_dp\_tstrutbox \_else\_lower\_dp\_tstrutbox\_hbox{}\_fi}\_the\_tabitemr}

```

Users put optional spaces around the table item typically, i.e. they write `& text &` instead of `&text&`. The left space is ignored by the internal T<sub>E</sub>X 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 T<sub>E</sub>X `\hideskip` macro: `&\hideskip text\hideskip&`.

```
228 \def\unsskip{\ifmmode\else\ifdim\lastskip>\zo \unskip\fi\fi}
```

The `\fL`, `\fR`, `\fC` and `\fX` macros only do 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.

```
237 \let\fL=\raggedright
238 \def\fR{\leftskip=0pt plus 1fill \relax}
239 \def\fC{\leftskip=0pt plus 1fill \rightskip=0pt plus 1fill \relax}
240 \def\fX{\leftskip=0pt plus 1fil \rightskip=0pt plus 1fil \parfillskip=0pt plus 2fil \relax}
241 \long\def\fS #1\unsskip{\noindent \setbox0=\vbox{\noindent #1\endgraf \ea}%
242   \ifnum\prevgraf=1 \hfil \fi #1\unsskip
243 }
244 \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 to suppress the negative `\lineskip` declared by `\tablinespace`.

```
254 \def\crl{\crrc\noalign{\hrule}}
255 \def\crl1{\crrc\noalign{\hrule\kern\hhkern\hrule}}
256 \def\zerotabrule {\noalign{\hrule height\zo width\zo depth\zo}}
257
258 \def\crli{\crrc \zerotabrule \omit
259   \gdef\dditem{\omit\tablinefil}\gdef\vvitem{\kern\vvkern\vrule}\gdef\vvitemA{\vrule}%
260   \vvleft\tablinefil\ddlinedata\crrc \zerotabrule}
261 \def\crl1i{\crli\noalign{\kern\hhkern}\crli}
262 \def\tablinefil{\leaders\hrule\hfil}
263
264 \def\crlp#1{\crrc \zerotabrule \noalign{\kern-\drulewidth}%
265   \omit \xdef\crlplist{#1}\xdef\crlplist{\ea\ea\crlpA\crlplist,\end,%
266   \global\tmpnum=0 \gdef\dditem{\omit\crlpD}%
267   \gdef\vvitem{\kern\vvkern\drulewidth}\gdef\vvitemA{\kern\drulewidth}%
268   \vvleft\crlpD\ddlinedata \global\tmpnum=0 \crrc \zerotabrule}
269 \def\crlpA#1,{\ifx\end#1\else \crlpB#1-\end,\ea\crlpA\fi}
270 \def\crlpB#1#2-#3,{\ifx\end#3\xdef\crlplist{\crlplist#1#2,}\else\crlpC#1#2-#3,\fi}
271 \def\crlpC#1-#2-#3,{\tmpnum=#1\relax
272   \loop \xdef\crlplist{\crlplist\the\tmpnum,}\ifnum\tmpnum<#2\advance\tmpnum by 1 \repeat}
273 \def\crlpD{\incr\tmpnum \edef\tmpa{\noexpand\isinlist\noexpand\crlplist,\the\tmpnum,}}%
274   \tmpa\iftrue \kern-\drulewidth \tablinefil \kern-\drulewidth\else\hfil \fi}
275
276 \def\tskip{\afterassignment\tskipA \tmpdim}
277 \def\tskipA{\gdef\dditem{}\gdef\vvitem{}\gdef\vvitemA{}\gdef\tabstrutA{}}%
278   \vbox to\tmpdim{\ddlinedata \crrc
279   \zerotabrule \noalign{\gdef\tabstrutA{\copy\tstrutbox}}}
280
281 \public \crl \crl1 \crli \crl1i \crlp \tskip ;
```

The `\mspan{<number>}[<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.

```
289 \def\mspan{\omit \afterassignment\mspanA \mscount=}
290 \def\mspanA[#1]#2{\loop \ifnum\mscount>1 \cs{span}\omit \advance\mscount-1 \repeat
291   \count1=\colnum \colnum=0 \def\tmpa{\tabdata={}\scantabdata#1\relax
292   \colnum=\count1 \setbox0=\vbox{\halign\ea{\ea\bgroup\the\tabdata\egroup\cr#2\cr}%
293   \global\setbox8=\lastbox}%
294   \setbox0=\hbox{\unhbox8 \unskip \global\setbox8=\lastbox}%
295   \unhbox8 \ignorespaces}
296 \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 a one-digit number. If you want to set more digits then use braces.

```
308 \def\vspan#1#2#{\vspanA{#1#2}}
309 \def\vspanA#1#2{\vtop to\zo{\hbox{\lower \dimexpr
310   #1\dimexpr(\ht\tstrutbox+\dp\tstrutbox)/2\relax
311   -\dimexpr(\ht\tstrutbox+\dp\tstrutbox)/2\relax \hbox{#2}}\vss}}
312 \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`.

table.opm

```
323 \_newdimen\_drulewidth \_drulewidth=0.4pt
324 \_let\_orihrule=\hrule \_let\_orivrule=\vrule
325 \_def\_rulewidth{\_afterassignment\_rulewidthA \_drulewidth}
326 \_def\_rulewidthA{\_edef\_hrule{\_orihrule height\_drulewidth}%
327 \_edef\_vrule{\_orivrule width\_drulewidth}%
328 \_let\_rulewidth=\_drulewidth
329 \_public \vrule \hrule \rulewidth;}
330 \_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.

table.opm

```
340 \_long\_def\_frame#1{%
341 \_hbox{\_vrule\_vtop{\_vbox{\_hrule\_kern\_vbkern
342 \_hbox{\_kern\_hhkern\_relax#1\_kern\_hhkern}%
343 }\_kern\_vbkern\_hrule}\_vrule}}
344 \_public \frame ;
```

`\eqbox` and `\eqboxsize` are implemented here. The widths of all `\eqboxes` are saved to the `.ref` file in the format `\_Xeqlbox{<label>}{<size>}`. The `.ref` file is read again and maximum box width for each `<label>` is saved to `\_eqb:<label>`.

table.opm

```
353 \_def\_Xeqlbox#1#2{%
354 \_ifcsname\_eqb:#1\_endcsname
355 \_ifdim #2>\_cs{\_eqb:#1}\_relax \_sdef{\_eqb:#1}{#2}\_fi
356 \_else \_sdef{\_eqb:#1}{#2}\_fi
357 }
358 \_def\_eqbox #1[#2]#3{\_setbox0=\_hbox{#3}%
359 \_openref \_immediate\_wref \_Xeqlbox{#2}{\_the\_wd0}}%
360 \_ifcsname\_eqb:#2\_endcsname
361 \_hbox to\_cs{\_eqb:#2}{\_ifx r#1\_hfill\_fi\_hss\_unhbox0\_hss\_ifx l#1\_hfill\_fi}%
362 \_else \_box0 \_fi
363 }
364 \_def\_eqboxsize [#1]#2{\_trycs{\_eqb:#1}{#2}}
365
366 \_public \eqbox \eqboxsize ;
```

## 2.31 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 `\pagtotal` and does `\vsplit` of `\vbox` 6 to columns with a height of such free space. This is done only if we have enough amount of material in `\vbox` 6 to fill the full page by columns. This is repeated in a loop until we have less amount of material in `\vbox` 6. Then we run `\_balancecolumns` which balances the last part of the columns. Each part of printed material is distributed to the 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\beginmulti #1 {\par\begin\wipetpar\multiskip\penalty0 \def\Ncols{#1}
29 \setbox6=\vbox\begin \let\setxsize=\relax \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=\divide\dimen0 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{\begin % 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{\begin \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 ;

```

## 2.32 Citations, bibliography

### 2.32.1 Macros for citations and bibliography preloaded in the format

```

3 \_codedecl \_cite {Cite, Bibliography <2021-04-13>} % loaded in format

```

cite-bib.opm

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.

```

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-bib.opm

`\_bibp` expands to `\bibpart/`. By default, `\bibpart` is empty, so internal links are in the form `cite:/<number>`. If `\bibpart` is set to `<bibpart>`, then internal links are `cite:<bibpart>/<number>`.

```

23 \_def\_bibp{\_the\_bibpart/} % unique name for each bibliography list

```

cite-bib.opm

`\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 a 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

```

43 \_def\_cite[#1]{\_citeA#1,,,\_printsavedcites}}
44 \_def\_nocite[#1]{\_citeA#1,,,\_printsavedcites}}
45 \_def\_rcite[#1]{\_citeA#1,,,\_printsavedcites}}
46 \_def\_ecite[#1]{\_bgroup\_citeA#1,,,\_ea\_eciteB\_savedcites;}
47 \_def\_eciteB#1,#2,#3{\_if?#1\_relax #3\_else \_ilink[cite:\_bibp#1]{#3}\_fi\_egroup}
48 \_def\_savedcites{}
49
50 \_public \_cite \_nocite \_rcite \_ecite ;

```

`<bib-marks>` may be numbers or a special text related to cited bib-entry. It depends on `\nonumcitations` and on used bib-style. The mapping from `<label>` to `<bib-mark>` is done when `\bib` or `\usebib` is processed. These macros store the information to `\_Xbib{<bibpart>}{<label>}{<number>}{<nonumber>}` where `<number>` and `<nonumber>` are two variants of `<bib-mark>` (numbered or text-like). This information is read from `.ref` file and it is saved to macros `\_bib:<bibpart>/<label>` and `\_bim:<bibpart>/<number>`. First one includes `<number>` and second one includes `<nonumber>`. The `\_lastbn:<bibpart>` macro includes last number of bib-entry used in the document with given `<bibpart>`. A designer can use it to set appropriate indentation when printing the list of all bib-entries.

cite-bib.opm

```

69 \_def\_Xbib#1#2#3#4{\_sxdef\_bib:#1/#2}{\_bibbn{#3}&}%
70 \_if~#4~\_else\_sxdef\_bim:#1/#3}{#4}\_fi\_sxdef\_lastbn:#1}{#3}

```

`\_citeA <label>`, processes one label from the list of labels given in the parameter of `\cite`, `\nocite`, `\rcite` or `\ecite` macros. It adds the `<label>` to a global list `\_ctlst:<bibpart>/` which will be used by `\usebib` (it must know what `<labels>` are used in the document to pick-up only relevant bib-entries from the database. Because we want to save space and to avoid duplications of `<label>` in the `\_ctlst:<bibpart>/`, we distinguish four cases:

- $\langle label \rangle$  was not declared by  $\backslash\_Xbib$  before and it is first such a  $\langle label \rangle$  in the document: Then  $\backslash\_bib:\langle bibpart \rangle/\langle label \rangle$  is undefined and we save label using  $\backslash\_addcitelist$ , write warning on the terminal and define  $\backslash\_bib:\langle bibpart \rangle/\langle label \rangle$  as empty.
- $\langle label \rangle$  was not declared by  $\backslash\_Xbib$  before but it was used previously in the document: Then  $\backslash\_bib:\langle bibpart \rangle/\langle label \rangle$  is empty and we do nothing (only data to  $\backslash\_savedcites$  are saved).
- $\langle label \rangle$  was declared by  $\backslash\_Xbib$  before and it is first such  $\langle label \rangle$  used in the document: Then  $\backslash\_bib:\langle bibpart \rangle/\langle label \rangle$  includes  $\backslash\_bibnn\{\langle number \rangle\}\&$  and we test this case by the command  $\backslash if \&\backslash\_bibnn\{\langle number \rangle\}\&$ . This is true when  $\backslash\_bibnn\{\langle number \rangle\}$  expands to empty. The  $\langle label \rangle$  is saved by  $\backslash\_addcitelist$  and  $\backslash\_bib:\langle bibpart \rangle/\langle label \rangle$  is re-defined directly as  $\langle number \rangle$ .
- $\langle label \rangle$  was declared by  $\backslash\_Xbib$  and it was used previously in the document. Then we do nothing (only data to  $\backslash\_savedcites$  are saved).

The  $\backslash\_citeA$  macro runs repeatedly over the whole list of  $\langle labels \rangle$ .

```

99 \def\citeA #1#2,{\if#1,\else
100   \if *#1\addcitelist{*}\ea\skiptorelax \fi
101   \ifcsname _bib:\bibp#1#2\endcsname \else
102     \addcitelist{#1#2}%
103     \opwarning{\the\bibpart} \noexpand\cite [#1#2] unknown. Try to TeX me again\openref
104     \incr\unresolvedrefs
105     \addto\savedcites{?,}\def\sortcitesA{\lastcitenum=0
106     \ea\gdef \csname _bib:\bibp#1#2\endcsname {}%
107     \ea\skiptorelax \fi
108   \ea\ifx \csname _bib:\bibp#1#2\endcsname \empty
109     \addto\savedcites{?,}\def\sortcitesA{\lastcitenum=0
110     \ea\skiptorelax \fi
111   \def\bibnn##1{}%
112   \if &\csname _bib:\bibp#1#2\endcsname
113     \def\bibnn##1#2{##1}%
114     \addcitelist{#1#2}%
115     \sxdef{\bib:\bibp#1#2}{\csname _bib:\bibp#1#2\endcsname}%
116   \fi
117   \edef\savedcites{\savedcites \csname _bib:\bibp#1#2\endcsname,}%
118   \relax
119   \ea\citeA\fi
120 }
121 \let\bibnn=\relax
```

Because we implement possibility of more independent bibliography lists distinguished by  $\langle bibpart \rangle$ , the  $\backslash\_addcitelist\{\langle label \rangle\}$  macro must add the  $\langle label \rangle$  to given  $\backslash\_ctlst:\langle bibpart \rangle/$ .

When  $\backslash\_addcitelist$  is processed before  $\backslash\_usebib$ , then  $\backslash\_citeI[\langle label \rangle]$  is added.  $\backslash\_usebib$  will use this list for selecting right records from .bib file. Then  $\backslash\_usebib$  sets  $\backslash\_ctlst:\langle bibpart \rangle/$  to  $\backslash\_write$ . If  $\backslash\_addcitelist$  is processed after  $\backslash\_usebib$ , then  $\backslash\_Xcite\{\langle bibpart \rangle/\}\{\langle label \rangle\}$  is saved to the .ref file. The  $\backslash\_Xcite$  creates  $\backslash\_ctlstB:\langle bibpart \rangle/$  as a list of saved  $\backslash\_citeI[\langle label \rangle]$ . Finally,  $\backslash\_usebib$  concatenates both lists  $\backslash\_ctlst:\langle bibpart \rangle/$  and  $\backslash\_ctlstB:\langle bibpart \rangle/$  in the second  $\text{\TeX}$  run.

```

138 \def\addcitelist#1{%
139   \unless \ifcsname _ctlst:\bibp\endcsname \sxdef{\ctlst:\bibp}{}\fi
140   \ea \ifx \csname _ctlst:\bibp\endcsname \write
141     \openref \immediate\wref\Xcite{\bibp}{#1}%
142   \else \global \ea\addto \csname _ctlst:\bibp\endcsname {\citeI[#1]}\fi
143 }
144 \def\Xcite#1#2{%
145   \unless \ifcsname _ctlstB:#1\endcsname \sxdef{\ctlstB:#1}{}\fi
146   \global \ea\addto \csname _ctlstB:#1\endcsname {\citeI[#2]}%
147 }
```

The  $\langle bib\text{-}marks \rangle$  (in numeric or text form) are saved in  $\backslash\_savedcites$  macro separated by commas. The  $\backslash\_printsavedcites$  prints them by normal order or sorted if  $\backslash\_sortcitations$  is specified or condensed if  $\backslash\_shordcitations$  is specified.

The  $\backslash\_sortcitations$  appends the dummy number 300000 and we suppose that normal numbers of bib-entries are less than this constant. This constant is removed after the sorting algorithm. The  $\backslash\_shordcitations$  sets simply  $\backslash\_lastcitenum=1$ . 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).



```

163 \def\printsavedcites{\sortcitesA
164 \chardef\tmpb=0 \ea\_citeB\_savedcites,%
165 \ifnum\tmpb>0 \printdashcite{\the\tmpb}\fi
166 }
167 \def\sortcitesA{}
168 \def\sortcitations{%
169 \def\sortcitesA{\edef\savedcites{300000,\ea\_ea\_sortcitesB\_savedcites,%
170 \def\tmpa###1300000,{\def\savedcites{###1}}\ea\_tmpa\_savedcites}%
171 }
172 \def\sortcitesB #1,{\if $1$%
173 \else
174 \mathchardef\tmpa=#1
175 \edef\savedcites{\ea\_ea\_sortcitesC \savedcites\_end
176 \ea\_sortcitesB
177 \fi
178 }
179 \def\sortcitesC#1,{\ifnum\tmpa<#1\edef\tmpa{\the\tmpa,#1}\ea\_sortcitesD
180 \else\edef\savedcites{\savedcites#1,}\ea\_sortcitesC\_fi}
181 \def\sortcitesD#1\_end{\edef\savedcites{\savedcites\_tmpa,#1}}
182
183 \def\_citeB#1,{\if$1$\_else
184 \if?#1\_relax??%
185 \else
186 \ifnum\_lastcitenum=0 % only comma separated list
187 \printcite{#1}%
188 \else
189 \ifx\_citesep\_empty % first cite item
190 \lastcitenum=#1\_relax
191 \printcite{#1}%
192 \else % next cite item
193 \advance\_lastcitenum by1
194 \ifnum\_lastcitenum=#1\_relax % cosecutive cite item
195 \mathchardef\tmpb=\_lastcitenum
196 \else % there is a gap between cite items
197 \lastcitenum=#1\_relax
198 \ifnum\tmpb=0 % previous items were printed
199 \printcite{#1}%
200 \else
201 \printdashcite{\the\tmpb}\printcite{#1}\chardef\tmpb=0
202 \fi\_fi\_fi\_fi\_fi
203 \ea\_citeB\_fi
204 }
205 \def\_shortcitations{\_lastcitenum=1 }
206
207 \def\_printcite#1{\_citesep
208 \_ilink[cite:\_bibp#1]{\_citelinkA{#1}}\_def\_citesep{,\_hskip.2em\_relax}}
209 \def\_printdashcite#1{\_ifmode-\_else\_hbox{--}\_fi\_ilink[cite:\_bibp#1]{\_citelinkA{#1}}}
210 \def\_citesep{}
211
212 \def\_nonumcitations{\_lastcitenum=0\_def\_sortcitesA{\_def\_etalchar##1{$~{##1}$}%
213 \_def\_citelinkA##1{\_trycs\_bim:\_bibp##1}
214 {##1\_opwarning{\_noexpand\nonumcitations + empty bibmark. Maybe bad bib-style}}}%
215 }
216 \def\_citelinkA{}
217
218 \_public \nonumcitations \sortcitations \shortcitations ;

```

The `\bib` [*label*] or `\bib` [*label*] =*{(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{<bibpart>}{<label>}{<number>}{<nonumber>}` into `.ref` file immediately using `\_wbib{<label>}{<number>}{<nonumber>}`. This is the core of creation of mapping from *labels* to *number* and *nonumber*.

`\_bibA` and `\_bibB` implement the scanner of the optional argument with the `\bibmark`.

`\_bibgl` is `\relax` by default but `\slides` do `\let\_bibgl=\_global`.

`\_dbib{<label>}` creates destination for hyperlinks.

```

234 \def\_bib[#1]{\_def\_tmp{\_isnextchar={\_bibA[#1]}{\_bibmark={}\_bibB[#1]}}%
235 \ea\_tmp\_romannumeral-`.} % ignore optional space

```

```

236 \_def\_bibA[#1]=#2{\_bibmark={#2}\_bibB[#1]}
237 \_def\_bibB[#1]{\_par \_bbskip
238 \_bibgl\_advance\_bibnum by1
239 \_noindent \_def\_tmpb{#1}\_dbib{#1}\_wbib{#1}{\_the\_bibnum}{\_the\_bibmark}%
240 \_printbib \_ignorespaces
241 }
242 \_def\_dbib#1{\_dest[cite:\_bibp\_the\_bibnum]\_printlabel{#1}}
243 \_def\_wbib#1#2#3{%
244 \_ifx\_wref\_wrefrelax\_else \_immediate\_wref\_Xbib{\_the\_bibpart}{#1}{#2}{#3}\_fi
245 \_unless \_ifcscname bib:\_bibp#1\_endcscname \_Xbib{\_the\_bibpart}{#1}{#2}{#3}\_fi
246 }
247 \_let\_bibgl=\_relax
248
249 \_public \_bib ;

```

The `\_printbib` prints the bib-entry itself. You can re-define it if you want a different design. The `\_prtibib` 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 `\_bbskip` macro.

```

266 \_def \_printbib {\_hangindent=\_iindent
267 \_ifx\_citelinkA\_empty \_hskip\_iindent \_llap{[\_the\_bibnum]}\_fi
268 }
269 \_def \_bbskip {\_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 used first. The `usebib.opm` file loads the `librarian.tex` for scanning the `.bib` files. See the section 2.32.2, where the file `usebib.opm` is documented.

```

279 \_def\_usebib{\_par \_opinput {usebib.opm} \_usebib}
280 \_def\_usebib{\_usebib}

```

`cite-bib.opm`

`\_nobibwarning` [*list of bib-labels*] declares a list of bib labels which are not fully declared in `.bib` file but we want to suppress the warning about it. List of bib labels are comma-separated case sensitive list without spaces.

```

290 \_def\_nobibwarnlist{,}
291 \_def\_nobibwarning[#1]{\_global\_addto\_nobibwarnlist{#1},}
292 \_public \_nobibwarning ;

```

`cite-bib.opm`

## 2.32.2 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 bibliographic references (only those declared by `\_cite[]` or `\_nocite[]` in the text). The formatting of such references is defined in the style file.

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 exceptions in your `local.bib` file for your document.

The `bib-⟨style⟩.opm` declares entry types (like `@BOOK`, `@ARTICLE`) and declares their mandatory and optional fields (like `author`, `title`). When a mandatory field is missing in an entry in the `.bib` file then a warning is printed on the terminal about it. You can suppress such warnings by command `\_nobibwarning` [*list of bib-labels*], where `⟨bib-labels⟩` is a comma-separated list of labels (without spaces) where missing mandatory fields will be no warned.

Old `.bib` files may use the obscure notation for accents like `{\_o}`. Recommendation: convert such old files to Unicode encoding. If you are unable to do this then you can set `\_bibtexhook={\_oldaccents}`.

### 2.32.3 Notes for bib-style writers

The `.bib` files include records in the format:

```
@<entry-type>{<label>,  
  <field-name> = "<field-data>",  
  <field-name> = "<field-data>",  
  ...etc  
}
```

see the file `demo/op-biblist.bib` for a real example. The `<entry-types>` and `<field-names>` are case insensitive.

Ancient BibTeX has read such files and has generated files appropriate for reading by L<sup>A</sup>T<sub>E</sub>X. It has worked with a set of `<entry-types>`, see the [www](http://en.wikipedia.org/wiki/BibTeX) page <http://en.wikipedia.org/wiki/BibTeX>. The set of entry types listed on this [www](http://en.wikipedia.org/wiki/BibTeX) page is de facto the BibTeX standard. The OpTeX bib style writer must “declare” all such entry types and more non-standard entry types can be declared too if there is a good reason for doing it. The word “declare” used in the previous sentence means that a bib-style writer must define the printing rules for each `<entry-type>`. The printing rules for `<entry-type>` include: which fields will be printed, in what order, by what format they will be printed on (italic, caps, etc.), which fields are mandatory, which are optional, and which are ignored in `.bib` records.

The style writer can be inspired by two styles already done: `bib-simple.opm` and `bib-iso690.opm`. The second one is documented in detail in section 2.32.5.

The printing rules for each `<entry-type>` must be declared by `\sdef{<_print:<entry-type>}` in `bib-<style>.opm` file. The `<entry-type>` has to be lowercase here. OpTeX supports following macros for a more comfortable setting of printing rules:

- `\_bprinta` [`<field-name>`] {`<if defined>`} {`<if not defined>`}. The part `<if defined>` is executed if `<field-name>` is declared in `.bib` file for the entry which is currently processed. Else the part `<if not defined>` is processed. The part `<if defined>` can include the `*` parameter which is replaced by the value of the `<field-name>`.
- The part `<if not defined>` can include the `\_bibwarning` command if the `<field-name>` is mandatory.
- `\_bprintb` [`<field-name>`] {`<if defined>`} {`<if not defined>`}. 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 the `####1` parameter for internal `\_bprintb`. But if `\_bprinta` commands are nested then the parameter is not duplicated.
- `\_bprintc` `\macro` {`<if non-empty>`}. The `<if non-empty>` part is executed if `\macro` is non-empty. The `*` parameter can be used, it is replaced by the `\macro`.
- `\_bprintv` [`<field1>`,`<field2>`,...] {`<if defined>`} {`<if not defined>`}. The part `<if defined>` is executed if `<field1>` or `<field2>` or ... is defined, else the second part `<if not defined>` is executed. There is one filed name or the list field names separated by commas. The parts cannot include any parameters.

There are two special field-names: `!author` and `!editor`. The processed list of authors or editors are printed here instead of raw data, see the commands `\_authorname` and `\_editorname` below.

The bib-style writer can define `_print:BEGIN` and/or `_print:END`. They are executed at the beginning or end of each `<entry-type>`. 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).

The `\_bibmark={something}` can be declared, for instance in the `_print:END` macro. Such “bibmark” is saved to the `.ref` file and used in next T<sub>E</sub>X run as `\cite` marks when `\nonumcitations` is set.

Moreover, the bib-style writer must declare the format of special fields `author` and `editor`. These fields include a list of names, each name is precessed individually in a loop. The `\_authorname` or `\_editorname` is called for each name on the list. The bib-style writer must define the `\_authorname` and `\_editorname` commands in order to declare the format of printing each individual name. The following control sequences can be used in these macros:

- `\_NameCount`: the number of the currently processed author in the list
- `\_namecont`: the total number of the authors in the list
- `\_Lastname`, `\_Firstname`, `\_Von`, `\_Junior`: the parts of the name.

The whole style file is read in the group during the `\usebib` command is executed before typesetting the reference list. Each definition or setting is local here.

The auto-generated phrases (dependent on current language) can be used in bib-style files by `\_mtext{bib}.\langle identifier \rangle`, where  $\langle identifier \rangle$  is an identifier of the phrase and the phrase itself is defined by `\_sdef{\_mt: bib.\langle identifier \rangle:\langle language \rangle}{\langle phrase \rangle}`. See section 2.37.3 for more detail. Phrases for  $\langle identifiers \rangle$ : and, etal, edition, citedate, volume, number, prepages, postpages, editor, editors, available, availablealso, bachthesis, mastthesis, phdthesis are defined already, see the end of section 2.37.3.

If you are using non-standard field-names in .bib database and bib-style, you have to declare them by `\_CreateField {\langle fieldname \rangle}`.

You can declare `\_SortingOrder` in the manner documented by librarian package.

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 auto-citations before other citations, then you can mark your entries in .bib file by `sortedby = "@"`, because this character is sorted before A.

## 2.32.4 The usebib.opm macro file loaded when \usebib is used

```
3 \_codedecl \MakeReference {Reading bib databases <2021-04-13>} % loaded on demand by \usebib
```

Loading the librarian.tex macro package. See texdoc librarian for more information about it.

We want to ignore `\errmessage` and we want not to create `\jobname.lbr` file.

```
13 \_def\errmessage#1{}
14 \_def\newwrite#1{\_csname lb@restoreat\_endcsname \_endinput}
15 \_def\_tmpb{\\_catcode\_ =12 \_input librarian \_catcode\_ =11 }\_tmpb
16 \_let\errmessage=\_errmessage
17 \_let\newwrite=\_newwrite
18
19 \_private \BibFile \ReadList \SortList \SortingOrder \NameCount \AbbreviateFirstname
20 \CreateField \RetrieveFieldInFor \RetrieveFieldIn ;
```

The `\usebib` command.

```
26 \_def\_usebib/#1 (#2) #3 {%
27 \_let\_citeI=\_relax \_xdef\_citelist{\\_trycs{\_ctlst:\_bibp}{}}\_trycs{\_ctlstB:\_bibp}{}}%
28 \_global \_ea\_let \_csname \_ctlst:\_bibp\_endcsname =\_write
29 \_ifx\_citelist\_empty
30 \_opwarning{No cited items. \_noexpand\usebib ignored}%
31 \_else
32 \_bgroup \_par
33 \_emergencystretch=.3\_hsize
34 \_def\_optexbibstyle{#2}%
35 \_setctable\_optexcatcodes
36 \_ea \_skiptoendinput \_input languages.opm
37 \_input bib-#2.opm
38 \_the \_bibtexhook
39 \_ifcsname \_mt: bib.and:\_cs{\_lan:\_the\_language}\_endcsname \_else
40 \_opwarning{\_string\usebib: No phrases for language
41 "\_cs{\_lan:\_the\_language}" (using "en")}%
42 \_language=0 \_chardef\_documentlanguage=0
43 \_fi
44 \_def\_tmp##1[*]##2\_relax{\_def\_tmp{##2}}\_ea\_tmp\_citelist[*]\_relax
45 \_ifx\_tmp\_empty\_else % there was \nocite[*] used.
```

```

46      \setbox0=\_vbox{\_hsize=\_maxdimen \_def\_citelist{}\_adef@{\_readbibentry}%
47      \_input #3.bib
48      \_ea}\_ea\_def\_ea\_citelist\_ea{\_citelist}%
49      \_fi
50      \_def\_citeI[#1]{\_csname lb@cite\_endcsname{##1}{\_bibp}{}}\_citelist
51      \_BibFile{#3}%
52      \_if s#1\_SortList{\_bibp}\_fi
53      \_ReadList{\_bibp}%
54      \_restorectable
55      \_egroup
56      \_fi
57 }
58 \_long\_def\_skiptoendinput#1\_endinput{}
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.

usebib.opm

```

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}}%
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\_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 bocks new reading of .bib
112 \_catcode\@=\_tmpnum

```

Main action per each entry.

usebib.opm

```

118 \_def\_MakeReference{\_par \_bibskip
119 \_bibgl\_advance\_bibnum by1

```

```

120 \_isdefined{\_bim:\_bibp\_the\_bibnum}\_iftrue
121   \edef\_tmpb{\_csname\_bim:\_bibp\_the\_bibnum\_endcsname}%
122   \_bibmark=\_ea{\_tmpb}%
123 \_else \_bibmark={}\_fi
124 \_edef\_tmpb{\_EntryKey}%
125 \_noindent \_dbib\_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     \_wbib \_EntryKey {\_the\_bibnum}{\_the\_bibmark}%
139   }\_par
140 }

```

The `\_bprinta`, `\_bprintb`, `\_bprintc`, `\_bprintv` commands used in the style files:

usebib.opm

```

147 \_def\_bprinta {\_bprintb*}
148 \_def\_bprintb #1[#2#3]{%
149   \_def\_bibfieldname{#2#3}%
150   \_if!#2\_relax
151     \_def\_bibfieldname{#3}%
152     \_RetrieveFieldIn{#3}\_bibfield
153     \_ifx\_bibfield\_empty\_else
154       \_RetrieveFieldIn{#3number}\_namecount
155       \_def\_bibfield{\_csname\_Read#3\_ea\_endcsname \_csname\_pp:#3\_endcsname}%
156     \_fi
157   \_else
158     \_RetrieveFieldIn{#2#3}\_bibfield
159   \_fi
160   \_if^#1^%
161     \_ifx\_bibfield\_empty \_ea\_ea\_ea \_doemptyfield
162     \_else \_ea\_ea\_ea \_dofullfield \_fi
163   \_else \_ea \_bprintaA
164   \_fi
165 }
166 \_def\_dofullfield#1#2{\_def\_dofield##1{#1}\_ea\_dofield\_ea{\_bibfield}}
167 \_def\_doemptyfield#1#2{\_def\_dofield##1{#2}\_ea\_dofield\_ea{\_bibfield}}
168 \_let\_Readauthor=\_ReadAuthor \_let\_Readeditor=\_ReadEditor
169 \_def\_bprintaA #1#2{\_ifx\_bibfield\_empty #2\_else\_bprintaB #1**\_eee\_fi}
170 \_def\_bprintaB #1*#2*#3\_eee{\_if^#3^#1\_else\_ea\_bprintaC\_ea{\_bibfield}{#1}{#2}\_fi}
171 \_def\_bprintaC #1#2#3{#2#1#3}
172 \_def\_bprintc#1#2{\_bprintcA#1#2**\_relax}
173 \_def\_bprintcA#1#2*#3*#4\_relax{\_ifx#1\_empty \_else \_if^#4^#2\_else#2#1#3\_fi\_fi}
174 \_def\_bprintv [#1]#2#3{\_def\_tmpa{#2}\_def\_tmpb{#3}\_bprintvA #1,,}
175 \_def\_bprintvA #1,{%
176   \_if^#1^\_tmpb\_else
177     \_RetrieveFieldIn{#1}\_tmp
178     \_ifx\_tmp\_empty
179     \_else \_tmpa \_def\_tmpb{}\_def\_tmpa{}%
180   \_fi
181   \_ea \_bprintvA
182   \_fi
183 }
184 \_sdef{\_pp:author}{\_letNames\_authorname}
185 \_sdef{\_pp:editor}{\_letNames\_editorname}
186 \_def\_letNames{\_let\_Firstname=\_Firstname \_let\_Lastname=\_Lastname
187   \_let\_Von=\_Von \_let\_Junior=\_Junior
188 }

```

Various macros + multilingual. Note that `\_nobibwarnlist` is used in `\_bibwarning` and it is set by `\nobibwarning` macro.



```

195 \_def\_bibwarning{%
196   \_ea\_isinlist \_ea\_nobibwarnlist\_ea{\_ea,\EntryKey,}\_iffalse
197   \_opwarning{Missing field "\_bibfieldname" in [\EntryKey]}\_fi}

```

### 2.32.5 Usage of the bib-iso690 style

This is the iso690 bibliographic style used by OpTeX.

See op-biblist.bib for an example of the .bib input. You can try it by:

```

\fontfam[LMfonts]
\nocite[*]
\usebib/s (iso690) op-biblist
\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 badly interpreted.

The entry names and field names are case insensitive. If there exists a data field not 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 a special printing of various fields.

There can be only one option field per each entry with (maybe) 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 in various formats (the von part is typically missing):

```

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

```

Only the Lastname part is mandatory. 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 the semifinal and final author the word “and” (or something different depending on the current language) is printed.

The first author is printed in reverse order: “LASTNAME, Firstname(s) von, After” and the other authors 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 the 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 are printed in this 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 are 8 or more authors then only the first author is printed.
```

Examples:

```
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 = "auetal",
```

output: GREEN, John, Bob BROWN, Alice BLACK et al.

If you need to add a text before or after the author's 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 the default behavior. Another example: `autrim:1` means that all Firstnames are trimmed.

```
author = "John Green and Bob Brown and Alice Black",
option = "auetal autrim:1",
```

output: GREEN, J., B. BROWN, A. BLACK et al.

If you need to write a team name or institution instead of authors, replace all spaces by `\_` in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after the 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 the 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 has to be used for Lastnames in the `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 a colon. Next, the optional secondary title has to be printed in an 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 the 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 `<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's 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, maybe 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 a different current language is used) before URL. It means, that the output of the 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 the 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 field only in the `@ARTICLE` entry. It means, that if you need to visible them in the `@INBOOK`, `@INPROCEEDINGS` etc. entries, then you must use the 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 the field is missing then a 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 occur.

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 the 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 the 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 this case. If it is used then it has precedence before the 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 for 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 the 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 of numbers. This style file auto-generates these marks in the form “Lastname of the first author, comma, space, the year” if the 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 like 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 (i.e. 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 placing the auto-citations at 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 the semifinal and final author) maybe 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 of the language (for example In:). Unfortunately, the bibTeX supposes that the entry data are not fully included in the fields so the automaton has to add some text during processing (“ed.”, “Vol.”, “see also”, etc.). But what language has to be chosen?

The current value of the \language register at the start of the .bib processing is described as the language of the outer document. This language is used for technical notes regardless of the entry language. Moreover, each entry can have the `lang` field (short name of the language). This language is used for ed./vyd., vol./ročník, etc. and it is used for hyphenation too. If the `lang` is not set then the outer document language is used.

You can use `\_Mtext{bib.<identifier>}` if you want to use a phrase dependent on outer document language (no on entry language). Example:

```
howpublished = "\_Mtext{bib.blue-ray}"
```

Now, you can set the variants of `bib.blue-ray` phrase for various languages:



```
\_sdef{\_mt:bib.blue-ray:en} {Blue-ray disc}
\_sdef{\_mt:bib.blue-ray:cs} {Blue-ray disk}
```

## Summary of non-standard fields

This style uses the following fields unknown by bib<sub>TEX</sub>:

```
option      ... options separated by spaces
lang        ... the language two-letter code of one entry
ednote      ... edition 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 of 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
```

Other options in the option field are silently ignored.

## 2.32.6 Implementation of the bib-iso690 style

```
3 \_codedecl \_undefined {BIB style (iso690) <2021-04-07>} % 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
```

**\\_maybetod** (alias **\.** in the style file group) does not put the 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
```

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\_biboptionsi{\_space \_the\_toks1 \_space \_the\_biboptions \_space}%
35 }
```

## Formating of Author/Editor lists.

bib-iso690.opm

```

41 \_def\_firstauthorformat{%
42   \_upper{\_Lastname}\_bprintc\_Firstname{, *}\_bprintc\_Von{ *}\_bprintc\_Junior{, *}%
43 }
44 \_def\_otherauthorformat{%
45   \_bprintc\_Firstname{* }\_bprintc\_Von{* }\_upper{\_Lastname}\_bprintc\_Junior{, *}%
46 }
47 \_def\_commonname{%
48   \_ifnum\_NameCount=1
49     \_firstauthorformat
50     \_ifx\_dobibmark\_undefined \_edef\_dobibmark{\_Lastname}\_fi
51   \_else
52     \_ifnum0\_namecount=\_NameCount
53     \_ifx\_maybeetal\_empty \_bibconjunctionand\_else , \_fi
54     \_else , \_fi
55     \_otherauthorformat
56   \_fi
57 }
58 \_def\_authorname{%
59   \_ifnum\_NameCount>0\_namecount\_relax\_else \_commonname \_fi
60   \_ifnum\_NameCount=0\_namecount\_relax \_maybeetal \_fi
61 }
62 \_let\_editorname=\_authorname
63
64 \_def\_prepareauedoptions#1{%
65   \_def\_mabyetal{}\\_csname lb@abbreviatefalse\_endcsname
66   \_biboptionvalue{#1max}\_authormax
67   \_biboptionvalue{#1min}\_authormin
68   \_biboptionvalue{#1pre}\_authorpre
69   \_biboptionvalue{#1print}\_authorprint
70   \_isbiboption{#1etal}\_iftrue \_def\_maybeetal{\_Mtext{bib.etal}}\_fi
71   \_biboptionvalue{#1trim}\_autrim
72   \_let\_namecountrow=\_namecount
73   \_ifx\_authormax\_empty \_else
74     \_ifnum 0\_authormax<0\_namecount
75     \_edef\_namecount{\_ifx\_authormin\_empty\_authormax\_else\_authormin\_fi}%
76     \_def\_maybeetal{\_Mtext{bib.etal}}%
77   \_fi\_fi
78   \_ifx\_autrim\_empty \_def\_autrim{10000}\_fi
79   \_ifnum\_autrim=0 \_def\_autrim{10000}\_fi
80   \_ifnum 0\_namecount<\_autrim\_relax \_else \_AbbreviateFirstname \_fi
81 }
82 \_def\_maybeetal{}
83
84 \_ifx\_upper\_undefined
85   \_ifx\_caps \_undefined \_def\_upper{\_uppercase\_ea}\_else
86     \_def\_upper#1{{\caps\_rm #1}}\_fi
87 \_fi
88 \_let\_upper=\_upper

```

## Preparing bib-mark (used when \nonumcitations is set).

bib-iso690.opm

```

94 \_def\_setbibmark{%
95   \_ifx\_dobibmark\_undefined \_def\_dobibmark{}\_fi
96   \_RetrieveFieldIn{bibmark}\_tmp
97   \_ifx\_tmp\_empty \_RetrieveFieldIn{year}\_tmp \_edef\_tmp{\_dobibmark, \_tmp}\_fi
98   \_bibmark=\_ea{\_tmp}%
99 }

```

## Setting phrases.

bib-iso690.opm

```

105 \_def\_bibconjunctionand{\_Mtext{bib.and}}
106 \_def\_preurl{\_Mtext{bib.available}}
107 \_let\_predoi=\_preurl
108 \_def\_postedition{\_mtext{bib.edition}}
109 \_def\_Inclause{In:~}
110 \_def\_prevolume{\_mtext{bib.volume}}
111 \_def\_prenumber{\_mtext{bib.number}}
112 \_def\_prepages{\_mtext{bib.prepages}}
113 \_def\_posteditor{\_ifnum0\_namecountrow>1 \_Mtext{bib.editors}\_else\_Mtext{bib.editor}\_fi}

```

`\_Mtext{<identifier>}` expands to a phrase by outer document language (no entry language).

bib-iso690.opm

```
120 \_chardef\_documentlanguage=\_language
121 \_def\_Mtext#1{\_csname _mt:#1:\_csname _lan:\_the\_documentlanguage\_endcsname\_endcsname}
122
123 \_CreateField {lang}
124 \_def\_setlang#1{\_ifx#1\_empty \_else
125     \_ifcsname _mt:bib.and:#1\_endcsname \_language=\_csname _#1Patt\_endcsname \_relax
126     \_else \_opwarning{No phrases for "#1" used by [\EntryKey] in .bib}%
127     \_fi\_fi
128 }
```

Non-standard field names.

bib-iso690.opm

```
134 \_CreateField {ednote}
135 \_CreateField {citedate}
136 \_CreateField {numbering}
137 \_CreateField {isbn}
138 \_CreateField {issn}
139 \_CreateField {doi}
140 \_CreateField {url}
141 \_CreateField {bibmark}
```

Sorting.

bib-iso690.opm

```
147 \_SortingOrder{name,year}{lfbvj}
148 \_SpecialSort {key}
```

Supporting macros.

bib-iso690.opm

```
154 \_def\_bibwarninga{\_bibwarning}
155 \_def\_bibwarningb{\_bibwarning}
156
157 \_def\_docitedate #1/#2/#3/#4\_relax{[\_Mtext{bib.citedate}]%
158     \_if^#2^#1\_else
159     \_if^#3^#1/#2\_else
160     \_csf\_csf\_lan:\_the\_documentlanguage}dateformat}#1/#2/#3\_relax
161     \_fi\_fi ]%
162 }
163 \_def\_doyear#1{
164     \_biboptionvalue{yearprint}\_yearprint
165     \_ifx\_yearprint\_empty#1\_else\_def\YEAR{#1}\_yearprint\_fi
166 }
167 \_def\_preparenumbering{%
168     \_def\VOL{\_RetrieveField{volume}}%
169     \_def\NO{\_RetrieveField{number}}%
170     \_def\PP{\_RetrieveField{pages}}%
171 }
172 \_def\_prepareednote{%
173     \_def\EDN{\_RetrieveField{edition}}%
174     \_def\ADDR{\_RetrieveField{address}}%
175     \_def\PUBL{\_RetrieveField{publisher}}%
176     \_def\YEAR{\_RetrieveField{year}}%
177     \_def\AU{\_bprintb[!author]{\_doauthor0{####1}}{}}%
178     \_def\ED{\_bprintb[!editor]{\_doeditor0{####1}}{}}%
179     \_preparenumbering
180 }
181 \_def\_doedition#1{%
182     \_biboptionvalue{editionprint}\_editionprint
183     \_ifx\_editionprint\_empty#1\_postedition\_else\_def\ED{#1}\_editionprint\_fi
184 }
185 \_def\_doauthor#1#2{\_prepareauedoptions{au}\_let\_iseditorlist=\_undefined
186     \_if1#1\_def\AU{#2}\_else\_let\_authorprint=\_empty\_fi
187     \_ifx\_authorprint\_empty #2\_else \_authorprint\_fi
188 }
189 \_def\_doeditor#1#2{\_prepareauedoptions{ed}\_let\_firstauthorformat=\_otherauthorformat
190     \_if1#1\_def\ED{#2}\_else\_let\_authorprint=\_empty\_fi
191     \_ifx\_authorprint\_empty #2\_posteditor\_else \_authorprint\_fi
192 }
```

Entry types.

```

198 \_sdef{_print:BEGIN}{%
199 \_readbiboptions
200 \_biboptionvalue{titlepost}\_titlepost
201 \_isbiboption{unpublished}\_iftrue \_let\_bibwarninga=\_relax \_let\_bibwarningb=\_relax \_fi
202 \_isbiboption{nowarn}\_iftrue \_let\_bibwarning=\_relax \_fi
203 \_isbiboption{urlalso}\_iftrue \_def\_preurl{\_Mtext{bib.availablealso}}\_fi
204 \_RetrieveFieldIn{lang}\_langentry \_setlang\_langentry
205 }
206 \_sdef{_print:END}{%
207 \_bprinta [note] {*.}\}%
208 \_setbibmark
209 }
210 \_def\_bookgeneric#1{%
211 \_bprinta [howpublished] {[*].\ }\}%
212 \_bprintb [edition] {\_doedition{##1}\.\ }\}%
213 \_bprinta [ednote] {*.}\ }\}%
214 \_bprinta [address] {*\_bprintv[publisher]{:}\_bprintv[year]{,}\.\ }\_bibwarninga}%
215 \_bprinta [publisher] {*\_bprintv[year]{,}\.\ }\_bibwarninga}%
216 \_bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}\.\ }\ }%
217 {\_bibwarning}%
218 \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{\.\.\ }\}%
219 \_bprinta [citedate] {\_docitedate*///\_relax.\ }\}%
220 #1%
221 \_bprinta [series] {*.}\ }\}%
222 \_bprinta [isbn] {ISBN~*.\ }\_bibwarningb}%
223 \_bprinta [issn] {ISSN~*.\ }\}%
224 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }\}%
225 \_bprintb [url] {\_preurl\_url{##1}. }\}%
226 }
227 \_sdef{_print:book}{%
228 \_bprintb [!author] {\_doauthor1{##1}\.\ }\_bibwarning}%
229 \_bprintb [title] {\_em##1}\_bprintc\_titlepost{\.\ *}\_bprintv[howpublished]{\.\.\ }\ }%
230 {\_bibwarning}%
231 \_bookgeneric{%
232 }
233 \_sdef{_print:article}{%
234 \_biboptionvalue{journalpost}\_journalpost
235 \_bprintb [!author] {\_doauthor1{##1}\.\ }\_bibwarning}%
236 \_bprinta [title] {*. \_bprintc\_titlepost{*.}\ }\_bibwarning}%
237 \_bprintb [journal] {\_em##1}\_bprintc\_journalpost{\.\ *}\_bprintv[howpublished]{\.\.\ }\ }%
238 {\_bibwarninga}%
239 \_bprinta [howpublished] {[*].\ }\}%
240 \_bprinta [address] {*\_bprintb[publisher]{:}\.\ }\}%
241 \_bprinta [publisher] {*, }\}%
242 \_bprinta [month] {*, }\}%
243 \_bprintb [year] {\_doyear{##1}\_bprintv[volume,number,pages]{,}\.\.\ }\}%
244 \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{\.\.\ }\ }%
245 {\_bprinta [volume] {\_prevolume*\_bprintv[number,pages]{,}\.\.\ }\}%
246 {\_bprinta [number] {\_prenumber*\_bprintv[pages]{,}\.\.\ }\}%
247 {\_bprintb [pages] {\_prepages\_hbox{##1}\_bprintv[citedate]{\.\.\ }\ }%
248 {\_bibwarninga}%
249 \_bprinta [citedate] {\_docitedate*///\_relax.\ }\}%
250 \_bprinta [issn] {ISSN~*.\ }\}%
251 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }\}%
252 \_bprintb [url] {\_preurl\_url{##1}. }\}%
253 }
254 \_sdef{_print:inbook}{%
255 \_let\_bibwarningb=\_relax
256 \_bprintb [!author] {\_doauthor1{##1}\.\ }\_bibwarning}%
257 \_bprinta [title] {*.}\\_bibwarning}%
258 \_Inclause
259 \_bprintb [!editor] {\_doeditor1{##1}\.\ }\}%
260 \_bprintb [booktitle] {\_em##1}\_bprintc\_titlepost{\.\ *}\_bprintv[howpublished]{\.\.\ }\ }%
261 {\_bibwarning}%
262 \_bookgeneric{\_bprintb [pages] {\_prepages\_hbox{##1}. }\}%
263 }
264 \_slet{_print:inproceedings}{\_print:inbook}
265 \_slet{_print:conference}{\_print:inbook}
266

```

```

267 \_sdef{\_print:thesis}{%
268 \_bprintb [!author] {\_doauthor1{##1}\. \ }{\_bibwarning}%
269 \_bprintb [title] {\_em##1}\_bprintc\_titlepost{\. \ *}\_bprintv[howpublished]{\{.\} \ }%
270 {\_bibwarning}%
271 \_bprinta [howpublished] {[*]. \ }{%
272 \_bprinta [address] {\*\_bprintv[school]{.}\_bprintv[year]{.}\. \ }{\_bibwarning}%
273 \_bprinta [school] {\*\_bprintv[year]{.}\. \ }{\_bibwarning}%
274 \_bprinta [month] {\*, }{%
275 \_bprintb [year] {\_doyear{##1}\_bprintv[citedate]{.}\. \ }{\_bibwarning}%
276 \_bprinta [citedate] {\_docitedate*///\_relax. \ }{%
277 \_bprinta [type] {\*\_bprintv[ednote]{.}\. \ }%
278 {\_ifx\_thesistype\_undefined\_bibwarning
279 \_else\_thesistype\_bprintv[ednote]{.}\. \ } \_fi}%
280 \_bprinta [ednote] {\*. \ }{%
281 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}. \ }{%
282 \_bprintb [url] {\_preurl\_url{##1}. \ }{%
283 }
284 \_sdef{\_print:phdthesis}{\_def\_thesistype{\_Mtext{bib.phdthesis}}\_cs{\_print:thesis}}
285 \_sdef{\_print:mastersthesis}{\_def\_thesistype{\_Mtext{bib.mastthesis}}\_cs{\_print:thesis}}
286 \_sdef{\_print:bachelorsthesi}{\_def\_thesistype{\_Mtext{bib.bachthesis}}\_cs{\_print:thesis}}
287
288 \_sdef{\_print:generic}{%
289 \_bprintb [!author] {\_doauthor1{##1}\. \ }{\_bibwarning}%
290 \_bprintb [title] {\_em##1}\_bprintc\_titlepost{\. \ *}\_bprintv[howpublished]{\{.\} \ }%
291 {\_bibwarning}%
292 \_bprinta [howpublished] {[*]. \ }{%
293 \_bprinta [ednote] {\_prepareednote*\_bprintv[citedate]{.}\. \ }{\_bibwarning}%
294 \_bprinta [year] {\_bibwarning}%
295 \_bprinta [citedate] {\_docitedate*///\_relax. \ }{%
296 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}. \ }{%
297 \_bprintb [url] {\_preurl\_url{##1}. \ }{%
298 }
299 \_slet{\_print:booklet}{\_print:generic}
300 \_slet{\_print:incolleciton}{\_print:generic}
301 \_slet{\_print>manual}{\_print:generic}
302 \_slet{\_print:proceedings}{\_print:generic}
303 \_slet{\_print:techreport}{\_print:generic}
304 \_slet{\_print:unpublished}{\_print:generic}
305
306 \_sdef{\_print:misc}{\_let\_bibwarning=\_relax \_cs{\_print:generic}}

```

## 2.33 Sorting and making Index

makeindex.opm

```
3 \_codedecl \_makeindex {Makeindex and sorting <2021-02-15>} % loaded in format
```

`\_makeindex` implements sorting algorithm at  $\TeX$  macro-language level. You need not any external program.

There are two passes in the sorting algorithm. The 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 the secondary pass is started. It distinguishes 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 the sorting algorithm. The Czech rules (cs) are implemented here:

makeindex.opm

```

25 \_def \_sortingdatacs {%
26 /,{ },-,&,@,%
27 aAäÄáÁ,%
28 bB,%
29 cC,%
30 čČ,%
31 dDďĎ,%

```

```

32 eEéĚěĚĚ,%
33 fF,%
34 gG,%
35 hH,%
36 ^T^U^V,% ch Ch CH
37 iIíĚ,%
38 jJ,%
39 kK,%
40 lLĺĚ,%
41 mM,%
42 nNňĚ,%
43 oOöŮóôŮ,%
44 pP,%
45 qQ,%
46 rRřŘ,%
47 řŘ,%
48 sS,%
49 šŠ,%
50 tTťĚ,%
51 uUúŮůůŮ,%
52 vV,%
53 wW,%
54 xX,%
55 yYýĚ,%
56 zZ,%
57 žŽ,%
58 0,1,2,3,4,5,6,7,8,9,'%
59 }

```

Characters ignored by the sorting algorithm are declared in `\_ignoredchars<iso-code>`. The compound characters (two or more characters interpreted as one character in the sorting algorithm) are mapped to single invisible characters in `\_compoundchars<iso-code>`. Czech rules declare `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 the 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 a subset of the 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 with 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 codes. 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\old{ 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,\advance\tmpnum by1 \lccode`##1=\tmpnum \fi
135 \ea\act \fi}%
136 \tmpnum=64 \ea\act \sortingdata \relax
137 }

```

Strings to be sorted are prepared in `\,<string>` control sequences (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 the first character from both converted strings, compares them and if it is equal then calls itself recursively else gives the 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 a comma when it starts. It ends with sorted items in `\_iilist` without commas. So `\_dosorting` macro must prepare commas between items.

makeindex.opm

```

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+neco cele znova
225 \_edef\_iilist{\_ea}\_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 with 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.

makeindex.opm

```

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`.

makeindex.opm

```

285 \_def\_makeindex{\_par
286 \_ifx\_iilist\_empty \_opwarning{index data-buffer is empty. TeX me again}%
287 \_incr\_unresolvedrefs
288 \_else
289 \_dosorting \_iilist % sorting \_iilist
290 \_bgroup
291 \_rightskip=Opt plus1fil \_exhyphenpenalty=10000 \_leftskip=\_iindent
292 \_ea\_xargs \_ea\_printindexitem \_iilist ;\_par
293 \_egroup
294 \_fi
295 }
296 \_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.

makeindex.opm

```

306 \_def\_printindexitem #1{%
307 \_ifcsname \_csstring #1\_endcsname
308 \_ea\_ea\_ea \_printii \_csname \_csstring #1\_endcsname &%
309 \_else
310 \_ea\_ea\_ea \_printii \_ea\_ignorefirst \_csstring #1&%
311 \_fi
312 \_ea\_printiipages #1&
313 }

```

`\_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}

```

makeindex.opm

```

330 \_def\_printii #1#2{%
331 \_ismacro\_lastii{#1}\_iffalse \_newiiletter{#1}{#2}\_def\_lastii{#1}\_fi
332 \_gdef\_currii{#1#2}\_the\_everyii\noindent
333 \_hskip-\_iindent \_ignorespaces\_printiiA#1#2//}
334 \_def\_printiiA #1/{\_if^#1\_let\_previi=\_currii \_else
335 \_ea\_scanprevii\_previi/&\_edef\_tmpb{\_detokenize{#1}}%
336 \_ifx\_tmpa\_tmpb \_iiemdash \_else#1 \_gdef\_previi{}\_fi
337 \_ea\_printiiA\_fi
338 }
339 \_def\_iiemdash{\_kern.1em---\_space}
340 \_def\_lastii{}
341 \_def\_newiiletter#1#2{}
342
343 \_def\_scanprevii#1/#2&{\_def\_previi{#2}\_edef\_tmpa{\_detokenize{#1}}}
344 \_def\_previi{} % previous index item

```

`\_printiipages` `<pglist>&` gets `<pglist>` in the form `<pg>:<type>,<pg>:<type>,...<pg>:<type>` and it converts them to `<pg>`, `<pg>`, `<from>--<to>`, `<pg>` etc. The same pages must be printed only once and continuous consequences of pages must be compressed to the form `<from>-<to>`. Moreover, the consequence is continuous only if all pages have the same `<type>`. Empty `<type>` is most common, pages with `b` `<type>` 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 solve these tasks.

makeindex.opm

```

358 \def\printiipages#1&{\let\pgtype=\undefined \tmpnum=0 \printpages #1,:\_par}
359 \def\printpages#1:#2,{% state automaton for comprimg pages
360 \ifx,#1,\uselastpgnum
361 \else \def\tmpa{#2}%
362 \ifx\pgtype\tmpa \else
363 \let\pgtype=\tmpa
364 \uselastpgnum \usepgcomma \pgprint#1:{#2}%
365 \tmpnum=#1 \returnfi \fi
366 \ifnum\tmpnum=#1 \returnfi \fi
367 \advance\tmpnum by1
368 \ifnum\tmpnum=#1 \ifx\lastpgnum\undefined \usepgdash\fi
369 \edef\lastpgnum{\the\tmpnum:\pgtype}%
370 \returnfi \fi
371 \uselastpgnum \usepgcomma \pgprint#1:{#2}%
372 \tmpnum=#1
373 \relax
374 \ea\printpages \fi
375 }
376 \def\returnfi #1\relax{\fi}
377 \def\uselastpgnum{\ifx\lastpgnum\undefined
378 \else \ea\pgprint\lastpgnum \let\lastpgnum=\undefined \fi
379 }
380 \def\usepgcomma{\ifnum\tmpnum>0, \fi} % comma+space between page numbers
381 \def\usepgdash{\hbox{--}} % dash in the <from>--<to> form

```

You can re-define  $\backslash\pgprint \langle gpageno \rangle:\{\langle iitype \rangle\}$  if you need to implement more  $\langle iitypes \rangle$ .

makeindex.opm

```

388 \def\pgprint #1:#2{%
389 \ifx ,#2,\pgprintA{#1}\returnfi \fi
390 \ifx b#2{\bf \pgprintA{#1}}\returnfi \fi
391 \ifx i#2{\it \pgprintA{#1}}\returnfi \fi
392 \ifx u#2\pgu{\pgprintA{#1}}\returnfi \fi
393 \pgprintA{#1}\relax
394 }
395 \def\pgprintA #1{\_ilink[pg:#1]{\_cs{\_pgi:#1}} % \ilink[pg:<gpageno>]{<pageno>}
396 \def\pgu#1{\_leavevmode\_vtop{\_hbox{#1}\kern.3ex\_hrule}}

```

The  $\backslash\iindex\{\langle word \rangle\}$  puts one  $\langle word \rangle$  to the index. It writes  $\backslash\_Xindex\{\langle word \rangle\}\{\langle iitype \rangle\}$  to the .ref file. All other variants of indexing macros expand internally to  $\backslash\iindex$ .

makeindex.opm

```

404 \def\iindex#1{\_isempty{#1}\_iffalse
405 \openref{\def~{ }\_ewref\_Xindex{#1}{\_iitiesaved}}}\_fi}
406 \_public \iindex ;

```

The  $\backslash\_Xindex\{\langle word \rangle\}\{\langle iitype \rangle\}$  stores  $\backslash,\langle word \rangle$  to the  $\backslash\_iilist$  if there is the 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. Moreover, 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  $\backslash\_pgi:\langle gpageno \rangle$  is defined as  $\langle pageno \rangle$ .

makeindex.opm

```

418 \def \_iilist {}
419 \def \_Xindex #1#2{\_ea\_XindexA \_csname ,#1\_ea\_endcsname \_currpage {#2}}
420 \def \_XindexA #1#2#3#4{% #1=\,<word> #2=<gpageno> #3=<pageno> #4=<iitype>
421 \ifx#1\_relax \_global\_addto \_iilist {#1}%
422 \_gdef #1{#2:#4}%
423 \else \_global\_addto #1{,#2:#4}%
424 \_fi
425 \_sxddef{\_pgi:#2}{#3}%
426 }

```

The implementation of macros  $\backslash\ii$ ,  $\backslash\iid$ ,  $\backslash\iis$  follows. Note that  $\backslash\ii$  works in the horizontal mode in order to the  $\backslash\write$  whatsit is not broken from the following word. If you need to keep vertical mode, use  $\backslash\iindex\{\langle word \rangle\}$  directly.

The  $\backslash\iitype \{\langle type \rangle\}$  saves the  $\langle type \rangle$  to the  $\backslash\_iitiesaved$  macro. It is used in the  $\backslash\iindex$  macro.

```

438 \def\ii #1 {\leavevmode\def\tmp{#1}\_iiA #1,\_def\_iitypesaved{}}
439
440 \def\_iiA #1,{\_if$#1$\_else\_def\tmpa{#1}%
441 \_ifx\tmpa\_iiatsign \_ea\_iiB\tmp,,\_else\_iindex{#1}\_fi
442 \_ea\_iiA\_fi}
443 \def\_iiatsign{0}
444
445 \def\_iiB #1,{\_if$#1$\_else \_iiC#1/\_relax \_ea\_iiB\_fi}
446 \def\_iiC #1/#2\_relax{\_if$#2$\_else\_iindex{#2#1}\_fi}
447
448 \def\_iid #1 {\leavevmode\_iindex{#1}#1\_futurelet\tmp\_iid\_def\_iitypesaved{}}
449 \def\_iidD{\_ifx\tmp,\_else\_ifx\tmp.\_else\_space\_fi\_fi}
450
451 \def\_iis #1 #2{{\_def~{ }\_global\_sdef{_,#1}{#2}}\_ignorespaces}
452
453 \def\_iitypesaved{}
454 \def\_iitype #1{\_def\_iitypesaved{#1}\_ignorespaces}
455
456 \_public \ii \iid \iis \iitype ;

```

## 2.34 Footnotes and marginal notes

```

3 \_codedecl \fnote {Footnotes, marginal notes OpTeX <2020-05-26>} % loaded in format

```

`\_gfnotenum` is a counter which counts footnotes globally in the whole document.

`\_lfnotenum` is a 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 this case.

`\fnotenum` is a macro that 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.

```

18 \_newcount\_gfnotenum \_gfnotenum=0
19 \_newcount\_lfnotenum
20
21 \_newifi \_ifpgfnote
22 \_def \_fnotenumglobal {\_def\_fnotenum{\_the\_gfnotenum}\_pgfnotefalse}
23 \_def \_fnotenumchapters {\_def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse}
24 \_def \_fnotenumpages {\_def\_fnotenum{\_trycs{\_fn:\_the\_gfnotenum}{?}}\_pgfnotetrue}
25 \_fnotenumchapters % default are footnotes counted from one in each chapter
26 \_def \_fnotenum{\_fnotenum}
27 \_public \fnotenumglobal \fnotenumchapters \fnotenumpages ;
28 \_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$\mathbox{\dagger}$\or$\mathbox{\ddagger}$\or$\mathbox{\dagger\dagger}$\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 the footnote itself, then you can define `\_printfnotemarkA` and `\_printfnotemarkB`.

The `\fnotelinks<colorA><colorB>` implements the hyperlinked footnotes (from text to footnote and backward).

```

48 \_def \_printfnotemark {$\_{\_fnotenum}$} % default footnote mark
49 \_def \_printfnotemarkA {\_printfnotemark} % footnote marks used in text
50 \_def \_printfnotemarkB {\_printfnotemark} % footnote marks used in front of footnotes
51
52 \_def \_fnotelinks#1#2{% <inText color> <inFootnote color>

```

```

53 \def\printfnoteA{\link[fnt:\the\_gfnotenum]{#1}{\printfnote}%
54 \dest[fnt:\the\_gfnotenum]}%
55 \def\printfnoteB{\link[fnt:\the\_gfnotenum]{#2}{\printfnote}%
56 \dest[fnt:\the\_gfnotenum]}%
57 }
58 \public \fnotelinks ;

```

Each footnote saves the `\_Xfnote` (without parameter) to the `.ref` file (if `\openref`). We can create the mapping from `\_gfnotenum` to `\_pgfnotenum` in the macro `\_fn:\_gfnotenum`. Each `\_Xpage` macro sets the `\_lfnotenum` to zero.

fnotes.opm

```

67 \def \_Xfnote {\_incr\_lfnotenum \_incr\_gfnotenum
68 \_sxdef{\_fn:\_the\_gfnotenum}{\_the\_lfnotenum}}

```

The `\_fnote {<text>}` macro is simple, `\_fnotemark` and `\_fnotetext` does the real work.

fnotes.opm

```

75 \def\_fnote{\_fnotemark1\_fnotetext}
76 \def\_fnotemark#1{\_advance\_gfnotenum by#1\_advance\_lfnotenum by#1\_relax \_printfnoteA}}

```

The `\_fnotetext` calls `\_opfootnote` which is equivalent to plain TeX `\vfootnote`. It creates new data to Insert `\footins`. The only difference is that we can propagate a macro parameter into the Insert group before the text is printed (see section 2.18). This propagated macro is `\_fnset` which sets smaller fonts.

Note that `\vfootnote` and `\_opfootnote` don't read the text as a parameter but during the normal horizontal mode. This is the reason why catcode changes (for example in-line verbatim) can be used here.

fnotes.opm

```

90 \def\_fnotetext{\_incr\_gfnotenum \_incr\_lfnotenum % global increment
91 \_ifpgfnote \_openref \_fi
92 \_wref \_Xfnote}%
93 \_ifpgfnote \_ifcsname \_fn:\_the\_gfnotenum \_endcsname \_else
94 \_opwarning{unknown \_noexpand\_fnote mark. TeX me again}%
95 \_incr\_unresolvedrefs
96 \_fi\_fi
97 \_opfootnote\_fnset\_printfnoteB
98 }
99 \def\_fnset{\_everypar={}\_scalemain \_typoscale[800/800]}
100
101 \_public \fnote \fnotemark \fnotetext ;

```

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:\_mnotenum` as `\_right` or `\_left` when the `.ref` file is read. The `\_ifnum 0<0#2` trick returns true if `\_pageno` has a numeric type and false if it is a non-numeric type (Roman numeral, for example). We prefer to use `\_pageno`, but only if it has the numeric type. We use `\_gpageno` in other cases.

fnotes.opm

```

113 \_newcount\_mnotenum \_mnotenum=0 % global counter of mnotes
114 \_def \_Xmnote {\_incr\_mnotenum \_ea \_XmnoteA \_currpage}
115 \_def \_XmnoteA #1#2{\_ #1=<gpageno> #2=<pageno>
116 \_sxdef{\_mn:\_the\_mnotenum}{\_ifodd\_numtype{#2}{#1} \_right \_else \_left \_fi}}
117 \_def \_numtype #1#2{\_ifnum 0<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 `\_mnoteD{<text>}` macro sets the position of the marginal note. 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

```

135 \_def\_mnote #1{\_ifx^#1\_else \_mnoteC#1\_end \_fi \_mnoteD}
136 \_def\_mnoteC up#1\_end{\_mnoteskip=#1\_relax} % \_mnote up<dimen> {<text>} syntax
137 \_long\_def\_mnoteD#1{\_ifvmode {\_mnoteA{#1}}\_nobreak\_vskip-\_baselineskip \_else
138 \_lower\_dp\_strutbox\_hbox{\_vadjust{\_kern-\_dp\_strutbox \_mnoteA{#1}}\_kern\_dp\_strutbox}%
139 \_fi
140 }
141 \_public \_mnote ;

```



The `\mnoteskip` is a `dimen` value that denotes the vertical shift of marginal note from its normal position. A positive value means shift up, negative down. The `\mnoteskip` register is set to zero after the marginal note is printed. The new syntax `\mnote up<dimen>{<text>}` is possible too, but public `\mnoteskip` is kept for backward compatibility.

fnotes.opm

```
151 \_newdimen\mnoteskip
152 \_public \mnoteskip ;
```

The `\_mnoteA` macro does the real work. The `\_lrmnote{<left>}{<right>}` uses only first or only second parameter depending on the left or right marginal note.

fnotes.opm

```
160 \_long\_def\_mnoteA #1{\_incr\_mnotenum
161 \_ifx\_mnotesfixed\_undefined
162 \_ifcsname \_mn:\_the\_mnotenum \_endcsname
163 \_edef\_mnotesfixed{\_cs{\_mn:\_the\_mnotenum}}}%
164 \_else
165 \_opwarning{unknown \_noexpand\_mnote side. TeX me again}\_openref
166 \_incr\_unresolvedrefs
167 \_def\_mnotesfixed{\_right}%
168 \_fi\_fi
169 \_hbox to0pt{\_wref\_Xmnote{}}\_everypar={}%
170 \_lrmnote{\_kern-\_mnotesize \_kern-\_mnoteindent}{\_kern\_hsize \_kern\_mnoteindent}%
171 \_vbox to0pt{\_vss \_setbox0=\_vtop{\_hsize=\_mnotesize
172 \_lrmnote{\_leftskip=0pt plus 1fill \_rightskip=0pt}
173 {\_rightskip=0pt plus 1fil \_leftskip=0pt}}%
174 {\_the\_everymnote\_noindent#1\_endgraf}}}%
175 \_dp0=0pt \_box0 \_kern\_mnoteskip \_global\_mnoteskip=0pt}\_hss}%
176 }
177 \_def \_lrmnote#1#2{\_ea\_ifx\_mnotesfixed\_left #1\_else #2\_fi}
```

We don't want to process `\fnote`, `\fnotemark`, `\mnote` in TOC, headlines nor outlines.

fnotes.opm

```
184 \_regmacro {\_def\fnote#1{}} {\_def\fnote#1{}} {\_def\fnote#1{}}
185 \_regmacro {\_def\fnotemark#1{}} {\_def\fnotemark#1{}} {\_def\fnotemark#1{}}
186 \_regmacro {\_def\mnote#1{}} {\_def\mnote#1{}} {\_def\mnote#1{}}
```

## 2.35 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.

### 2.35.1 \report and \letter styles

styles.opm

```
3 \_codedecl \report {Basic styles of OpTeX <2021-03-10>} % 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 2.35.2.

```

62 \_def\_slides{\_par
63   \_opinput{slides.opm}
64   \_adeft*{\_relax\_ifmode*\_else\_ea\_startitem\_fi}
65 }
66 \_public \slides ;

```

## 2.35.2 \slides style for presentations

```

3 \_codedecl \slideshow {Slides style for OpTeX <2021-04-22>} % 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 \_ifx\_fontnamegen\_undefined \_fontfam[Heros]
16   \_let\_ttfont=\_undefined \_famvardef\_ttfont{\_setfontsize{mag1.15}\_tt}
17 \_fi
18 \_typo[16/19]
19 \_def\_urlfont{}
20 \_everytt={\_typo[13/16] \_advance\_hsize by10mm}
21 \_fontdef\_fixbf{\_bf}
22
23 \_nopagenumbers
24 \_parindent=0pt
25 \_ttindent=5mm
26 \_parskip=5pt plus 4pt minus2pt
27 \_rightskip=0pt plus 1fil
28 \_ttindent=10pt
29 \_def\_ttskip{\_smallskip}
30
31 \_onlyrgb % RGB color space is better for presentations

```

The bottom margin is set to 3 mm. If we use 1 mm, then the baseline of `\footline` is 2 mm from the bottom page. This is the 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.

```

41 \_footlinedist=1mm
42 \_footline={\_hss \_rlap{%
43   \_rlap{\Grey\_kern.2\_hoffset\_vrule height6mm depth2mm width.8\_hoffset}%
44   \_hbox to\_hoffset{\White\_hss\_folio\_kern3mm}}}

```

The `\subtit` is defined analogically like `\tit`.

```

50 \_eoldef\_subtit#1{\_vskip20pt {\_leftskip=0pt plus1fill \_rightskip=\_leftskip
51   \_subtitfont #1\_nbp}}

```

The `\pshow<num>` prints the text in invisible (transparent) font when `\layernum<num>`. The transparency is set by `\pdfpageresources` primitive.

```

59 \pdfpageresources{/ExtGState << /Invisible << /Type /ExtGState /ca 0 /CA 0 >>
60   /Visible << /Type /ExtGState /ca 1 /CA 1 >> >>}
61 \addto\_morepgresources{/Invisible << /Type /ExtGState /ca 0 /CA 0 >>
62   /Visible << /Type /ExtGState /ca 1 /CA 1 >>}
63 \def\Invisible {\_pdfliteral{/Invisible gs}}
64 \def\Visible {\_pdfliteral{/Visible gs}}
65 \def\Transparent {\Invisible \_aftergroup \Visible}
66
67 \_def\_use#1#2{\_ifnum\_layernum#1\_relax#2\_fi}
68 \_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 a nested level of items and `\pg+` is used then `\egroups` macro expands to the right number of `\egroups` to close the 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+`.

```

80 \_newcount\_gilevel
81 \_def\*{*}
82 \_adef*\_relax\_ifmode*\_else\_ea\_startitem\_fi} % defined also in styles.opm
83 \_sdef{\_item:X}{\Blue\_raise.2ex\_fullrectangle{.8ex}\_kern.5em}
84 \_sdef{\_item:x}{\Blue\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
85 \_style X
86 \_def\_egroups{\_par\_global\_gilevel=\_ilevel \_egroup}
87 \_everylist={\_novspaces \_ifcase\_ilevel \_or \_style x \_else \_style - \_fi
88   \_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.

```

95 \_def\_pg#1{\_cs{\_spg:#1}}
96 \_sdef{\_spg;}{\_vfil\_break \_lfnotenumreset}
97 \_sdef{\_spg.}{\_endslides}
98 \_sdef{\_spg+}{\_par}

```

The `\_endslides` is defined as `\_end` primitive (preceded by `\_byehook`), but slide-designer can redefine it. For example, [OpTeX trick 0029](#) shows how to define clickable navigation to the pages and how to check the data integrity at the end of the document using `\_endslides`.

The `\bye` macro is redefined here as an alternative to `\pg.`

```

110 \_def\_endslides{\_byehook \_end}
111 \_def\bye{\_pg.}

```

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`.

```

119 \_def\_titfont{\_typosize[42/60]\_bf \Blue}
120 \_def\_subtitfont{\_typosize[20/30]\_bf}
121 \_def\_secfont{\_typosize[25/30]\_bf \Blue}
122
123 \_nonum \_notoc \_let\_resetnonumnotoc=\_relax
124 \_def\_printsec#1{\_par
125   \_abovetitle{\_penalty-400}\_bigskip
126   {\_secfont \_noindent \_leftskip=0pt plus1fill \_rightskip=\_leftskip
127     \_printrefnum[@\_quad]#1\_nbp}\_insertmark{#1}%
128   \_nobreak \_belowtitle{\_medskip}%
129 }

```

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  $\langle\textit{layered-text}\rangle$  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 and prints page by the `\_printlayers`

slides.opm

```

147 \_newbox\slidepage \_newbox\slidepageB
148 \_countdef\_slidelayer=1
149
150 \_def\slideshow{\_slidelayer=1 \_slideshowactive
151   \_let\slideopen=\_relax % first wins
152   \_setbox\slidepage=\_vbox\_bgroup\_bgroup}
153
154 \_def\_slideshowactive{%
155   \_sdef\_spg:;{\_closepage \_global\_slidelayer=1 \_resetpage \_openslide}
156   \_sdef\_spg:.\_closepage \_endslides}
157   \_sdef\_spg:+{\_closepage \_incr\_slidelayer \_decr\_pageno \_openslide}
158   \_let\_layers=\_layersactive
159   \_slidelinks % to prevent hyperlink-dests duplication
160 }
161 \_def\_openslide{\_setbox\slidepage=\_vbox\_bgroup\_bgroup \_setilevel
162   \_ifvoid\slidepage \_else \_unvbox\slidepage \_nointerlineskip\_lastbox \_fi}
163 \_def\_setilevel{\_loop \_decr\_gilevel \_ifnum\_gilevel<0 \_else \_begitem \_repeat}
164
165 \_def\_closepage{\_egroups \_egroup
166   \_ifnum \_maxlayers=0 \_unvcopy\slidepage \_vfil\_break
167   \_else \_begingroup \_setwarnslides \_layernum=0
168     \_loop
169     \_ifnum\_layernum<\_maxlayers \_advance\_layernum by1
170     \_printlayers \_vfil\_break
171     \_ifnum\_layernum<\_maxlayers \_incr\_slidelayer \_decr\_pageno \_fi
172     \_repeat
173     \_global\_maxlayers=0
174     \_incr\_layernum \_global\_setbox\slidepage=\_vbox{\_printlayers}%
175   \_endgroup
176   \_fi}
177 \_def\_resetpage{%
178   \_global\_setbox\slidepage=\_box\_voidbox \_global\_setbox\slidepageB=\_box\_voidbox
179   \_ifnotenumreset
180 }
181 \_def\_setwarnslides{%
182   \_def\pg##1{\_opwarning{\_string\pg##1 \_layersenv}\_def\pg####1{}}%
183   \_def\layers##1 {\_opwarning{\_string\layers\_space \_layersenv}\_def\layers####1{}}%
184 }
185 \_def\_layersenv{cannot be inside \_string\layers...\_string\endlayers, ignored}
186
187 \_def\_printlayers{\_unvcopy\slidepage \_prevdepth=\_dp\slidepage
188   {\_layertext \_endgraf}%
189   \_vskip\_parskip
190   \_unvcopy\slidepageB
191 }
192 \_let\_destboxori=\_destbox
193
194 \_newcount\_layernum \_newcount\_maxlayers
195 \_maxlayers=0
196
197 \_long\_def\_layersactive #1 #2\endlayers{%
198   \_par\_penalty0\_egroup\_egroup
199   \_gdef\_layertext{\_settinglayer#2}%
200   \_global\_maxlayers=#1
201   \_setbox\slidepageB=\_vbox\_bgroup\_bgroup
202   \_setbox0=\_vbox{{\_layernum=1 \_globaldefs=-1 \_layertext\_endgraf}}\_prevdepth=\_dp0
203 }
204 \_public \subtit \slideshow \pg \wideformat \use \pshow \layernum ;

```

`\slideopen` should be used instead `\slideshow` to deactivate it but keep the borders of groups.

slides.opm

```
211 \def\slideopen{\let\slideshow=\relax % first wins
212 \sdef{spg:;}{\egroups\vfil\break \lfnotenumreset\bgroup \setilevel}
213 \sdef{spg:;}{\egroups\endslides}
214 \sdef{spg:;}{\egroups\bgroup \setilevel}
215 \let\layersopen=\egroup \let\layersclose=\bgroup
216 \bgroup
217 }
218 \public \slideopen ;
```

When `\slideshow` is active then the destinations of internal hyperlinks cannot be duplicated to more “virtual” pages because hyperlink destinations have to be unique in the whole document.

The `\slideshow` creates boxes of typesetting material and copies them to more pages. So, we have to suppress creating destinations in these boxes. This is done in the `\slidelinks` macro. We can move creating these destinations to the output routine. `\sdestbox` is saved value of the original `\destbox` which is redefined to do only `\addto\destboxes{\sdestbox[<label>]}`. All destinations saved to `\destboxes` are created at the start of the next output routine in the `\pagedest` macro. The output routine removes `\destboxes`, so each destination is created only once.

Limitations of this solution: destinations are only at the start of the page, no at the real place where `\wlabel` was used. The first “virtual” page where `\wlabel` is used includes its destination. If you want to go to the final page of the partially uncovering ideas then use `\label[<label>]\wlabel{text}` in the last part of the page (before `\pg;`) or use `\pgref` instead `\ref`.

slides.opm

```
243 \def\slidelinks{%
244 \def \destbox{##1}{\edef\tmp{\noexpand\sdestbox{##1}}%
245 \global\ea\addto\ea\destboxes\ea\tmp}%
246 \def \pagedest {%
247 \hbox{\def\destheight{25pt}\sdestbox[pg:\the\pgpageno]\destboxes}%
248 \nointerlineskip \gdef\destboxes{}}%
249 }%
250 \ifx \dest\destactive \else \let\pagedest=\relax \fi
251 }
252 \let\sdestbox = \destbox
253 \def\destboxes{} % initial value of \destboxes
254 \let\bibgl=\global % \advance\bibnum must be global if they are at more pages
```

The `\settinglayer` is used in the `\layertext` macro to prevent printing “Duplicate label” warning when it is expanded. It is done by special value of `\slideshowook` (used by the `\label` macro). Moreover, the warning about illegal use of `\bib`, `\usebib` in `\layers` environment is activated.

slides.opm

```
264 \def\settinglayer{%
265 \def\slideshowook ##1##2{%
266 \def\bibB{##1}{\nousebib}\def\usebib/##1 (##2) ##3 {\nousebib}%
267 }
268 \def\nousebib{\opwarning{Don't use \noexpand\bib nor \noexpand\usebib in \string\layers}}
```

Default `\layers <num>` macro (when `\slideshow` is not activated) is simple. It prints the *<layered-text>* with `\layernum=<num>+1` because we need the result after last layer is processed.

slides.opm

```
276 \long\def\layers #1 #2\endlayers{\par
277 \layersopen {\layernum=\numexpr#1+1\relax #2\endgraf}\layersclose}
278 \let\layersopen=\relax
279 \let\layersclose=\relax
280
281 \def\layers{\layers}
```

We must to redefine `\fnotenumpages` because the data from `.ref` file are less usable for implementing such a 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

```
291 \def \fnotenumpages {\def\fnotenum{\the\lfnotenum}\pgfnotefalse
292 \def\lfnotenumreset{\global\lfnotenum=0 }}
293 \let \lfnotenumreset=\relax
294 \public \fnotenumpages ;
```

## 2.36 Logos

```
3 \_codedecl \TeX {Logos TeX, LuaTeX, etc. <2020-02-28>} % preloaded in format
```

logos.opm

Despite plain  $\TeX$  each macro for logos ends by `\ignoreslash`. This macro ignores the 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.

```
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 {\_isnextchar/\_ignoreit{}}
20
21 \_public \TeX \OpTeX \LuaTeX \XeTeX \ignoreslash ;
```

logos.opm

The `\slantcorr` macro expands to the slant-correction of the current font. It is used to shifting A if the `\LaTeX` logo is in italic.

```
28 \_protected\def \LaTeX{\_tmpdim=.42ex L\_kern-.36em \_kern \_slantcorr % slant correction
29 \_raise \_tmpdim \_hbox{\_thefontscale[710]A}%
30 \_kern-.15em \_kern-\_slantcorr \_TeX}
31 \_def\_slantcorr{\_ea\_ignorept \_the\_fontdimen1\_font\_tmpdim}
32
33 \_public \LaTeX ;
```

logos.opm

`\OPmac`, `\CS` and `\csplain` logos.

```
39 \_def\_OPmac{\_leavevmode
40 \_lower.2ex\_hbox{\_thefontscale[1400]O}\_kern-.86em P{\_em mac}\_ignoreslash}
41 \_def\_CS{\$\_cal C$\_kern-.1667em\_lower.5ex\_hbox{\$\_cal S$\_ignoreslash}
42 \_def\_csplain{\_CS plain\_ignoreslash}
43
44 \_public \OPmac \CS \csplain ;
```

logos.opm

The expandable versions of logos used in Outlines need the expandable `\ignslash` (instead of the `\ignoreslash`).

```
51 \_def\_ignslash#1{\_ifx/#1\_else #1\_fi}
52 \_regmacro {}{}{% conversion for PDF outlines
53 \_def\TeX\TeX\_ignslash}\_def\OpTeX\OpTeX\_ignslash}%
54 \_def\LuaTeX\LuaTeX\_ignslash}\_def\XeTeX\XeTeX\_ignslash}%
55 \_def\LaTeX\LaTeX\_ignslash}\_def\OPmac\OPmac\_ignslash}%
56 \_def\CS\CS}\_def\csplain\csplain\_ignslash}%
57 }
58 \_public \ignslash ;
```

logos.opm

## 2.37 Multilingual support

### 2.37.1 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 `UnicodeData.txt` is read if this file exists in your  $\TeX$  distribution. The format is specified at <http://www.unicode.org/L2/L1999/UnicodeData.html>. We read only L1 (lowercase letters), Lu (uppercase letters) and Lo (other letters) and set appropriate codes. The scanner of `UnicodeData.txt` is implemented here in the group (lines 6 to 15). After the group is closed then the file `uni-lcuc.opm` is leaved by `\endinput`.

If the file `UnicodeData.txt` does not exist then internal data are used. They follow to the end of the file `uni-lcuc.opm`.



```

3 \_wterm{Setting lccodes and uccodes for Unicode characters <2021-04-07>} % preloaded in format.
4
5 \_isfile{UnicodeData.txt}\_iftrue
6 \_begingroup
7   \_sdef{lc:Ll}#1#2#3#4{\_global\_lccode"#2="#2 \_global\_uccode"#2="#0#3 }
8   \_sdef{lc:Lu}#1#2#3#4{\_global\_lccode"#2="#0#4 \_global\_uccode"#2="#2 }
9   \_sdef{lc:Lo}#1#2#3#4{\_global\_lccode"#2="#2 \_global\_uccode"#2="#2 }
10  \_def\_pa#1;#2;#3;#4;#5;#6;#7;#8;#9;{\_ifx#1;\_else\_ea\_pb\_fi{#1}{#3}}
11  \_def\_pb#1#2#3;#4;#5;#6;#7;#8 {\_csname lc:#2\_endcsname\_pc{#1}{#6}{#7}\_pa}
12  \_def\_pc#1#2#3{ } % ignored if the character hasn't Ll, Lu, nor Lo type
13  \_everyeof={;};};};};} % end of file
14  \_ea\_pa\_input UnicodeData.txt
15 \_endgroup \_endinput \_fi % \endinput here, if UnicodeData.txt was loaded
16
17 % If UnicodeData.txt not found, we have internal copy here from csplain, 2014:
18
19 \_def\_tmp #1 #2 {\_ifx#1\_else
20   \_lccode"#1="#1
21   \_ifx.#2%
22     \_uccode"#1="#1
23   \_else
24     \_uccode"#2="#2
25     \_lccode"#2="#1
26     \_uccode"#1="#2
27   \_fi
28   \_ea \_tmp \_fi
29 }
30 \_tmp
31 00AA .
32 00B5 039C
33 00BA .
34 00E0 00C0
35 00E1 00C1
36 00E2 00C2
37 00E3 00C3
38 00E4 00C4
39 00E5 00C5

```

...etc., 15900 similar lines (see [uni-lcuc.opm](#))

## 2.37.2 Hyphenations

```

3 \_codedecl \_langlist {Initialization of hyphenation patterns <2021-03-29>} % preloaded in format

```

The  $\langle iso-code \rangle$  means a shortcut of language name (mostly by ISO 639-1). The following control sequences are used for language switching:

- $\backslash\_lan:\langle number \rangle$  expands to  $\langle iso-code \rangle$  of the language. The  $\langle number \rangle$  is an internal number of languages used as a value of  $\backslash\_language$  register.
- $\backslash\_ulan:\langle long-lang \rangle$  expands to  $\langle iso-code \rangle$  too. This is transformation from long name of language (lowercase letters) to  $\langle iso-code \rangle$ .
- $\backslash\_ \langle iso-code \rangle Patt$  (for example  $\backslash\_csPatt$ ) is the language  $\langle number \rangle$  declared by  $\backslash\_chardef$ .
- $\backslash \langle iso-code \rangle lang$  (for example  $\backslash enlang$ ,  $\backslash slang$ ,  $\backslash sklang$ ,  $\backslash delang$ ,  $\backslash pllang$ ) is language selector. It exists in two states
  - Initialization state: when  $\backslash \langle iso-code \rangle lang$  is used first then it must load the patterns into memory using Lua code. If it is done then the  $\backslash \langle iso-code \rangle lang$  re-defines itself to the processing state.
  - Processing state: it only sets  $\backslash\_language=\backslash\_ \langle iso-code \rangle Patt$ , i.e it selects the hyphenation patterns. It does a little more language-dependent work, as mentioned below.
- $\backslash\_langspecific:\langle isocode \rangle$  is processed by  $\backslash \langle iso-code \rangle lang$  and it should include language-specific macros declared by the user or macro designer.

The USenglish patters 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

```

`\_preplang`  $\langle iso-code \rangle$   $\langle long-lang \rangle$   $\langle hyph-file-spec \rangle$   $\langle number \rangle$   $\langle pre-hyph \rangle$   $\langle post-hyph \rangle$  prepares the  $\langle iso-code \rangle$ lang to its initialization state. Roughly speaking, it does:

```

\_chardef{\_}  $\langle iso-code \rangle$ Patt =  $\langle number \rangle$ 
\_def{\_lan:}  $\langle number \rangle$  {  $\langle iso-code \rangle$  }
\_def{\_ulan:}  $\langle long-lang \rangle$  {  $\langle iso-code \rangle$  }
\_def{\_}  $\langle iso-code \rangle$ lang { %
  \_loadpattrs  $\langle hyph-file-spec \rangle$   $\langle number \rangle$   $\langle long-lang \rangle$  % loads patterns using Lua code
  \gdef{\_}  $\langle iso-code \rangle$ lang {\_uselang{  $\langle iso-code \rangle$  }\_}  $\langle iso-code \rangle$ Patt  $\langle pre-hyph \rangle$   $\langle post-hyph \rangle$  }
  \_}  $\langle iso-code \rangle$ lang % runs itself in processing state
}
\_def{\_}  $\langle iso-code \rangle$ lang {\_}  $\langle iso-code \rangle$ lang} % public version \_}  $\langle iso-code \rangle$ lang

```

You can see that  $\langle iso-code \rangle$ lang runs `\_loadpattrs` and `\_uselang` first (in initialization state) and it runs only `\_uselang` when it is called again (in processing state).

hyphen-lan.opm

```

64 \_def{\_preplang #1 #2 #3 #4 #5 { %
65   \_ea\_chardef{\_} \_csname\_ #1Patt\_endcsname=#4
66   \_sdef{\_lan:#4}{#1}\_lowercase{\_sdef{\_ulan:#2}}{#1}%
67   \_def{\_next{\_} \_noexpand\_} \_csname\_ #1lang\_endcsname}%
68   \_ea\_edef{\_} \_csname\_ #1lang\_endcsname { %
69     \_noexpand\_loadpattrs #3 #4 #2 % loads patterns
70     \_gdef{\_next{\_} \_noexpand\_uselang{#1}\_} \_csname\_ #1Patt\_endcsname #5}% re-defines itself
71     \_next % runs itself in processing state
72   }
73   \_addto{\_langlist{ #1(#2)}}%
74   \_sdef{#1lang\_} \_ea{\_} \_csname\_ #1lang\_endcsname}% unprefix \_isocode>lang
75 }

```

`\_loadpattrs`  $\langle hyph-file-spec \rangle$   $\langle number \rangle$   $\langle long-lang \rangle$  loads hyphenation patterns and hyphenation exceptions for given language and registers them as `\_language=` $\langle number \rangle$ .

The  $\langle hyph-file-spec \rangle$  is a part of full file name which is read: `hyph- $\langle hyph-file-spec \rangle$ .tex`. The patterns and hyphenation exceptions are saved here in UTF-8 encoding. The  $\langle hyph-file-spec \rangle$  should be a list of individual  $\langle hyph-file-spec \rangle$ 's separated by commas, see the language Serbian below for an example.

hyphen-lan.opm

```

89 \_def{\_loadpattrs#1 #2 #3 { %
90   \_wlog{Loading hyphenation #3: (#1) \_string{language=#2}%
91   \_begingroup\_setbox0=\_vbox{ % we don't want spaces in horizontal mode
92     \_language=#2\_def{\_} {#3}%
93     \_let{patterns}=\_patterns \_let{hyphenation}=\_hyphenation \_def{message##1}{ %
94       \_loadpattrsA #1,, %
95     } \_endgroup
96   }
97   \_def{\_loadpattrsA #1,{\_ifx,#1,\_else
98     \_isfile{hyph-#1}\_iftrue \_opinput{hyph-#1}%
99     \_else \_opwarning{No hyph. patterns #1 for \_, missing package?}%
100     \_def{\_opwarning##1}{\_fi
101     \_ea \_loadpattrsA \_fi
102   }

```

`\_uselang`  $\langle iso-code \rangle$   $\langle iso-code \rangle$ Patt  $\langle pre-hyph \rangle$   $\langle post-hyph \rangle$

sets `\_language`, `\lefthyphenmin`, `\righthyphenmin` and runs `\frenchspacing`. This default language-dependent settings should be re-declared by `\_langspecific:  $\langle iso-code \rangle$`  which is run finally (it is `\_relax` by default, only `\_langspecific:en` runs `\nonfrenchspacing`).

hyphen-lan.opm

```

113 \_def{\_uselang#1#2#3#4{\_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\_relax
114   \_frenchspacing % \nonfrenchspacing can be set in \_cs{\_langspecific:lan}
115   \_cs{\_langspecific:#1}%
116 }

```

The `\uselanguage`  $\{\langle long-lang \rangle\}$  is defined here (for compatibility with e-plain users).

hyphen-lan.opm

```
122 \_def\_uselanguage#1{\_lowercase{\_cs{\_cs{\_ulan:#1}lang}}}  
123 \_public \uselanguage ;
```

The numbers for languages are declared as fixed constants (no auto-generated). This concept is inspired by 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

```
133 \_preplang enus USenglishmax en-us 100 23  
134 \_preplang engb UKenglish en-gb 101 23  
135 \_preplang it Italian it 102 22  
136 \_preplang ia Interlingua ia 103 22  
137 \_preplang id Indonesian id 104 22  
138  
139 \_preplang cs Czech cs 115 23  
140 \_preplang sk Slovak sk 116 23  
141 \_preplang de nGerman de-1996 121 22  
142 \_preplang fr French fr 122 22  
143 \_preplang pl Polish pl 123 22  
144 \_preplang cy Welsh cy 124 23  
145 \_preplang da Danish da 125 22  
146 \_preplang es Spanish es 126 22  
147 \_preplang sl Slovenian sl 128 22  
148 \_preplang fi Finnish fi 129 22  
149 \_preplang hu Hungarian hu 130 22  
150 \_preplang tr Turkish tr 131 22  
151 \_preplang et Estonian et 132 23  
152 \_preplang eu Basque eu 133 22  
153 \_preplang ga Irish ga 134 23  
154 \_preplang nb Bokmal nb 135 22  
155 \_preplang nn Nynorsk nn 136 22  
156 \_preplang nl Dutch nl 137 22  
157 \_preplang pt Portuguese pt 138 23  
158 \_preplang ro Romanian ro 139 22  
159 \_preplang hr Croatian hr 140 22  
160 \_preplang zh Pinyin zh-latn-pinyin 141 11  
161 \_preplang is Icelandic is 142 22  
162 \_preplang hsb Uppersorbian hsb 143 22  
163 \_preplang af Afrikaans af 144 12  
164 \_preplang gl Galician gl 145 22  
165 \_preplang kmr Kurmanji kmr 146 22  
166 \_preplang tk Turkmen tk 147 22  
167 \_preplang la Latin la 148 22  
168 \_preplang lac classicLatin la-x-classic 149 22  
169 \_preplang lal liturgicalLatin la-x-liturgic 150 22  
170 \_preplang elm monoGreek el-monoton 201 11  
171 \_preplang elp Greek el-polyton 202 11  
172 \_preplang grc ancientGreek grc 203 11  
173 \_preplang ca Catalan ca 204 22  
174 \_preplang cop Coptic cop 205 11  
175 \_preplang mn Mongolian mn-cyrl 206 22  
176 \_preplang sa Sanskrit sa 207 13  
177 \_preplang ru Russian ru 208 22  
178 \_preplang uk Ukrainian uk 209 22  
179 \_preplang hy Armenian hy 210 12  
180 \_preplang as Assamese as 211 11  
181 \_preplang hi Hindi hi 212 11  
182 \_preplang kn Kannada kn 213 11  
183 \_preplang lv Latvian lv 215 22  
184 \_preplang lt Lithuanian lt 216 22  
185 \_preplang ml Malayalam ml 217 11  
186 \_preplang mr Marathi mr 218 11  
187 \_preplang or Oriya or 219 11  
188 \_preplang pa Panjabi pa 220 11  
189 \_preplang ta Tamil ta 221 11  
190 \_preplang te Telugu te 222 11  
191
```

192	<code>\_preplang</code>	<code>be</code>	Belarusian	<code>be</code>	223	22
193	<code>\_preplang</code>	<code>bg</code>	Bulgarian	<code>bg</code>	224	22
194	<code>\_preplang</code>	<code>bn</code>	Bengali	<code>bn</code>	225	11
195	<code>\_preplang</code>	<code>cu</code>	churchslavonic	<code>cu</code>	226	12
196	<code>\_preplang</code>	<code>deo</code>	oldGerman	<code>de-1901</code>	227	22
197	<code>\_preplang</code>	<code>gsw</code>	swissGerman	<code>de-ch-1901</code>	228	22
198	<code>\_preplang</code>	<code>eo</code>	Esperanto	<code>eo</code>	229	22
199	<code>\_preplang</code>	<code>fur</code>	Friulan	<code>fur</code>	230	22
200	<code>\_preplang</code>	<code>gu</code>	Gujarati	<code>gu</code>	231	11
201	<code>\_preplang</code>	<code>ka</code>	Georgian	<code>ka</code>	232	12
202	<code>\_preplang</code>	<code>mk</code>	Macedonian	<code>mk</code>	233	22
203	<code>\_preplang</code>	<code>oc</code>	Occitan	<code>oc</code>	234	22
204	<code>\_preplang</code>	<code>pi</code>	Pali	<code>pi</code>	235	12
205	<code>\_preplang</code>	<code>pms</code>	Piedmontese	<code>pms</code>	236	22
206	<code>\_preplang</code>	<code>rm</code>	Romansh	<code>rm</code>	237	22
207	<code>\_preplang</code>	<code>sr</code>	Serbian	<code>sh-cyrl,sh-latn</code>	238	22
208	<code>\_preplang</code>	<code>sv</code>	Swedish	<code>sv</code>	239	22
209	<code>\_preplang</code>	<code>th</code>	Thai	<code>th</code>	240	23
210	<code>\_preplang</code>	<code>ethi</code>	Ethiopic	<code>mul-ethi</code>	241	11
211	<code>\_preplang</code>	<code>fis</code>	schoolFinnish	<code>fi-x-school</code>	242	11

The `\langlist` includes names of all languages which are ready to load and use their hyphenation patterns. This list is printed to the terminal and to log at iniTeX state here. It can be used when processing documents too.

hyphen-lan.opm

```
219 \message{Language hyph.patterns ready to load: \_langlist.
220   Use \_string\<shortname>lang to initialize language,
221   \_string\cslang\_space for example}
222
223 \_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 the selected language, you can define 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 set language-specific phrases (like `\today`) by this code. Another concept is used for such tasks. See the section 2.37.3 for more details.

### 2.37.3 Multilingual phrases and quotation marks

languages.opm

```
3 \_codedecl \_mtext {Languages <2021-01-21>} % 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 the declared word and `\<lang>` is a 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}{Tabulka}
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:

```

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 Chaptire Tableau Figure Matière
41 \_langw it Capitolo Tabella Fig. Oggetto
42 \_langw pl Rozdział Tabela Ilustracja Temat

```

...etc. (see `languages.opm`)

You can add more words as you wish. For example `\today` macro:

```

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}{#5}\_sdef{mt:m6:#1}{#5}%
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}{#5}\_sdef{mt:m12:#1}{#5}%
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 \_monthw it gennaio febbraio marzo aprile maggio giugno
68             luglio agosto settembre ottobre novembre dicembre
69
70
71 \_sdef{mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
72 \_sdef{mt:today:cs}{\_mtext{m\_the\_month} \_the\_year}
73 \_slet{mt:today:sk}{\_mt:today:cs}
74
75 \_def \_today{\_mtext{today}}
76 \_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 the `\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 the meaning of these two types of quotes.

`\enquotes`, `\csquotes`, `\dequotes`, `\frquotes` etc. are defined here.

```

93 \_def \_enquotes {\_quoteschars ""' }
94 \_def \_csquotes {\_quoteschars " , ' }
95 \_def \_frquotes {\_quoteschars "«» }
96 \_let \_plquotes = \_frquotes
97 \_let \_esquotes = \_frquotes
98 \_let \_grquotes = \_frquotes
99 \_let \_ruquotes = \_frquotes
100 \_let \_itquotes = \_frquotes
101 \_let \_skquotes = \_csquotes
102 \_let \_dequotes = \_csquotes

```

The `\quoteschars<lqq><rqq><lq><rq>` defines `\"` and `\'` as `\_qqA` in in normal mode and as expandable macros in outline mode. We want to well process the common cases: `\`&`` or `\`~{``. This is the reason why the quotes parameter is read in verbatim mode and retokenized again by `\scantextokens`. We want to allow to quote the quotes mark itself by `\`~"~`. This is the reason why the sub-verbatim

mode is used when the first character is { in the parameter.

The `\` is defined as `\_qqA\_qqB⟨lq⟩⟨rq⟩` and `\` as `\_qqA\_qqC⟨lq⟩⟨rq⟩`. The `\_qqA\_qqB⟨clq⟩⟨crq⟩` runs `\_qqB⟨lq⟩⟨rq⟩⟨text⟩`.

The `\_regquotes\`""`⟨L⟩⟨R⟩` does `\def\#1"\{⟨L⟩#1⟨R⟩}` for outlines but the `"` separator is active (because `"` and `'` are active in `\pdfunidef`).

```

118 \_def \_quoteschars #1#2#3#4{\_def\_altquotes{\_quoteschars#3#4#1#2}\_public\_altquotes;%
119 \_protected\_def \{"\_qqA\_qqB#1#2}\_protected\_def \{'\_qqA\_qqC#3#4}%
120 \_regmacro{}{}{\_regquotes\""#1#2\_regquotes\''#3#4}}
121
122 \_def\_qqA#1#2#3{\_bgroup\_setverb \_catcode\ =10
123 \_isnextchar\_bgroup{\_catcode\={1 \_catcode\}=2 #1#2#3}{#1#2#3}}
124 \_long\_def\_qqB#1#2#3'\_egroup#1\_scantextokens{#3#2}
125 \_long\_def\_qqC#1#2#3'\_egroup#1\_scantextokens{#3#2}
126 \_def\_regquotes#1#2#3#4{\_bgroup \_lccode\~=#2\_lowercase{\_egroup \_def#1#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 the first backslash. First, declare `\⟨iso-code⟩quotes`, then `\altquotes` (if needed) and finally `\activequotes`.

```

136 \_def\_activequotes{\_let\_actqq=\\_adef"\_actqq\_let\_actq=\'\_adef'\_actq}%
137 \_regmacro{}{}{\_adef"\_actq\_adef'\_actq}}
138
139 \_public \_quoteschars \activequotes \enquotes \csquotes \skquotes \frquotes \plquotes
140 \esquotes \grquotes \ruquotes \itquotes \dequotes ;

```

Bibliography references generated by `\usebib` uses more language-dependent phrases. They are declared here. We don't want to save all these phrases into the format, so the trick with `\_endinput` is used here. When `\usebib` is processed then the following part of the file `languages.opm` is read again.

Only phrases of few languages are declared here now. If you want to declare phrases of your language, please create an "issue" or a "request" at <https://github.com/olsak/OpTeX> or send me an email with new phrases for your language (or language you know:). I am ready to put them here. Temporarily, you can put your definitions into `\bibtexhook` token list.

```

156 \_endinput % don't save these \def's to the format
157
158 \_def\_langb#1 #2#3#4#5#6#7#8#9{\_def\_mbib#1##2{\_sdef{\_mt:bib.##2:#1}{##1}}%
159 \_mbib{#2}{and} \_mbib{#3}{etal} \_mbib{#4}{edition} \_mbib{#5}{citedate} \_mbib{#6}{volume}%
160 \_mbib{#7}{number} \_mbib{#8}{prepares} \_mbib{#9}{postpages} \_langbA}
161 \_def\_langbA#1#2#3#4#5#6#7{\_mbib{#1}{editor} \_mbib{#2}{editors} \_mbib{#3}{available}%
162 \_mbib{#4}{availablealso} \_mbib{#5}{bachthesis} \_mbib{#6}{mastthesis} \_mbib{#7}{phdthesis}}
163
164 \_langb en {, and } { et al.} { ed.} {cit.~} {Vol.~} {No.~} {pp.~} {~p.} {,~ed.} {,~eds.}
165 {Available from } {Available also from }
166 {Bachelor's Thesis} {Master's Thesis} {Ph.D. Thesis}
167 %-----
168 \_langb cs { a } { a~kol.} { vyd.} {vid.~} {ročník~} {č.~} {s.~} {~s.} {,~editor} {,~editoři}
169 {Dostupné na } {Dostupné též na }
170 {Bakalářská práce} {Diplomová práce} {Disertační práce}
171 \_langb sk { a } { a~kol.} { vyd.} {vid.~} {ročník~} {č.~} {s.~} {~s.} {,~editor} {,~editoři}
172 {Dostupné na } {Dostupné tiež na }
173 {Bakalárska práca} {Diplomová práca} {Dizertačná práca}
174
175 % \_<lang>dateformat year/month/day\relax, for example: \_csdateformat 2020/05/21\relax
176 % This is used in iso690 bib-style when the field "citedate" is used.
177
178 \_def\_endateformat #1/#2/#3\relax{#1-#2-#3}
179 % \_csdateformat 2020/05/21\relax -> \hbox{21. 5. 2020}
180 \_def\_csdateformat #1/#2/#3\relax{\hbox{\_tmpnum=#3 \_the\_tmpnum. \_tmpnum=#2 \_the\_tmpnum. #1}}
181 \_let\_skdateformat =\_csdateformat

```

## 2.38 Other macros

Miscellaneous macros are here.

```

3 \_codedecl \uv {Miscellaneous <2020-05-22>} % 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\_ignoresecond\_currpage}
17 \def\_lastpage {\_openref\_ea\_usessecond\_currpage}
18 \def\_currpage {{0}{?}}
19 \_public \lastpage \totalpages ;
```

We need `\uv`, `\clqq`, `\crqq`, `\flqq`, `\frqq`, `\uslang`, `\ehyph` `\chyph`, `\shyph`, for backward compatibility with  $\mathcal{E}$ plain. Codes are set according to Unicode because we are using Czech only in Unicode when Lua $\mathcal{T}$  $\mathcal{E}$ X is used.

others.opm

```
28
29 % for compatibility with csplain:
30
31 \_chardef\clqq=8222 \_chardef\crqq=8220
32 \_chardef\flqq=171 \_chardef\frqq=187
33 \_chardef\promile=8240
34
35 \_def\uv#1{\clqq#1\crqq}
36
37 \_let\uslang=\enlang \_let\ehyph=\enlang
38 \_let\chyph=\cslang \_let\shyph=\sklang
39 \_let\csUnicode=\csPatt \_let\czUnicode=\csPatt \_let\skUnicode=\skPatt
```

The `\letfont` was used in  $\mathcal{E}$ plain instead of `\fontlet`.

others.opm

```
45 \_let \letfont = \fontlet
```

Non-breaking space in Unicode.

others.opm

```
51 \let ^^a0=~
```

TikZ needs these funny control sequences.

others.opm

```
57 \_ea\_toksdef \_csname toks@\_endcsname=0
58 \_ea\_let \_csname voidb@x\_endcsname=\_voidbox
```

We don't want to read `opmac.tex` unless `\input opmac` is specified.

others.opm

```
64 \_def\OPmacversion{OpTeX}
```

We allow empty lines in math formulae. It is more comfortable.

others.opm

```
70 \_suppressmathparerror = 1
```

Lorem ipsum can be printed by `\lipsum[⟨range⟩]` or `\lorem[⟨range⟩]`, for example `\lipsum[3]` or `\lipsum[112-121]`, `max=150`.

First usage of `\lipsum` reads the  $\mathcal{L}$  $\mathcal{T}$  $\mathcal{E}$ X file `lipsum.ltd.tex` by `\_lipsumload` and prints the selected paragraph(s). Next usages of `\lipsum` prints the selected paragraph(s) from memory. This second and more usages of `\lipsum` are fully expandable. If you want to have all printings of `\lipsum` expandable, use dummy `\lipsum[0]` first.

`\lipsum` adds `\par` after each printed paragraph. If you don't need such `\par` here, use `\lipsumtext[⟨number⟩]`. This macro prints only one selected paragraph `⟨number⟩` and does not add `\par`.

others.opm

```
88 \_def\_lipsumtext[#1]{\_lipsumload\_cs{\_lip:#1}}
89 \_def\_lipsum[#1]{\_lipsumA #1\_empty-\_empty\_end}
90 \_def\_lipsumA #1-#2\_empty#3\_end{%
91   \_for num #1..\_ifx~#2~#1\_else#2\_fi \_do {\_lipsumtext[#1]\par}}
92 \_def\_lipsumload{%
93   \_setbox0=\_vbox{\_tmpnum=0 % vertical mode during \input lipsum.ltd.tex
94     \_def\ProvidesFile##1[##2]{}%
95     \_def\NewLipsumPar{\_incr\_tmpnum \_sxdef{\_lip:\_the\_tmpnum}}%
96     \_opinput {lipsum.ltd.tex}%
97     \_global\_let\_lipsumload=\_empty
98   }
99 \_public \lipsum \lipsumtext ;
100 \_let \lorem=\lipsum
```

## 2.39 Lua code embedded to the format

The file `optex.lua` is loaded into the format in `optex.ini` as byte-code and initialized by `\everyjob`, see section 2.1.

The file implements part of the functionality from `luatexbase` namespace, nowadays defined by `TeX` kernel. `luatexbase` deals with modules, allocators, and callback management. Callback management is a nice extension and is actually used in `OpTeX`. Other functions are defined more or less just to suit `luaotfload`'s use.

`optex.lua`

```
4
```

### GENERAL

```
6
```

Error function used by following functions for critical errors.

```
8 local function err(message)
9     error("\nerror: " .. message .. "\n")
10 end
```

For a `\chardef`'d, `\countdef`'d, etc., `csname` return corresponding register number. The responsibility of providing a `\XXdef`'d name is on the caller.

```
14 function registernumber(name)
15     return token.create(name).index
16 end
```

### ALLOCATORS

```
19 alloc = alloc or {}
```

An attribute allocator in Lua that cooperates with normal `OpTeX` allocator.

```
22 local attributes = {}
23 function alloc.new_attribute(name)
24     local cnt = tex.count["_attributealloc"] + 1
25     if cnt > 65534 then
26         tex.error("No room for a new attribute")
27     else
28         tex.setcount("global", "_attributealloc", cnt)
29         texio.write_nl("log", "'..'..name..'='\@attribute'..tostring(cnt))
30         attributes[name] = cnt
31         return cnt
32     end
33 end
```

`provides_module` is needed by older version of `luaotfload`

```
36 provides_module = function() end
```

### CALLBACKS

```
39 callback = callback or {}
```

Save `callback.register` function for internal use.

```
42 local callback_register = callback.register
43 function callback.register(name, fn)
44     err("direct registering of callbacks is forbidden, use 'callback.add_to_callback'")
45 end
```

Table with lists of functions for different callbacks.

```
48 local callback_functions = {}
```

Table that maps callback name to a list of descriptions of its added functions. The order corresponds with `callback_functions`.

```
51 local callback_description = {}
```

Table used to differentiate user callbacks from standard callbacks. Contains user callbacks as keys.

```
55 local user_callbacks = {}
```

Table containing default functions for callbacks, which are called if either a user created callback is defined, but doesn't have added functions or for standard callbacks that are “extended” (see `mlist_to_hlist` and its pre/post filters below).

```
60 local default_functions = {}
```

Table that maps standard (and later user) callback names to their types.

```
63 local callback_types = {
64   -- file discovery
65   find_read_file      = "exclusive",
66   find_write_file     = "exclusive",
67   find_font_file      = "data",
68   find_output_file    = "data",
69   find_format_file    = "data",
70   find_vf_file        = "data",
71   find_map_file       = "data",
72   find_enc_file       = "data",
73   find_pk_file        = "data",
74   find_data_file      = "data",
75   find_opentype_file  = "data",
76   find_truetype_file  = "data",
77   find_type1_file     = "data",
78   find_image_file     = "data",
79
80   open_read_file      = "exclusive",
81   read_font_file      = "exclusive",
82   read_vf_file        = "exclusive",
83   read_map_file       = "exclusive",
84   read_enc_file       = "exclusive",
85   read_pk_file        = "exclusive",
86   read_data_file      = "exclusive",
87   read_truetype_file  = "exclusive",
88   read_type1_file     = "exclusive",
89   read_opentype_file  = "exclusive",
90
91   -- data processing
92   process_input_buffer = "data",
93   process_output_buffer = "data",
94   process_jobname      = "data",
95   input_level_string   = "data",
96
97   -- node list processing
98   contribute_filter    = "simple",
99   buildpage_filter     = "simple",
100  build_page_insert    = "exclusive",
101  pre_linebreak_filter  = "list",
102  linebreak_filter     = "exclusive",
103  append_to_vlist_filter = "exclusive",
104  post_linebreak_filter = "reverselist",
105  hpack_filter         = "list",
106  vpack_filter         = "list",
107  hpack_quality        = "list",
108  vpack_quality        = "list",
109  process_rule         = "exclusive",
110  pre_output_filter    = "list",
111  hyphenate            = "simple",
112  ligaturing           = "simple",
113  kerning              = "simple",
114  insert_local_par     = "simple",
115  mlist_to_hlist       = "exclusive",
116
117  -- information reporting
118  pre_dump             = "simple",
119  start_run            = "simple",
120  stop_run             = "simple",
121  start_page_number    = "simple",
122  stop_page_number     = "simple",
123  show_error_hook      = "simple",
124  show_error_message   = "simple",
```

```

125 show_lua_error_hook = "simple",
126 start_file          = "simple",
127 stop_file           = "simple",
128 call_edit           = "simple",
129 finish_synctex      = "simple",
130 wrapup_run          = "simple",
131
132 -- pdf related
133 finish_pdffile       = "data",
134 finish_pdfpage       = "data",
135 page_order_index     = "data",
136 process_pdf_image_content = "data",
137
138 -- font related
139 define_font          = "exclusive",
140 glyph_not_found      = "exclusive",
141 glyph_info           = "exclusive",
142
143 -- undocumented
144 glyph_stream_provider = "exclusive",
145 provide_charproc_data = "exclusive",
146 }

```

Return a list containing descriptions of added callback functions for specific callback.

```

150 function callback.callback_descriptions(name)
151     return callback_description[name] or {}
152 end
153
154 local valid_callback_types = {
155     exclusive = true,
156     simple = true,
157     data = true,
158     list = true,
159     reverselist = true,
160 }

```

Create a user callback that can only be called manually using `call_callback`. A default function is only needed by "exclusive" callbacks.

```

164 function callback.create_callback(name, cbtype, default)
165     if callback_types[name] then
166         err("cannot create callback '"..name.."' - it already exists")
167     elseif not valid_callback_types[cbtype] then
168         err("cannot create callback '"..name.."' with invalid callback type '"..cbtype.."'")
169     elseif cbtype == "exclusive" and not default then
170         err("unable to create exclusive callback '"..name.."', default function is required")
171     end
172
173     callback_types[name] = cbtype
174     default_functions[name] = default or nil
175     user_callbacks[name] = true
176 end

```

Add a function to the list of functions executed when callback is called. For standard luatex callback a proxy function that calls our machinery is registered as the real callback function. This doesn't happen for user callbacks, that are called manually by user using `call_callback` or for standard callbacks that have default functions – like `mlist_to_hlist` (see below).

```

184 function callback.add_to_callback(name, fn, description)
185     if user_callbacks[name] or callback_functions[name] or default_functions[name] then
186         -- either:
187         -- a) user callback - no need to register anything
188         -- b) standard callback that has already been registered
189         -- c) standard callback with default function registered separately
190         -- (mlist_to_hlist)
191     elseif callback_types[name] then
192         -- This is a standard luatex callback with first function being added,
193         -- register a proxy function as a real callback. Assert, so we know
194         -- when things break, like when callbacks get redefined by future

```

```

195     -- luatex.
196     assert(callback_register(name, function(...)
197         return callback.call_callback(name, ...)
198     end))
199 else
200     err("cannot add to callback '"..name.."'" - no such callback exists")
201 end
202
203 -- add function to callback list for this callback
204 callback_functions[name] = callback_functions[name] or {}
205 table.insert(callback_functions[name], fn)
206
207 -- add description to description list
208 callback_description[name] = callback_description[name] or {}
209 table.insert(callback_description[name], description)
210 end

```

Remove a function from the list of functions executed when callback is called. If last function in the list is removed delete the list entirely.

```

214 function callback.remove_from_callback(name, description)
215     local descriptions = callback_description[name]
216     local index
217     for i, desc in ipairs(descriptions) do
218         if desc == description then
219             index = i
220             break
221         end
222     end
223
224     table.remove(descriptions, index)
225     local fn = table.remove(callback_functions[name], index)
226
227     if #descriptions == 0 then
228         -- Delete the list entirely to allow easy checking of "truthiness".
229         callback_functions[name] = nil
230
231         if not user_callbacks[name] and not default_functions[name] then
232             -- this is a standard callback with no added functions and no
233             -- default function (i.e. not mlist_to_hlist), restore standard
234             -- behaviour by unregistering.
235             callback_register(name, nil)
236         end
237     end
238
239     return fn, description
240 end

```

helper iterator generator for iterating over reverselist callback functions

```

243 local function reverse_ipairs(t)
244     local i, n = #t + 1, 1
245     return function()
246         i = i - 1
247         if i >= n then
248             return i, t[i]
249         end
250     end
251 end

```

Call all functions added to callback. This function handles standard callbacks as well as user created callbacks. It can happen that this function is called when no functions were added to callback – like for user created callbacks or `mlist_to_hlist` (see below), these are handled either by a default function (like for `mlist_to_hlist` and those user created callbacks that set a default function) or by doing nothing for empty function list.

```

260 function callback.call_callback(name, ...)
261     local cbtype = callback_types[name]
262     -- either take added functions or the default function if there is one
263     local functions = callback_functions[name] or {default_functions[name]}

```

```

264
265     if cbtype == nil then
266         err("cannot call callback '"..name.."'" - no such callback exists")
267     elseif cbtype == "exclusive" then
268         -- only one function, atleast default function is guaranteed by
269         -- create_callback
270         return functions[1](...)
271     elseif cbtype == "simple" then
272         -- call all functions one after another, no passing of data
273         for _, fn in ipairs(functions) do
274             fn(...)
275         end
276         return
277     elseif cbtype == "data" then
278         -- pass data (first argument) from one function to other, while keeping
279         -- other arguments
280         local data = (...)
281         for _, fn in ipairs(functions) do
282             data = fn(data, select(2, ...))
283         end
284         return data
285     end
286
287     -- list and reverselist are like data, but "true" keeps data (head node)
288     -- unchanged and "false" ends the chain immediately
289     local iter
290     if cbtype == "list" then
291         iter = ipairs
292     elseif cbtype == "reverselist" then
293         iter = reverse_ipairs
294     end
295
296     local head = (...)
297     local new_head
298     local changed = false
299     for _, fn in iter(functions) do
300         new_head = fn(head, select(2, ...))
301         if new_head == false then
302             return false
303         elseif new_head ~= true then
304             head = new_head
305             changed = true
306         end
307     end
308     return not changed or head
309 end

```

Create “virtual” callbacks `pre/post_mlist_to_hlist_filter` by setting `mlist_to_hlist` callback. The default behaviour of `mlist_to_hlist` is kept by using a default function, but it can still be overridden by using `add_to_callback`.

```

315 default_functions["mlist_to_hlist"] = node.mlist_to_hlist
316 callback.create_callback("pre_mlist_to_hlist_filter", "list")
317 callback.create_callback("post_mlist_to_hlist_filter", "reverselist")
318 callback_register("mlist_to_hlist", function(head, ...)
319     -- pre_mlist_to_hlist_filter
320     local new_head = callback.call_callback("pre_mlist_to_hlist_filter", head, ...)
321     if new_head == false then
322         node.flush_list(head)
323         return nil
324     elseif new_head ~= true then
325         head = new_head
326     end
327
328     -- mlist_to_hlist means either added functions or standard luatex behavior
329     -- of node.mlist_to_hlist (handled by default function)
330     head = callback.call_callback("mlist_to_hlist", head, ...)
331
332     -- post_mlist_to_hlist_filter

```



```

333     new_head = callback.call_callback("post_mlist_to_hlist_filter", head, ...)
334     if new_head == false then
335         node.flush_list(head)
336         return nil
337     elseif new_head ~= true then
338         head = new_head
339     end
340     return head
341 end)

```

Compatibility with L<sup>A</sup>T<sub>E</sub>X through luatexbase namespace. Needed for luaotfload.

```

345 luatexbase = {
346     registernumber = registernumber,
347     attributes = attributes,
348     provides_module = provides_module,
349     new_attribute = alloc.new_attribute,
350     callback_descriptions = callback.callback_descriptions,
351     create_callback = callback.create_callback,
352     add_to_callback = callback.add_to_callback,
353     remove_from_callback = callback.remove_from_callback,
354     call_callback = callback.call_callback,
355     callbacktypes = {}
356 }

```

## 2.40 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 a 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`. It is set to almost infinity (100000). You can set it to a more sensible value. Such a setting is valid only for the first following listing mode.

When the listing mode is finished by `\_doc` then the 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 compensation for this indentation.

The `\_cod` macro ignores the rest of the 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 an active link to the main documentation point of such control sequence and prints them in blue. Another 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 a hyperlink to the user documentation point if this point exists. Finally, the `\~<sequence>`` (for example `\~foo``) are hyperlinks to the user documentation point.

doc.opm

```

3 \_codedecl \printdoc {Macros for documentation printing <2020-04-28>}

```

General decalarations.

doc.opm

```

9 \_fontfam[lmfonts]
10 \_hyperlinks \Green \Green
11 \_enlang
12 \_enquotes

```

Maybe, somebody needs `\seccc` or `\secccc`?

doc.opm

```
18 \_eoldef\seccc#1{\_medskip \_noindent{\_bf#1}\_par\_nobreak\_firstnoindent}
19 \_def\secccc{\_medskip \_noindent $\_bullet$ }
```

`\enddocument` can be redefined.

doc.opm

```
25 \_let\enddocument=\_bye
```

A full page of listing causes underfull `\vbox` in output routine. We need to add a small tolerance.

doc.opm

```
32 \_pgbottomskip=0pt plus10pt minus2pt
```

The listing mode is implemented here. The `\maxlines` is maximal lines of code printed in the listing mode.

doc.opm

```
39 \_newcount \_maxlines \_maxlines=100000
40 \_public \_maxlines ;
41
42 \_eoldef\_cod#1{\_par \_wipeepar
43 \_vskip\_parskip \_medskip \_ttskip
44 \_begingroup
45 \_typo[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.

doc.opm

```
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 }
99 \_def\_remfirstunderscore#1{\_ea\_remfirstunderscoreA#1\_relax#1}
100 \_def\_remfirstunderscoreA#1#2\_relax#3{\_if\_#1\_def#3{#2}\_fi}

```

The lines in the listing mode have a yellow background.

doc.opm

```

106 \_def\Yellow{\_setcmykcolor{0.0 0.0 0.3 0.03}}
107
108 \_def\_printcodeline#1{\_advance \_maxlines by-1
109 \_ifnum \_maxlines<0 \_ea \_endverbprinting \_fi
110 \_ifx\_printfilename\_relax \_penalty \_ttpenalty \_fi \_vskip-4pt
111 \_noindent\_rlap{\Yellow \_vrule height8pt depth5pt width\_hsize}%
112 \_printfilename
113 \_indent \_printverblinenum #1\par}
114
115 \_def\_printfilename{\_hbox to0pt{%
116 \_hskip\_hsize\_vbox to0pt{\_vss\_llap{\Brown\docfile}\_kern7.5pt}\_hss}%
117 \_let\_printfilename=\_relax
118 }
119 \_everytt={\_let\_printverblinenum=\_relax}
120
121 \_long\_def\_endverbprinting#1\_end#2\_end{\_fi\_fi \_global\_maxlines=100000
122 \_noindent\_typsize[8/]\_dots etc. (see {\_tt\Brown\docfile})}

```

\docfile is currently documented file.

\printdoc and \printdoctail macros are defined here.

doc.opm

```

129 \_def\docfile{}
130 \_def\_printdoc #1 {\_par \_def\docfile{#1}%
131 \_everytt={\_ttshift=-15pt \_let\_printverblinenum=\_relax}%
132 \_ea\_cod \input #1
133 \_everytt={\_let\_printverblinenum=\_relax}%
134 \_def\docfile{}%
135 }
136 \_def\_printdoctail #1 {\_bgroup
137 \_everytt={\_ttline=-1 \_ea\_printdoctailA \_input #1 \_egroup}
138 {\_long\_gdef\_printdoctailA#1\_endcode{}}
139
140 \_public \printdoc \printdoctail ;

```

You can do \verbinuput \vitt{<filename>} (<from>-<to>) <filename> if you need analogical design like in listing mode.

doc.opm

```

147 \_def\_vitt#1{\_def\docfile{#1}\_ttline=-1
148 \_everytt={\_typsize[8/10]\_let\_printverblinenum=\_printcodeline \_medskip}}
149
150 \_public \vitt ;

```

The Index entries are without the trailing backslash. We must add it when printing Index.

doc.opm

```

157 \_addto \_ignoredcharsen {\_} % \foo, \_foo is the same in the first pass of sorting
158 \_def\_printii #1#2&{%
159 \_ismacro\_lastii{#1}\_iffalse \_newiiletter{#1}{#2}\_def\_lastii{#1}\_fi
160 \_gdef\_currii{#1#2}\_the\_everyii\_noindent
161 \_hskip-\_iindent \_ignorespaces\_printiiA\bslash#1#2//}
162
163 \_def\_printiipages#1&{\_let\_pgtype=\_undefined \_tmpnum=0
164 {\_rm\_printpages #1,.,\_par}}
165
166 \_sdef{\_tocl:1}{#1#2#3{\_nofirst\_bigskip
167 \_bf\_llaptoclink{#1}{#2}\_hfill \_pgn{#3}\_tocpar\_medskip}

```

The <something> will be print as <something>.

doc.opm

```

173 \_let\lt=<
174 \_catcode`\<=13
175
176 \_def<#1>{\_langl\hbox{\it#1\}/}\rangle$}
177 \_everyintt{\_catcode`\<=13 }

```

If this macro is loaded by `\load` then we need to initialize catcodes using the `\_afteroad` macro.

doc.opm

```
184 \_def\_afterload{\_catcode\<=13 \_catcode\`=13 }
```

Main documentation points and hyperlinks to/from it. Main documentation point: `\`\foo``. User-level documentation point: `\^`\foo`, first occurrence only. The next occurrences are only links to the main documentation point. Link to user-level documentation point: `\~`\foo`. If user-level documentation point follows the main documentation point then use `\_forwardlink\`\foo``.

doc.opm

```
196 \_verbchar`
197
198 \_def\`#1{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
199   \_ifcsname cs:\_tmp\_endcsname\_else \_dest[cs:\_tmp]\_fi
200   \_sxdef{cs:\_tmp}{}%
201   \_hbox{\_ifcsname cs:\_tmp\_endcsname
202     \_link[cs:\_tmp]{\Red}{\_tt\_csstring\\\tmp}\_else
203     {\_tt\Red\_csstring\\\tmp}\_fi}%
204 }
205 \_def\_forwardlink\`#1{{\_slet{cs:\_csstring#1}{relax}\`#1}}
206
207 \_def\^`#1{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
208   \_hbox{\_ifcsname cs:\_tmp\_endcsname \_else \_dest[cs:\_tmp]\_sxdef{cs:\_tmp}{}\_fi
209     \_link[cs:\_tmp]{\Blue}{\_tt\_string#1}}%
210   \_futurelet\_next\_cslinkA
211 }
212 \_def\_cslinkA{\_ifx\_next\`\_ea\_ignoreit \_else \_ea\_ea\_ea\`\_ea\_string\_fi}
213
214 \_def\~`#1{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
215   \_hbox{\_link[cs:\_tmp]{\Blue}{\_tt\_string#1}}%
216   \_futurelet\_next\_cslinkA
217 }
```

## Index

`\_aboveliskip` 126  
`\_abovetitle` 121, 124  
`\activequotes` 186  
`\_addcitelist` 150  
`\_addcolor` 110  
`\_additcorr` 102  
`\address` 25, 175  
`\_addtabitemx` 145  
`\addto` 28, 38, 54, 103  
`\_addtomodlist` 76  
`\adef` 17, 28, 38  
`\adots` 85  
`\advancepageno` 103, 105  
`\afterfi` 28, 41  
`\_afteritcorr` 102  
`\_afterload` 52  
`\_allocator` 39  
`\allowbreak` 56  
`\altquotes` 185–186  
`\_asciisortingtrue` 170  
`\_athe` 128  
`\_authorname` 153  
`\b` 57  
`\_backgroundbox` 104  
`\backgroundpic` 138  
`\bbchar` 80, 95  
`\begblock` 14, 27, 128  
`\begitems` 13–14, 27, 48, 127  
`\begmulti` 19, 27, 48, 147  
`\_begoutput` 103–104, 118  
`\begtt` 16–18, 27, 47, 104, 129, 131  
`\_begtti` 129  
`\_belowliskip` 126  
`\_belowtitle` 121, 124  
`\bf` 8–9, 64–65, 80, 95  
`\bgroup` 37  
`\bi` 8–9, 64–65, 80, 95  
`\bib` 20–21, 27, 151  
`\_bibA` 151  
`\_bibB` 151  
`\_bibgl` 151  
`\bibmark` 149, 151, 153  
`\_bibnn` 150  
`\bibnum` 115, 149  
`\biboptions` 48, 158  
`\_bibp` 149  
`\bibpart` 21, 48, 149  
`\_bibskip` 152  
`\bibtexhook` 48, 152  
`\_bibwarning` 153, 156  
`\big` 84  
`\Big` 84  
`\bigbreak` 56  
`\bigg` 84  
`\Bigg` 84  
`\biggl` 84  
`\Biggl` 84  
`\biggm` 84  
`\Biggm` 84  
`\biggr` 84  
`\Biggr` 84  
`\bigl` 84  
`\Bigl` 84  
`\bigm` 84  
`\Bigm` 84  
`\bigr` 84  
`\Bigr` 84  
`\bigskip` 56  
`\Black` 108  
`\Blue` 21, 108  
`\bmod` 87  
`\boldify` 67, 102  
`\boldmath` 9, 79, 81, 90–91, 101  
`\_boldunimath` 91  
`\bordermatrix` 88  
`\_bordermatrixwithdelims` 88  
`\boxlines` 175  
`\bp` 28, 53  
`\_bp` 53  
`\_bprinta` 153, 156  
`\_bprintb` 153, 156  
`\_bprintc` 153, 156  
`\_bprintv` 153, 156  
`\bracedparam` 52  
`\break` 56  
`\Brown` 108  
`\bslash` 38  
`\buildrel` 87  
`\bye` 38, 59  
`\_byehook` 38  
`\c` 57  
`\cal` 80, 95  
`\caption` 10–12, 27, 125  
`\_captionsep` 125  
`\cases` 88  
`\catalogexclude` 78  
`\catalogmathsample` 78  
`\catalogonly` 78  
`\catalogsample` 78  
`\catcode` 53  
`\cdots` 85  
`\centerline` 56  
`\chap` 10, 12, 17–18, 26, 53, 121, 123  
`\_chapfont` 67, 121  
`\_chapx` 122  
`\_checkexists` 34  
`\chyph` 24, 187  
`\_circle` 138–139  
`\circleparams` 50  
`\cite` 12, 20–21, 27, 149, 152  
`\_citeA` 149  
`\_citeborder` 12, 116  
`\_citeI` 150  
`\clipincircle` 23, 141  
`\clipinoval` 23, 141  
`\_clipinpath` 141  
`\clqq` 187  
`\cmykcolordef` 110  
`\_cmyktorgb` 109  
`\cnvinfo` 51  
`\_cod` 28, 33–34, 54  
`\code` 16–17, 27, 47, 128  
`\_codedecl` 28, 33–35  
`\colnum` 145  
`\_colorcrop` 109  
`\colordef` 21–22, 28, 107–109, 111  
`\_colordefFin` 109  
`\_colorstackpop` 109  
`\_colorstackpush` 109  
`\_colorstackset` 109  
`\colsep` 48  
`\commentchars` 18, 130–132  
`\_commoncolordef` 110  
`\_completepage` 103–104  
`\_compoundchars` 168  
`\_compoundcharscs` 168  
`\_compoundcharsen` 168  
`\cong` 87  
`\_corrmsize` 81, 92  
`\cramped` 90  
`\crl` 15, 143, 146  
`\crli` 15, 142, 144, 146  
`\crl1` 15, 146  
`\crl1i` 15, 142, 146  
`\crlp` 15, 142, 146  
`\crqq` 187  
`\cs` 28, 38  
`\CS` 180  
`\cskip` 10, 125  
`\cslang` 24, 181  
`\csplain` 180  
`\csquotes` 24, 185  
`\_ctablelist` 51  
`\_currfamily` 74  
`\_currpage` 113, 117, 187  
`\currstyle` 90  
`\_currV` 70, 75  
`\currvar` 8–9, 63–65, 67–68, 76  
`\Cyan` 21, 108  
`\d` 57  
`\_dbib` 151  
`\_ddlinedata` 144

<code>\ddots</code> 85	<code>\_ensureblack</code> 104	<code>\fontfam</code> 5, 7, 9, 27, 29, 60,
<code>\_decdigits</code> 53	<code>\eoldef</code> 28, 52, 129	64–65, 67–68, 73–74, 77–79
<code>\decr</code> 28, 38	<code>\eqalign</code> 49, 88	<code>\_fontfeatures</code> 70, 78
<code>\_defaultfontfeatures</code> 78	<code>\eqalignno</code> 10, 88	<code>\fontlet</code> 28, 61–65
<code>\defaultitem</code> 14, 48, 127	<code>\eqbox</code> 28, 142, 147	<code>\_fontloaded</code> 62
<code>\delang</code> 24, 181	<code>\eqboxsize</code> 143, 147	<code>\_fontnamegen</code> 69–70, 74
<code>\dequotes</code> 24, 185	<code>\eqlines</code> 49, 88	<code>\_fontselector</code> 62
<code>\dest</code> 13, 115	<code>\eqmark</code> 10, 12, 27, 88, 126	<code>\footins</code> 103–105, 174
<code>\_destactive</code> 115	<code>\eqspace</code> 49, 88	<code>\footline</code> 6, 50, 103–104
<code>\_destboxes</code> 179	<code>\eqstyle</code> 49, 88	<code>\footlinedist</code> 6, 50
<code>\_destheight</code> 115	<code>\everycapitonf</code> 48	<code>\footnote</code> 7, 103, 105
<code>\displaylines</code> 88	<code>\everycapitont</code> 48	<code>\_footnoterule</code> 103–104
<code>\do</code> 41–42	<code>\everyii</code> 49, 171	<code>\footstrut</code> 105
<code>\_do</code> 42	<code>\everyintt</code> 17, 47	<code>\foreach</code> 28, 41
<code>\dobystyle</code> 90	<code>\everyitem</code> 48	<code>\_foreach</code> 42
<code>\_doc</code> 28, 33–34, 54	<code>\everylist</code> 14, 48	<code>\foreachdef</code> 28, 43
<code>\_docompound</code> 169	<code>\everymnote</code> 49	<code>\_forlevel</code> 42
<code>\doloadmath</code> 90–91	<code>\everytable</code> 49, 142	<code>\_formatcmyk</code> 108–109
<code>\_doresizefont</code> 62, 74	<code>\everytocline</code> 48, 117	<code>\_formatgrey</code> 108
<code>\_doresizetfmfont</code> 62	<code>\everytt</code> 16–18, 47, 129	<code>\_formatrgb</code> 108–109
<code>\_doresizeunifont</code> 62, 74, 78	<code>\_ewref</code> 113	<code>\fornum</code> 28, 42
<code>\_doshadow</code> 140	<code>\expr</code> 28, 53	<code>\_fornumB</code> 42
<code>\_dosorting</code> 170	<code>\_expr</code> 53	<code>\fornumstep</code> 42
<code>\dospecials</code> 55	<code>\_famalias</code> 73, 77	<code>\fR</code> 15, 146
<code>\dosupereject</code> 56, 103	<code>\_famdecl</code> 66, 69–71, 74	<code>\frak</code> 80, 95
<code>\doteq</code> 87	<code>\_famdepend</code> 76	<code>\frame</code> 15, 23, 147
<code>\dotfill</code> 58	<code>\_famfrom</code> 73, 78	<code>\frqq</code> 187
<code>\dots</code> 57	<code>\_faminfo</code> 73, 77–78	<code>\frquotes</code> 185
<code>\_douseK</code> 109	<code>\_famtext</code> 73, 77	<code>\fS</code> 15, 144, 146
<code>\_doverbininput</code> 130	<code>\famvardef</code> 64–67, 69, 71,	<code>\_fsetV</code> 70, 75
<code>\_dowhichtfm</code> 63	75–76	<code>\_fullrectangle</code> 128
<code>\downbracefill</code> 58	<code>\_famvardefA</code> 76	<code>\_fvars</code> 70, 75
<code>\draft</code> 7, 106	<code>\fC</code> 15, 146	<code>\fX</code> 15, 146
<code>\_dsp</code> 132	<code>\fcolor</code> 139	<code>\_getforstack</code> 42
<code>\ea</code> 34	<code>\_ffonum</code> 75	<code>\_gfnotenum</code> 173
<code>\_ea</code> 34	<code>\filbreak</code> 56	<code>\goodbreak</code> 56
<code>\ecite</code> 20, 149	<code>\_fillstroke</code> 108, 139	<code>\gpageno</code> 103–104, 115
<code>\_editorname</code> 153	<code>\_firstnoindent</code> 10, 122, 124	<code>\_greekdef</code> 93
<code>\egroup</code> 37	<code>\fixmnotes</code> 7, 174	<code>\Green</code> 108
<code>\ehyph</code> 24, 187	<code>\fL</code> 15, 146	<code>\Grey</code> 108
<code>\eject</code> 56	<code>\flqq</code> 187	<code>\headline</code> 6, 50, 103–104
<code>\em</code> 8, 102	<code>\fmtname</code> 30	<code>\headlinedist</code> 6, 50, 104
<code>\empty</code> 37	<code>\_fnfborder</code> 13, 116	<code>\_hexprint</code> 120
<code>\endblock</code> 14, 27, 128	<code>\fnote</code> 7, 17, 27, 103, 174	<code>\hglsue</code> 56
<code>\_endcode</code> 28, 33–35	<code>\fnotelinks</code> 13, 173	<code>\hhkern</code> 49
<code>\endgraf</code> 55	<code>\fnotemark</code> 7, 174	<code>\hicolor</code> 135
<code>\endinsert</code> 11, 105	<code>\fnotenum</code> 173	<code>\hicolors</code> 48
<code>\enditems</code> 13, 27, 48, 127	<code>\fnotenumchapters</code> 7, 122,	<code>\_hicomments</code> 132
<code>\endline</code> 55	173	<code>\hidewidth</code> 57
<code>\endmulti</code> 19, 27, 48, 147	<code>\fnotenumglobal</code> 7, 173	<code>\hisyntax</code> 18, 129, 133, 135
<code>\_endnamespace</code> 28, 33, 35	<code>\fnotenumpages</code> 7, 173, 179	<code>\hphantom</code> 86
<code>\_endoutput</code> 103	<code>\_fnotestack</code> 109	<code>\hrulefill</code> 58
<code>\_endslides</code> 177	<code>\fnotetext</code> 7, 174	<code>\hyperlinks</code> 12–13, 20, 27,
<code>\endtt</code> 16–18, 27, 47, 129, 131	<code>\_fnset</code> 128, 174	115–116, 122
<code>\enlang</code> 24, 181	<code>\_fntborder</code> 13, 116	<code>\ialign</code> 57
<code>\enquotes</code> 24, 185	<code>\folio</code> 26, 105	<code>\_ifAleB</code> 169
<code>\enskip</code> 56	<code>\fontdef</code> 28, 60–62, 64–66, 76	<code>\_ifexistfam</code> 44, 69
<code>\enspace</code> 56		<code>\_iflocalcolor</code> 108



<code>\_ifmathloading</code> 91	<code>\langlist</code> 24, 184	<code>\_math</code> 86
<code>\_ifmathsb</code> 82	<code>\_langw</code> 24, 184	<code>\mathbox</code> 10, 90, 92
<code>\_ifpgfnote</code> 173	<code>\lastpage</code> 26, 187	<code>\mathfaminfo</code> 74
<code>\_ignoredchars</code> 168	<code>\LaTeX</code> 180	<code>\mathhexbox</code> 57
<code>\ignoreit</code> 28, 38	<code>\layernum</code> 177–178	<code>\_mathloadingfalse</code> 90–91
<code>\ignorept</code> 28, 53	<code>\layers</code> 178–179	<code>\_mathloadingtrue</code> 91
<code>\ignoresecond</code> 28, 38	<code>\_layertext</code> 178–179	<code>\mathpalette</code> 86
<code>\ignoreslash</code> 180	<code>\lcolor</code> 139	<code>\mathsboff</code> 33, 82
<code>\ii</code> 18–19, 26, 172	<code>\ldots</code> 85	<code>\mathsbon</code> 33, 82, 131
<code>\iid</code> 18–19, 172	<code>\leavevmode</code> 57	<code>\mathstrut</code> 86
<code>\iindent</code> 14, 47	<code>\leftarrowfill</code> 58	<code>\mathstyles</code> 28, 89–90
<code>\iindex</code> 172	<code>\leftline</code> 56	<code>\matrix</code> 87
<code>\iis</code> 19, 172	<code>\letfont</code> 187	<code>\maxlines</code> 193–194
<code>\iitype</code> 19, 172	<code>\letter</code> 25, 27, 175	<code>\_maybetod</code> 163
<code>\_iitypesaved</code> 172	<code>\_lfnote</code> 173	<code>\medbreak</code> 56
<code>\ilevel</code> 14, 48	<code>\LightGrey</code> 108	<code>\medskip</code> 56
<code>\ilink</code> 13, 115	<code>\line</code> 56	<code>\_mergesort</code> 170
<code>\_inchap</code> 123	<code>\link</code> 115	<code>\_mfontfeatures</code> 92
<code>\incircle</code> 23, 50, 139	<code>\_linkactive</code> 115–116	<code>\_mfontsrule</code> 101–102
<code>\incr</code> 28, 38	<code>\_linkdimens</code> 115	<code>\midinsert</code> 11, 105
<code>\ingnslash</code> 180	<code>\lipsum</code> 26, 187	<code>\mit</code> 80
<code>\_initfontfamily</code> 71, 74	<code>\_lipsumload</code> 187	<code>\mnote</code> 7, 27, 49, 174
<code>\initunifonts</code> 60, 62, 74	<code>\lipsumtext</code> 187	<code>\_mnoteA</code> 175
<code>\_inkdefs</code> 136	<code>\_listfamnames</code> 77	<code>\_mnoteD</code> 174
<code>\inkinspic</code> 22, 136	<code>\_listskipA</code> 126	<code>\mnoteindent</code> 49
<code>\_inmath</code> 95	<code>\_listskipamount</code> 48, 127	<code>\_mnotesfixed</code> 174
<code>\inoval</code> 23, 50, 139	<code>\_listskipB</code> 126	<code>\mnotesize</code> 7, 49
<code>\_inputref</code> 112	<code>\llap</code> 56	<code>\mnoteskip</code> 175
<code>\_insec</code> 123	<code>\_llaptoclink</code> 117	<code>\moddef</code> 64–65, 67, 69–70, 74, 76
<code>\_insecc</code> 123	<code>\lmfil</code> 49	<code>\_modlist</code> 76
<code>\_insertmark</code> 124	<code>\load</code> 25, 27, 34, 52, 193, 196	<code>\morecolors</code> 21, 111
<code>\insertoutline</code> 13, 119	<code>\loadboldmath</code> 90–92	<code>\mspan</code> 15, 146
<code>\_insertshadowresources</code> 140	<code>\loadmath</code> 9, 65, 90–91, 93	<code>\_mtext</code> 125, 184
<code>\inspic</code> 22, 27, 47, 136	<code>\_loadmathfamily</code> 81	<code>\_Mtext</code> 162, 165
<code>\_inspicA</code> 136	<code>\_loadpattr</code> 182	<code>\multispan</code> 15, 57
<code>\_inspicB</code> 136	<code>\_loadumathfamily</code> 92	<code>\_mv</code> 138
<code>\_interliskip</code> 126	<code>\localcolor</code> 108	<code>\_namespace</code> 27–28, 33–35
<code>\_isAleB</code> 169	<code>\loggingall</code> 38	<code>\narrower</code> 57
<code>\isdefined</code> 28, 43	<code>\loop</code> 28, 41	<code>\_narrowlastlinecentered</code> 126
<code>\isempty</code> 28, 43	<code>\_loop</code> 41	<code>\nbb</code> 38
<code>\isequal</code> 28, 43	<code>\lorem</code> 26, 187	<code>\nbpar</code> 122, 124
<code>\isfile</code> 28, 43–44	<code>\_lrmnote</code> 175	<code>\_negationof</code> 98
<code>\isfont</code> 28, 43–44	<code>\LuaTeX</code> 180	<code>\negthinspace</code> 56
<code>\isinlist</code> 28, 43–44	<code>\lwidth</code> 139	<code>\newattribute</code> 40
<code>\ismacro</code> 28, 43	<code>\Magenta</code> 108	<code>\newbox</code> 40
<code>\isnextchar</code> 28, 44	<code>\magnification</code> 59	<code>\newcatcodetable</code> 40
<code>\istoksempy</code> 28, 43	<code>\magscale</code> 6, 27, 107	<code>\newcount</code> 28, 40
<code>\it</code> 8, 64–65, 80, 95	<code>\magstep</code> 55	<code>\newcurrfontsize</code> 63, 102
<code>\item</code> 57	<code>\magstephalf</code> 55	<code>\newdimen</code> 28, 40
<code>\itemitem</code> 57	<code>\mainbaselineskip</code> 8, 101	<code>\newfam</code> 40
<code>\itemnum</code> 127	<code>\mainfosize</code> 8, 101	<code>\_newfontloaded</code> 62
<code>\jointrel</code> 85	<code>\_makefootline</code> 104	<code>\newif</code> 28, 33, 41
<code>\_keepmeaning</code> 62	<code>\_makeheadline</code> 103–104	<code>\_newifi</code> 28, 33, 41
<code>\kv</code> 28, 54–55	<code>\makeindex</code> 18–19, 24, 26, 167, 171	<code>\_newiiletter</code> 171
<code>\_kvscan</code> 55	<code>\maketoc</code> 18, 26, 118, 122	<code>\newinsert</code> 40
<code>\_kvunknown</code> 55	<code>\margins</code> 5–6, 27, 29, 103–104, 106	<code>\newmuskip</code> 40
<code>\label</code> 12, 26, 114, 122, 179		

<code>\newread</code> 40	<code>\opwarning</code> 28, 38	<code>\_printinverbatim</code> 128–129
<code>\newskip</code> 40	<code>\_othe</code> 122	<code>\_printitem</code> 127
<code>\newtoks</code> 40	<code>\outlines</code> 13, 27, 118	<code>\_printlabel</code> 114
<code>\newwrite</code> 40	<code>\_outlinesA</code> 118	<code>\_printlayers</code> 178
<code>\nextpages</code> 50	<code>\_outlinesB</code> 118	<code>\_printnumberedpar</code> 126
<code>\nl</code> 10, 124	<code>\_oval</code> 138–139	<code>\_printrefnum</code> 122–124
<code>\nobibwarning</code> 152, 156	<code>\ovalparams</code> 23, 50	<code>\_printsavedcites</code> 150
<code>\_nobibwarnlist</code> 156	<code>\overbrace</code> 86	<code>\_printsec</code> 10, 121, 177
<code>\nobreak</code> 56	<code>\overlapmargin</code> 139	<code>\_printsecc</code> 10, 121
<code>\nocite</code> 20, 149, 152	<code>\overleftarrow</code> 86	<code>\_printtit</code> 121
<code>\_nofirst</code> 117	<code>\overrightarrow</code> 86	<code>\_printverb</code> 129–130, 132
<code>\nointerlineskip</code> 56	<code>\_pagecontents</code> 103–104	<code>\_printverblin</code> 129–130
<code>\noloadmath</code> 9, 65, 91	<code>\_pagedest</code> 103–104, 179	<code>\_printverblinenum</code> 130
<code>\nonfrenchspacing</code> 46, 182	<code>\pageinsert</code> 105	<code>\private</code> 28, 32, 34
<code>\nonum</code> 10, 122–123	<code>\pageno</code> 26, 103, 105	<code>\pshow</code> 177
<code>\nonumcitations</code> 20, 27, 149	<code>\_paramtabdeclarep</code> 145	<code>\ptmunit</code> 81
<code>\nopagenumbers</code> 6, 105	<code>\pcent</code> 38	<code>\ptunit</code> 8, 81
<code>\normalbottom</code> 105	<code>\_pdfborder</code> 116	<code>\public</code> 28, 32, 34
<code>\normalcatcodes</code> 52	<code>\pdfrotate</code> 22, 137	<code>\_putforstack</code> 42
<code>\normalmath</code> 9, 79, 81, 90–91, 101	<code>\pdfscale</code> 22, 137	<code>\putpic</code> 24, 138
<code>\_normalunimath</code> 91–92	<code>\pdfunidef</code> 118, 120, 186	<code>\puttext</code> 24, 138
<code>\_nospaceafter</code> 52	<code>\_pdfunidefB</code> 120	<code>\_putttpenalty</code> 130
<code>\nospec</code> 24, 138	<code>\pg</code> 177	<code>\_qqA</code> 186
<code>\not</code> 89, 98	<code>\pgbackground</code> 7, 50, 103–104	<code>\_qqB</code> 186
<code>\notin</code> 87	<code>\_pgborder</code> 12–13, 116	<code>\qqquad</code> 56
<code>\notoc</code> 10, 123	<code>\pgbottomskip</code> 50, 103, 105	<code>\quad</code> 56
<code>\novspaces</code> 14, 127	<code>\_pgn</code> 117	<code>\quoteschars</code> 185
<code>\_nsprivate</code> 33, 35	<code>\_pgprint</code> 172	<code>\raggedbottom</code> 105
<code>\_nspublic</code> 33, 35	<code>\pgref</code> 12, 27, 114, 179	<code>\raggedright</code> 57
<code>\null</code> 37	<code>\phantom</code> 86	<code>\ratio</code> 23, 139
<code>\numberedpar</code> 11, 126	<code>\picdir</code> 22, 47	<code>\rcite</code> 20, 27, 149
<code>\obeylines</code> 55	<code>\picheight</code> 22, 47	<code>\readkv</code> 28, 54–55
<code>\obeyspaces</code> 55	<code>\_picparams</code> 136	<code>\_readverb</code> 129
<code>\offinterlineskip</code> 56	<code>\picw</code> 22, 47	<code>\Red</code> 21, 108
<code>\oldaccents</code> 29, 58, 152	<code>\picwidth</code> 22, 47	<code>\ref</code> 12, 27, 114, 179
<code>\onlycmyk</code> 21, 108–109	<code>\plaintexcatcodes</code> 51	<code>\_refborder</code> 12, 116
<code>\_onlyif</code> 70, 75	<code>\pllang</code> 24, 181	<code>\_refdecl</code> 113
<code>\onlyrgb</code> 21, 108–109	<code>\pmatrix</code> 87	<code>\regmacro</code> 13, 18, 103–104, 118, 121
<code>\ooalign</code> 57	<code>\pmod</code> 87	<code>\_regmark</code> 104, 118
<code>\_openfnotestack</code> 109	<code>\_preparesorting</code> 169	<code>\_regoptsizes</code> 61, 69, 71, 75
<code>\_openfnotestackA</code> 109	<code>\_prepareverbdata</code> 129–130, 135	<code>\_regoul</code> 118, 128
<code>\openref</code> 103, 113	<code>\_prepcommalist</code> 75	<code>\_regquotes</code> 186
<code>\_openrefA</code> 113	<code>\_prepinverb</code> 121	<code>\_regtfm</code> 61, 63
<code>\openup</code> 88	<code>\_preplang</code> 182	<code>\_regtoc</code> 118
<code>\_opfootnote</code> 105, 174	<code>\_prepoffsets</code> 103	<code>\_reloading</code> 63
<code>\opinup</code> 28, 51	<code>\prime</code> 84	<code>\_remifirstunderscore</code> 76
<code>\OPmac</code> 180	<code>\_printbib</code> 152–153	<code>\removelastskip</code> 56
<code>\opt</code> 52, 54	<code>\_printcaptionf</code> 125	<code>\_removeoutbraces</code> 120
<code>\optdef</code> 28, 52, 54	<code>\_printcaptiont</code> 125	<code>\_removeoutmath</code> 120
<code>\OpTeX</code> 180	<code>\_printchap</code> 10, 121	<code>\removespaces</code> 53
<code>\optexcatcodes</code> 51	<code>\_printcomments</code> 132	<code>\repeat</code> 28, 41
<code>\_optexoutput</code> 103–104	<code>\printdoc</code> 193, 195	<code>\_repeat</code> 41
<code>\optexversion</code> 30	<code>\printdoctail</code> 193, 195	<code>\replfromto</code> 134
<code>\_optfontalias</code> 72, 75	<code>\_printfnotemark</code> 173	<code>\replstring</code> 28, 53–54, 110, 120, 143
<code>\_optname</code> 71, 75	<code>\_printii</code> 171	<code>\replthis</code> 134
<code>\_optnameA</code> 75	<code>\_printiipages</code> 171	<code>\report</code> 25, 27, 175
<code>\_optsize</code> 61	<code>\_printindexitem</code> 171	

<code>\_resetfam</code> 76	<code>\_setlistskip</code> 126	<code>\_stripzeros</code> 110
<code>\resetmod</code> 65, 69–71	<code>\_setmainvalues</code> 100–101	<code>\strutbox</code> 56, 101
<code>\_resetnamespace</code> 33, 35	<code>\_setmainvaluesL</code> 101, 103	<code>\style</code> 13, 127
<code>\_resetnonumnotoc</code> 123	<code>\_setmathdimens</code> 81, 91	<code>\stylenum</code> 90
<code>\_resizefont</code> 62–63	<code>\_setmathfamily</code> 81	<code>\subject</code> 25, 175
<code>\resizethefont</code> 60, 62–63	<code>\_setmathfonts</code> 101–102	<code>\subtit</code> 177
<code>\restorectable</code> 28, 51	<code>\_setmathsizes</code> 79, 81, 92	<code>\supereject</code> 56
<code>\_restoremathsb</code> 131	<code>\_setnabla</code> 94	<code>\sxdef</code> 28, 38
<code>\_reversetfm</code> 63	<code>\_setnewmeaning</code> 76	<code>\_tabdata</code> 144
<code>\_rfontskipat</code> 62–63	<code>\_setprimarysorting</code> 168–169	<code>\_tabdeclarec</code> 16, 145
<code>\rgbcolordef</code> 22, 108–110	<code>\setrgbcolor</code> 107–108	<code>\_tabdeclarel</code> 145
<code>\_rgbtocmyk</code> 109	<code>\_setsecondarysorting</code> 169	<code>\_tabdeclarer</code> 145
<code>\rightarrowfill</code> 58	<code>\settabs</code> 29	<code>\tabiteml</code> 15, 49, 142
<code>\rightleftharpoons</code> 87	<code>\_settinglayer</code> 179	<code>\tabitemr</code> 15, 49, 142
<code>\rightline</code> 56	<code>\_setunimathdimens</code> 91	<code>\table</code> 14–15, 27, 49, 142–143
<code>\rlap</code> 56	<code>\_setverb</code> 128–129	<code>\_tableA</code> 143
<code>\rm</code> 8, 64–65, 80, 95	<code>\setwordspace</code> 65, 68, 78	<code>\_tableB</code> 143
<code>\_rmfixed</code> 103	<code>\setwsp</code> 78	<code>\_tablebox</code> 143
<code>\rotbox</code> 23, 137	<code>\_setxhsize</code> 104	<code>\_tableC</code> 144
<code>\rulewidth</code> 16, 147	<code>\shadow</code> 139	<code>\_tablew</code> 143–144
<code>\_runboldmath</code> 102	<code>\shadowb</code> 140	<code>\_tableW</code> 143
<code>\_savedcites</code> 149–150	<code>\shadowlevels</code> 140	<code>\tablinespace</code> 49, 143, 146
<code>\_savedttchar</code> 129	<code>\shadowmoveto</code> 140	<code>\tabskipl</code> 49, 142
<code>\_savedttcharc</code> 129	<code>\shordcitations</code> 150	<code>\_tabskipmid</code> 144
<code>\_savemathsb</code> 131	<code>\shortcitations</code> 20, 27, 150	<code>\tabskipr</code> 49, 142
<code>\sb</code> 84	<code>\showcolor</code> 111	<code>\tabspaces</code> 48
<code>\_scalebig</code> 84	<code>\showlabels</code> 12, 114	<code>\tabstrut</code> 49, 143
<code>\scalemain</code> 9, 101	<code>\shyph</code> 24, 187	<code>\tenbf</code> 59
<code>\scantabdata</code> 144, 146	<code>\_sizemscript</code> 81, 100	<code>\tenbi</code> 59
<code>\scantoel</code> 52, 117, 121, 123	<code>\_sizemsscript</code> 81, 100	<code>\tenit</code> 59
<code>\scantwodimens</code> 138	<code>\_sizemtext</code> 81, 100	<code>\tenrm</code> 59
<code>\script</code> 80, 95	<code>\_sizespec</code> 61, 63	<code>\tentt</code> 59
<code>\_scriptmff</code> 90, 92	<code>\skew</code> 86	<code>\_testAleB</code> 169
<code>\sdef</code> 6, 14, 24, 28, 38	<code>\skiptoel</code> 52	<code>\_testcommentchars</code> 130, 132
<code>\_sdestbox</code> 179	<code>\sklang</code> 24, 181	<code>\TeX</code> 180
<code>\sec</code> 10, 12, 17–18, 26, 53, 121, 123	<code>\slantcorr</code> 180	<code>\textindent</code> 57
<code>\secc</code> 10, 12, 18, 26, 53, 121, 123	<code>\slash</code> 56	<code>\_textmff</code> 90, 92
<code>\_seccfont</code> 67, 121	<code>\slet</code> 28, 38	<code>\_thecapnum</code> 125
<code>\_seccx</code> 122	<code>\_slidelayer</code> 178	<code>\_thecaptitle</code> 125
<code>\_secfont</code> 67, 121	<code>\_slidelinks</code> 179	<code>\_thechapnum</code> 122
<code>\secl</code> 125	<code>\_slideopen</code> 179	<code>\_thednum</code> 122
<code>\_seclp</code> 125	<code>\_slidepage</code> 178	<code>\_thefnum</code> 122
<code>\_sectionlevel</code> 122	<code>\_slidepageB</code> 178	<code>\thefontscale</code> 9, 27, 102
<code>\_secx</code> 122	<code>\slides</code> 25, 27, 138, 151, 176	<code>\thefontsize</code> 9, 27, 102
<code>\_setbaselineskip</code> 101	<code>\_slideshowook</code> 179	<code>\_theoutline</code> 124
<code>\setcmykcolor</code> 21, 107–108	<code>\slideshow</code> 178–179	<code>\_theseccnum</code> 122
<code>\_setcolor</code> 109	<code>\smallbreak</code> 56	<code>\_theseccnum</code> 122
<code>\setctable</code> 28, 51	<code>\smallskip</code> 56	<code>\_thetnum</code> 122
<code>\setff</code> 64–65, 67–68, 70, 78	<code>\smash</code> 86	<code>\thinspace</code> 56
<code>\_setflcolor</code> 139	<code>\sortcitations</code> 20, 27, 150	<code>\thisoutline</code> 13, 124
<code>\setfontcolor</code> 65, 68, 70, 78	<code>\_sortingdata</code> 167	<code>\thistable</code> 49, 142
<code>\setfontsize</code> 9, 60–61, 64–66, 68, 100	<code>\_sortingdatacs</code> 168	<code>\tit</code> 10, 26, 48, 121
<code>\setgreycolor</code> 107–108	<code>\sp</code> 84	<code>\_titfont</code> 67, 121
<code>\setletterspace</code> 65, 68, 70, 78	<code>\space</code> 37	<code>\_titskip</code> 48
	<code>\_sscriptmff</code> 92	<code>\_tmpcatcodes</code> 51
	<code>\_startitem</code> 127	<code>\_tmptoks</code> 53
	<code>\_startverb</code> 129–130	<code>\_tocborder</code> 12, 116
		<code>\_tocdotfill</code> 117

<code>\_tocline</code> 116–117	<code>\underbrace</code> 86	<code>\_viscanparameter</code> 130
<code>\_toclist</code> 116, 118	<code>\_unimathboldfont</code> 91	<code>\visiblesp</code> 132
<code>\_tocpar</code> 117	<code>\_unimathfont</code> 91	<code>\vphantom</code> 86
<code>\tocrefnum</code> 115, 117	<code>\_unresolvedrefs</code> 114	<code>\vspan</code> 16, 146
<code>\today</code> 185	<code>\_unsskip</code> 145	<code>\vvkern</code> 49
<code>\topglue</code> 56	<code>\upbracefill</code> 58	<code>\_wbib</code> 151
<code>\topins</code> 103–105	<code>\url</code> 12–13, 116	<code>\_whatresize</code> 62
<code>\topinsert</code> 11, 103, 105	<code>\_urlactive</code> 115–116	<code>\White</code> 108
<code>\totalpages</code> 26, 187	<code>\_urlborder</code> 12, 116	<code>\_wichtfm</code> 63
<code>\tracingall</code> 38	<code>\_urlfont</code> 67, 116	<code>\_wipeepar</code> 124
<code>\transformbox</code> 23, 136–137	<code>\usebib</code> 20, 27, 150, 152, 186	<code>\wlabel</code> 12, 114, 179
<code>\trycs</code> 28, 38	<code>\_usedirectly</code> 57	<code>\wlog</code> 28, 37
<code>\_tryloadfamslocal</code> 77	<code>\useit</code> 28, 38	<code>\_wref</code> 112–113
<code>\tsize</code> 49, 142, 144	<code>\useK</code> 107, 109, 111	<code>\wterm</code> 28, 35
<code>\tskip</code> 15, 146	<code>\_uselang</code> 182	<code>\xargs</code> 28, 34
<code>\tt</code> 13, 64–65, 67, 80, 95	<code>\uselanguage</code> 24, 183	<code>\_Xbib</code> 149–151
<code>\_ttfont</code> 67, 128	<code>\useoptex</code> 26, 187	<code>\_Xcite</code> 150
<code>\ttindent</code> 16, 18, 47	<code>\useOpTeX</code> 26, 187	<code>\_Xeqlbox</code> 147
<code>\ttline</code> 16–17, 47	<code>\usesecond</code> 28, 38	<code>\XeTeX</code> 180
<code>\_ttpenalty</code> 128, 130	<code>\uslang</code> 187	<code>\_Xfnote</code> 174
<code>\ttraggedright</code> 57	<code>\uv</code> 187	<code>\_xhsize</code> 104
<code>\ttshift</code> 47	<code>\_vcomments</code> 132	<code>\_Xindex</code> 172
<code>\_ttskip</code> 128	<code>\vdots</code> 85	<code>\_Xlabel</code> 114
<code>\_ttunifont</code> 74	<code>\_verbatimcatcodes</code> 128	<code>\_Xmnote</code> 174
<code>\typoscale</code> 8–9, 27, 65, 101–102	<code>\verbchar</code> 16–17, 27, 47, 129	<code>\_Xpage</code> 112–113, 117, 174, 187
<code>\typosize</code> 8–9, 27, 65, 67, 100–103	<code>\verbinput</code> 17–18, 27, 47, 130–131	<code>\Xrefversion</code> 113
<code>\ulink</code> 12–13, 115–116	<code>\vfootnote</code> 105, 174	<code>\_xscan</code> 135
<code>\_umahrangegreek</code> 93	<code>\vglue</code> 56	<code>\_xscanR</code> 135
<code>\_umahrangeGREEK</code> 93	<code>\_vidolines</code> 130	<code>\_Xtoc</code> 53, 116, 120
<code>\_umathcharholes</code> 93	<code>\_vifile</code> 128	<code>\Yellow</code> 21, 108
<code>\_umathrange</code> 93–94	<code>\_viline</code> 128	<code>\_zerotabrule</code> 146
<code>\underbar</code> 56	<code>\_vinolines</code> 130	<code>\_zo</code> 40
	<code>\_viscanminus</code> 130	<code>\_zoskip</code> 40