

# OpTeX

## Format Based on Plain TeX

Version 1.20

*Petr Olšák, 2020, 2021, 2022, 2023, 2024, 2025, 2026*

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

OpTeX is LuaTeX format with Plain TeX macros and with extended powerful macros, especially those taken from OPmac<sup>1</sup>. Only LuaTeX engine is supported (more exactly: LuaHBTeX engine is used from 2025).

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, transparency . . . . .	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
1.7.8	OpTeX tricks . . . . .	27
1.8	Summary . . . . .	28
1.9	API for macro writers . . . . .	28
1.10	Compatibility with Plain TeX . . . . .	30
1.11	Related documents . . . . .	30
<b>2</b>	<b>Technical documentation</b>	<b>31</b>
2.1	The main initialization file . . . . .	31
2.2	Basic principles of OpTeX sources . . . . .	33
2.2.1	Concept of namespaces of control sequences . . . . .	33
2.2.2	Macro files syntax . . . . .	34
2.2.3	Name spaces for package writers . . . . .	34
2.2.4	Summary about rules for external macro files published for OpTeX . . . . .	35

2.2.5	The implementation of the namespaces and macros for macro-files . . . . .	35
2.3	pdfTeX initialization . . . . .	37
2.4	Basic macros . . . . .	39
2.5	Allocators for TeX registers . . . . .	41
2.6	If-macros, loops, is-macros, cases . . . . .	43
2.6.1	Classical \newif . . . . .	43
2.6.2	Loops . . . . .	44
2.6.3	Is-macros and selection of cases . . . . .	46
2.7	Setting parameters . . . . .	48
2.7.1	Primitive registers . . . . .	48
2.7.2	Plain TeX registers . . . . .	50
2.7.3	Different settings than in plain TeX . . . . .	50
2.7.4	OpTeX parameters . . . . .	51
2.8	More OpTeX macros . . . . .	55
2.9	Using key=value format in parameters . . . . .	60
2.10	Plain TeX macros . . . . .	62
2.11	Preloaded fonts for text mode . . . . .	66
2.12	Using \font primitive directly . . . . .	67
2.12.1	The \setfontsize macro . . . . .	68
2.12.2	The \font-like commands summary . . . . .	69
2.12.3	The \fontlet declarator . . . . .	69
2.12.4	Optical sizes . . . . .	69
2.12.5	Font rendering . . . . .	69
2.12.6	Implementation of resizing . . . . .	69
2.13	The Font Selection System . . . . .	71
2.13.1	Terminology . . . . .	71
2.13.2	Font families, selecting fonts . . . . .	72
2.13.3	Math Fonts . . . . .	73
2.13.4	Declaring font commands . . . . .	73
2.13.5	The \fontdef declarator in detail . . . . .	73
2.13.6	The \famvardef declarator . . . . .	73
2.13.7	The \tt variant selector . . . . .	74
2.13.8	Font commands defined by \def . . . . .	75
2.13.9	Modifying font features . . . . .	75
2.13.10	Special font modifiers . . . . .	75
2.13.11	How to create the font family file . . . . .	76
2.13.12	How to write the font family file with optical sizes . . . . .	79
2.13.13	How to register the font family in the Font Selection System . . . . .	80
2.13.14	Implementation of the Font Selection System . . . . .	81
2.14	Preloaded fonts for math mode . . . . .	88
2.15	Math macros . . . . .	91
2.16	Unicode-math fonts . . . . .	100
2.16.1	Unicode-math macros preloaded in the format . . . . .	101
2.16.2	Macros and codes set when \loadmath is processed firstly . . . . .	104
2.16.3	More Unicode-math examples . . . . .	112
2.16.4	Printing all Unicode math slots in used math font . . . . .	112
2.17	Scaling fonts in document (high-level macros) . . . . .	113
2.18	Output routine . . . . .	115
2.19	Margins . . . . .	119
2.20	Colors . . . . .	120
2.20.1	Basic concept . . . . .	120
2.20.2	Color mixing . . . . .	121

2.20.3	Implementation	121
2.21	The <code>.ref</code> file	125
2.22	References	127
2.23	Hyperlinks	129
2.24	Making table of contents	131
2.25	PDF outlines	133
2.25.1	Nesting PDF outlines	133
2.25.2	Strings in PDF outlines	134
2.26	Chapters, sections, subsections	135
2.27	Lists, items	141
2.28	Verbatim, listings	143
2.28.1	Inline and “display” verbatim	143
2.28.2	Listings with syntax highlighting	147
2.29	Graphics	150
2.30	The <code>\table</code> macro, tables and rules	156
2.30.1	The boundary declarator :	156
2.30.2	Usage of the <code>\tabskip</code> primitive	157
2.30.3	Tables to given width	157
2.30.4	<code>\eqbox</code> : boxes with equal width across the whole document	158
2.30.5	Implementation of the <code>\table</code> macro and friends	158
2.31	Balanced multi-columns	163
2.32	Citations, bibliography	165
2.32.1	Macros for citations and bibliography preloaded in the format	165
2.32.2	The <code>\usebib</code> command	169
2.32.3	Notes for bib-style writers	169
2.32.4	Direct reading of <code>.bib</code> files	171
2.32.5	The <code>usebib.opm</code> macro file loaded when <code>\usebib</code> is used	171
2.32.6	Usage of the <code>bib-iso690</code> style	177
2.32.7	Implementation of the <code>bib-iso690</code> style	184
2.33	Sorting and making Index	188
2.34	Footnotes and marginal notes	196
2.35	Styles	198
2.35.1	<code>\report</code> and <code>\letter</code> styles	198
2.35.2	<code>\slides</code> style for presentations	199
2.36	Logos	202
2.37	Multilingual support	203
2.37.1	Lowercase, uppercase codes	203
2.37.2	Multilingual phrases and quotation marks	204
2.37.3	Languages declaration	206
2.37.4	Data for various languages	209
2.38	Other macros	210
2.39	Lua code embedded to the format	213
2.39.1	General	214
2.39.2	Allocators	215
2.39.3	Callbacks	216
2.39.4	PDF object utilities	221
2.39.5	Management of PDF page resources	223
2.39.6	Handling of colors and transparency using attributes	223
2.40	Printing documentation	226
2.40.1	Implementation	227

# Chapter 1

## User documentation

### 1.1 Starting with OpTeX

OpTeX is compiled as a format for LuaHBTeX. Maybe there is a command `optex` in your T<sub>E</sub>X 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 TeX's macro `\footnote` can be used as usual. But a new macro `\fnote{⟨text⟩}` is defined. The footnote mark is added automatically and it is numbered on each chapter from one<sup>2</sup>. The `⟨text⟩` is scaled to 80 %. User can redefine footnote mark or scaling, as shown in the 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 TeX run because the relevant information is stored in an external file and read from it again. If you need to place the notes only to the fixed margin write `\fixmnotes\right` or `\fixmnotes\left`.

The `⟨text⟩` is formatted as a little paragraph with the maximal width `\mnotesize` ragged left on the left margins or ragged right on the right margins. The first line of this little paragraph has its vertical position given by the position of `\mnote` in the text. The exceptions are possible by 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.

<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`



If you write `\fontfam[?]` then all font families registered in OpTeX are listed on the terminal and in the log file. If you write `\fontfam[catalog]` then a catalog of all fonts registered in OpTeX and available in your T<sub>E</sub>X system is printed. See also [this 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 T<sub>E</sub>X: `{\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.

### 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-rn3, ABC-X3
<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. You can declare `\swapeqno` if you want to put equation marks to the opposite side.

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]` (prints the number of the referenced object) or `\pgref[label]` (prints the page number). 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]`.

The text printed by `\ref` or `\pgref` can be given explicitly by `\ref[label]{text}` or `\pgref[label]{text}`. If the *text* includes the @ character, it is replaced by implicitly printed text. Example: `see \ref[lab]{section~@}` prints the same as `see section~\ref[lab]`, but first case creates larger active area for mouse clicking, when `\hyperlinks` are declared.

If there are forward referenced objects then users have to run T<sub>E</sub>X 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 `\fnotelinks`  $\langle color-fnt \rangle$   $\langle color-fnf \rangle$  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), ...

```

---

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

```

\style A % items of type A), B), C), ...
\style x % small rectangle
\style X % big rectangle
\style d % definition list, use {word}, see OpTeX trick 0108

```

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{|||lc|r||}{
  Month      & commodity & price  \crl
  January    & notebook  & \$ 700 \crli \tskip2pt
  February   & skateboard & \$ 100 \cr
  July       & yacht      & k\$ 170 \crl}

```

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 TeX. It marks the end of each row in the table. Moreover OpTeX defines following similar commands:

- `\crl` ... the end of the row with a horizontal line after it.
- `\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
\tabskipl=0pt \tabskipr=0pt % space before first and after last column

```

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

If there is an item that spans to more than one column in the table then the macro `\multispan{<number>}` (from Plain 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|}{ \cr1
\mspan3[|c|]{\bf Title} \cr1 \noalign{\kern\hhkern}\cr1i
first & second & third \cr1li
seven & eight & nine \cr1i}}
```

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|}}{\cr1p{3-8}
\mspan2[c|]{ } & \mspan3[c|]{Singular} & \mspan3[c|]{Plural} \cr1p{3-8}
\mspan2[c|]{ } & Neuter & Masculine & Feminine & Masculine & Feminine & Neuter \cr1
\vspan2{I} & Inclusive & \mspan3[c|]{\vspan2{0}} & \mspan3[c|]{X} \cr1p{2,6-8}
& Exclusive & \mspan3[c|]{ } & \mspan3[c|]{X} \cr1
\vspan2{II} & Informal & \mspan3[c|]{X} & \mspan3[c|]{X} \cr1p{2-8}
& Formal & \mspan6[c|]{X} \cr1
\vspan2{III} & Informal & \vspan2{0} & X & X & \mspan2[c|]{X} & \vspan2{0} \cr1p{2,4-7}
& Formal & & & & \mspan4[c|]{X} & \cr1
}
```

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).

		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				

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. OpTeX sets `\ttline=-1` by default.

The indentation of each line in display verbatim is controlled by `\ttindent` register. This register is set to the `\parindent` by default. Users can change the values of the `\parindent` and `\ttindent` independently.

The `\begtt` command starts the internal group in which the catcodes are changed. The group ends by `\endtt`. You can declare settings valid only inside this group at the same line where `\begtt` starts. For example:

```
\begtt \edef!{?} \edef?{!}
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 defined character as the given macro.

You can declare settings valid inside each `\begtt...\endtt` group by the `\everytt` register. There are tips for such global `\everytt` declarations 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
```

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

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

The `\begtt...\endtt` pair or `\verbatiminput` can be used for listings of codes. Automatic syntax highlighting is possible, for example `\begtt \hisyntax{C}` activates colors for C programs. Or `\verbatiminput \hisyntax{HTML}` (-) `file.html` can be used for HTML or XML codes. OpTeX implements syntax highlighting of C, Lua, Python, TeX, HTML, XML and more. For a declaration of a new language, see the section 2.28.2.

If the code is read by `\verbatiminput` 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 146) 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 don't have to 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, 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. The < character is sorted before A and > after Z and these characters aren’t printed if they are first in the index item.

### 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~{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 the other two possibilities documented in OPmac (using external BibT<sub>E</sub>X program) aren'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 commas 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 2.32.6 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, transparency

OpTeX provides a small number of color selectors: `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow`, `\Cyan`, `\Magenta`, `\White`, `\Grey`, `\LightGrey` and `\Black`. More such selectors can be defined by setting four CMYK components (using `\setcmykcolor`), or three RGB components (using `\setrgbcolor`) or one grey component (using `\setgreycolor`). For example

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

The color selectors work locally in groups like font selectors.

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 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. If `\onlyrgb` is set then `\colordef` works like `\rgbcolordef`.

The following example defines the macro for `colored text on colored background`. Usage: `\coloron<background><foreground>{<text>}`

The `\coloron` macro 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{Brown text on yellow background}
```

The `\transparency<number>` sets the transparency amount of following typesetting material until the current group is closed. The `<number>` must be in the range 0..255, zero means no transparency (solid objects), 255 means full transparency (invisible objects). You can see the effect when overlapping one object over another.

## 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 (or `\pich` in short) 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. More parameters can be included using the `\picparams` token list.

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 `TEX` to print labels in the same font as document font. `OpTEX` supports this feature by `\inkinspic {<filename>.pdf}` command. It reads and displays both: PDF image and 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 `TEX` and `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 `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.

<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.



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.

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

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

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

*text1* *text2* *text3*

You can see that TeX 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

moj.

The `\rotbox{<deg>}{<text>}` is shortcut for `\transformbox{\pdfrotate{<deg>}}{<text>}`.

#### 1.6.4 Ovals, circles

The `\inoval{<text>}` creates a box like this: text. 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 text. 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 \overlapmargins=N \hhkern=2pt \vbkern=2pt}
```

The macros `\clipinoval`  $\langle x \rangle$   $\langle y \rangle$   $\langle width \rangle$   $\langle height \rangle$   $\{\langle text \rangle\}$  and `\clipincircle` (with the same parameters) print the  $\langle text \rangle$  when a clipping path (oval or circle with given  $\langle width \rangle$  and  $\langle height \rangle$  shifted its center by  $\langle x \rangle$  to right and by  $\langle y \rangle$  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 right edge is used if the optional parameter `[r]` is given, see section 2.29 for more details.

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 `\langid` command in your document. For example `\delang` for German, `\cslang` for Czech, `\pllang` for Polish. The  $\langle lang-id \rangle$  is a shortcut of the language (mostly from ISO 639-1). You can list all available languages including their  $\langle lang-id \rangle$ 's by the `\langlist` macro. It prints now:

```
en(USenglish) enus(USenglishmax) engb(UKenglish) be(Belarusian) bg(Bulgarian) ca(Catalan) hr(Croatian) cs(Czech)
da(Danish) nl(Dutch) et(Estonian) fi(Finnish) fis(schoolFinnish) fr(French) de(nGerman) deo(oldGerman) gsw(swiss-
German) elm(monoGreek) elp(Greek) grc(ancientGreek) hu(Hungarian) is(Icelandic) ga(Irish) it(Italian) la(Latin)
lac(classicLatin) lal(liturgicalLatin) lv(Latvian) lt(Lithuanian) mk(Macedonian) pl(Polish) pt(Portuguese) ro(Ro-
manian) rm(Romansh) ru(Russian) srl(Serbian) src(SerbianCyril) sk(Slovak) sl(Slovenian) es(Spanish) sv(Swedish)
uk(Ukrainian) cy(Welsh) af(Afrikaans) hy(Armenian) as(Assamese) eu(Basque) bn(Bengali) nb(Bokmal) cop(Coptic)
cu(churchslavonic) eo(Esperanto) ethi(Ethiopic) fur(Friulan) gl(Galician) ka(Georgian) gu(Gujarati) hi(Hindi)
id(Indonesian) ia(Interlingua) kn(Kannada) kmr(Kurmanji) ml(Malayalam) mr(Marathi) mn(Mongolian) nn(Nynorsk)
oc(Occitan) or(Oriya) pi(Pali) pa(Panjabi) pms(Piedmontese) zh(Pinyin) sa(Sanskrit) ta(Tamil) te(Telugu) th(Thai)
tr(Turkish) tk(Turkmen) hsb(Uppersorbian) he(Hebrew)
```

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` $\{\text{Czech}\}$  works the same as `\cslang`. The `\uselanguage` parameter is case insensitive.

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

You can switch between language patterns by `\langiso-code` 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 `\langiso-code`. OpTeX declares these words only for few languages: Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak, Hebrew 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 2.37.2.

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 `\<iso-code>quotes`, then the control sequences `\"` and `\'` should be used like this: `\"<quoted text>"` or `\'<quoted text>'` (note that the terminating character is the same but it isn't escaped). This prints language-dependent normal or alternative quotes around *<quoted text>*. The language is specified by *<iso-code>*. OpTeX declares quotes only for Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English (`\csquotes`, `\dequotes`, ..., `\enquotes`). You can simply define your own quotes as shown in section 2.37.2. 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, scanbase]
```

does `\input qrcode.opm` and `\input scanbase.tex`. It saves local information about the fact that these file names (`qrcode`, `scanbase`) were loaded, 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 list of packages supported by OpTeX follows. Most of them are directly part of OpTeX:

- `math.opm` provides usable features for math typesetting and shows [how to create new packages](#).
- `qrcode.opm` enables to create QR codes.
- `tikz.opm` does `\input tikz.tex`, i.e. loads TikZ. It adds OpTeX-specific code.
- `mte.opm` includes settings for microtypographic extensions (protrusions+expanding fonts).
- `vlna.opm` enables to protect of one-letter prepositions and more things automatically.
- `emoji.opm` defines `\emoji{<name>}` command for colored emoticons.
- `minim-mp.opm` enables `\directmetapost` using `minim-mp` and `minim` packages.
- `pdfextra.opm` allows the use of many extra features from PDF standard (by M. Vlasák).

See these files in `optex/pkg/` or `optex/<pkgname>` for more information about them. The packages may have their documentation, try `texdoc <pkgname>`.

#### 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.

If the dot follows the argument before closing `]` (for example `\lipsum[3.]` or `\lipsum[3.1]`) then only first sentence from given paragraph is printed.

#### 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 :)
```

## 1.7.8 OpTeX tricks

The page [OpTeX tricks](#) shows many other features of OpTeX. They are of different nature and they are typically implemented by short chunks of macro code presented at the page.

Selected macros defined as an OpTeX trick can be used directly from your document without copying the code chunks into your macros. It is because these macros are “registered” in OpTeX (by `\_regtrick` internally) and if you use such a macro then OpTeX automatically loads the appropriate code chunk from an external file. These macros are listed here. More information about them are accessible via the external links.

- `\algol`, `\algolkw`, `\algolcap`, `\makeLOA`, `\algonum` enables to create pseudocode listings.
- `\beglua`, `\begLUA`, `\logginglua` writing LUA codes as LUA codes.
- `\cancel` prints a given text and the line/cross line over the text.
- `\clipbox` creates a box of given dimensions with clipped content.
- `\colortab` colored cells in the table.
- `\correctvsize` sets `\vsize` to fit lines exactly to pages.
- `\createfile`, `\begfile`, `\endfile` writes a code from the document to the given file.
- `\crtop`, `\crmid`, `\crbot` specific design of tables: only horizontal rules with different thickness.
- `\crx` alternating colored lines in tables.
- `\csvread` reads CSV databases and provides other macros for working with their data.
- `\dbitem`, `\dbwhere`, `\dbloop`, `\dbsort`, `\dbcreate`, `\dbinsert`, `\dbshow` etc., works with databases.
- `\directchar` prints the character directly, bypasses the ligature processing.
- `\directoutput` puts boxes to standalone pages adapted to the box dimensions.
- `\easylist` the depth of list is given by the number of `*`.
- `\fcread`, `\fullcite` citations by full bibliographic records.
- `\framedblocks` redefines `\begblock`, `\endblock` to create blocks in frames splittable to pages.
- `\ignoreinspic` the `\inspic` commands stop loading images, they are replaced by gray frames.
- `\import` allows to have subsets of document input files in separate directories.
- `\inspicto` behaves like `\inspic`, but saves PDF size when image is printed repeatedly.
- `\ismatch`, `\isinmacro` tests, if the given string or the macro body includes the given substring.
- `\ispageodd` tests, if the current point is at odd page regardless of asynchronous processing.
- `\incrrp`, `\thepp`, `\thepplast`, `\truepage` does per-page counting of objects.
- `\keystroke` prints given text in a keystroke-like frame.
- `\longtable` allows to break a table to more pages and repeats header.
- `\makeLOF`, `\makeLOT`, `\captionF`, `\captionT` create list of tables and list of figures similar to `\maketoc`.
- `\mathinexpr` allows short names `\sin(x)`, `\log(x)` instead `\math.sin(x)`, `\math.log(x)` in `\expr` arguments.
- `\onlyifnew` only define a macro if it is not already defined.
- `\pgforeground` adds material to the foreground of each page.
- `\pstart`, `\pend` displays line numbers of the marked text in the margin.
- `\rebox` modifies the vbox: its width will be equal to the wider line.
- `\replmacro` enables to patch existing macros using regular expression rules.
- `\roundframe` colored frames with rounded corners and many options.
- `\runsystem` runs the given external system command.
- `\shadedframe` colored rectangular frames with simple shadows.
- `\scaleto`, `\scaletoF` text font size changed to the desired width.
- `\seccc`, `\iniseccc` implements new level of subsections.
- `\shownodes` prints the list of nodes to the terminal.
- `\sethours`, `\setminutes`, `\setseconds`, `\setweekday` printing time, date, and day of week.
- `\style m`, `\keepstyle` creates lists with items numbered like subsections.
- `\settabs`, `\tabs` macros emulate tabulators of old typewriters.
- `\showpglists` shows good organized list of nodes of given pages to the log file.
- `\tabnodes` positions of table items are nodes, they can be used for drawing.
- `\tdnum` expands to the three-digits-group format of the given number.
- `\tnote` creates notes for table data printed just after the table.
- `\treedata`, `\tracingtreedata`, `\treenodes` `\showtree`, `\jsonread`, support tree data structures.
- `\ttlineref` verbatim lines referenced in text.
- `\vcent`, `\vbot` prints paragraphs in tables vertically centered or placed at bottom.
- `\thedimen` prints `\dimen` value using selected unit.
- `\twoblocks` allows printing bilingual texts in two columns veritically aligned.
- `\verbhyphens` setting breakable points for inline verbatim.
- `\xparshape` behaves like `\parshape` but its declaration is valid for more paragraphs.
- `\xreplstring`, `\ereplstring` behave like `\replstring` but are expandable and add more features.



## 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 list 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] % loadaing macro files
\fontfam [FamilyName] % selection of font family
\typoysize [font-size/baselineskip] % size setting of typesetting
\typoscale [factor-font/factor-baselineskip] % size scaling
\thefontsize [size] \thefontscale [factor] % current font size

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

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

\hyperlinks {color-in}{color-out} % PDF links activate as clickable
\outlines {level} % PDF will have a table of contents in the left tab
\magscale [factor] % resize typesetting, line/page breaking unchanged
\margins/pg format (left, right, top, bottom)unit % margins setting
\report \letter \slides % style declaration macros
```

## 1.9 API for macro writers

All T<sub>E</sub>X primitives and almost all OpT<sub>E</sub>X macros are accessible by two names: `\foo` (public or user namespace) and `\_foo` (private name space). For example `\hbox` and `\_hbox` means the same T<sub>E</sub>X primitive. More about it is documented in section 2.2.1.

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 `\_codedecl` macro below, so this macro is not available as `\codedecl`.

If you are writing a document or macros specific for the document, then use simply public namespace (`\foo`). If you are writing more general macros, then you should declare your own namespace by `\_namespace` macro and you have to follow the naming discipline described in sections 2.2.1 and 2.2.3.

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, `\aheadto \macro{<text>}` puts `<text>` at the begin.  
`\edef <char>{<body>}` defines `<char>` active character with meaning `<body>`. See also `\mathdef`.  
`\afterfi {<text>}{<ignored>}\fi` expands to `\fi<text>`, see also `\afterxfi`.  
`\basefilename \currfile` returns the name of the file currently read.  
`\bp {<dimen expression>}` expands `TeX` dimension to decimal number in bp without unit.  
`\casesof <token> <list of cases>` expands to a given case by the given `<token>`. See also `\qcasesof`, `\xcasesof`.  
`\_codedecl <sequence> {<info>}` is used at beginning of macro files.  
`\colordef \macro {<mix of colors>}` declares `\macro` as color switch.  
`\cs {<string>}` expands `\<string>`.  
`\cstochar <sequence>` converts `<sequence>` to `<character>` if there was `\let<sequence>=<character>`.  
`\_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`, `\istokempty`, `\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 a value given by `key=value`. See also `\trykv`, `\iskv`, `\readkv`, `\kvx`, `\nokvx`.  
`\loop ... \repeat` is classical Plain `TeX` loop. `\xloop ... \do ... \repeat` is an extension of `\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 `TeX` allocators.  
`\newif \iffoo` declares boolean `\iffoo` as in Plain `TeX`.  
`\_newifi \_iffoo` declares boolean `\_iffoo`.  
`\nospaceafter\macro`, `\nospaceafter{<text>}`, `\nospacefuturelet`: they ignore the following optional space.  
`\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.  
`\posx[<label>]`, `\posy[<label>]`, `\pospg[<label>]` provide coordinates of absolute position of the `\setpos[<label>]`.  
`\private <sequence> <sequence> ...` ; declares `<sequence>`s for private name space.  
`\public <sequence> <sequence> ...` ; declares `<sequence>`s for public name space.  
`\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 `\f@t` are removed and mostly replaced by control sequences prefixed by `_` (like `\_this`). Only a basic set of old Plain T<sub>E</sub>X control sequences like `\p@`, `\z@`, `\dimen@` are provided but not recommended for new macros.

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). Users should re-define unprefixed forms of control sequences without worries that something internal will be broken.

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 secant<sup>9</sup>.

## 1.11 Related documents

- [Typesetting math with OpT<sub>E</sub>X](#) – More details about math typesetting.
- [T<sub>E</sub>X in a Nutshell](#) – Summary about T<sub>E</sub>X principles, T<sub>E</sub>X primitive commands etc.
- [OpT<sub>E</sub>X catalog](#) – All fonts collected to `\fontfam` families are shown here.
- [OMLS](#) – OpT<sub>E</sub>X Markup Language Standard.
- [OpT<sub>E</sub>X - tips, tricks, howto](#) – Tips of macro codes for various purposes.

---

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

<sup>9</sup> Use `\$secant(x)$` to get `sec(x)`.

## 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{1.20 May 2026}
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 `\_codedecl` will be re-defined later.

```

39 % Basic .opm syntax:
40
41 \let\_endcode =\endinput
42 \def\_codedecl #1#2{\immediate\write-1{#2}}% 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 unimath-macros.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 languages.opm        % Languages macros
88 \input lang-decl.opm        % Languages declaration
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{\_banner^^J}%
98   \directlua{lua.bytecode[1]()}% load OpTeX's Lua code
99   \mathsbon % replaces \int_a^b to \int _a^b
100   \inputref % inputs \jobname.ref if exists
101 }
102
103 \dump % You can redefine \dump if additional macros are needed. Example:
104       % \let\dump=\relax \input optex.ini \input mymacros \_dump

```

## 2.2 Basic principles of OpTeX sources

### 2.2.1 Concept of namespaces of control sequences

OpTeX sets the category code of the “`_`” character to 11 (letter) and it is never changed.<sup>1</sup> So, we can always construct multiletter control sequence names from letters A–Z, a–z, and `_`. The “letter `_`” works in math mode as a subscript constructor because it is set as math active character (see section 2.15).

We distinguish following namespaces for multiletter control sequences:

- Only alphabetical names are in the *public namespace*. They are intended for end users when creating a document. Sometimes it is called *user namespace* too. For example `\hbox`, `\fontfam`, `\MyMacro`.
- Only alphabetical lowercase names prefixed by single “`_`” are in the *private namespace*. It is used in OpTeX internal macros. For example `\_hbox`, `\_fontsel`.
- Names in the form `\_⟨pkg⟩_⟨name⟩` are in the *package namespace*, see section 2.2.3. For example `\_qr_size`, `\_math_alist`.
- Names starting with two “`_`” are in the *reserved namespace*. They can be used for internal control sequences in font family files or in similar cases.
- Other names which include “`_`” but not as the first character can be used too, but with care, see the end of this section.

All TeX primitives are initialized with two control sequences with the same meaning: *prefixed* control sequence (in private namespace, for example `\_hbox`) and *unprefixed* control sequence (in public namespace, for example `\hbox`). All OpTeX macros intended for end users are initialized in these two forms too, for example `\_ref` and `\ref`.

Users can declare any control sequences in the public namespace without worrying that OpTeX behavior is changed. This is because OpTeX uses exclusively prefixed control sequences in its macros. For example, a user can declare `\def\fi{finito}` and nothing bad happens, if the user doesn’t use `\fi` in its original primitive meaning. You don’t have to know all TeX primitives and OpTeX macros, you can declare control sequences for your use in the public namespace without limitations and nothing bad will happen.

You can use control sequences from private or package namespace in a “read-only manner” without changing OpTeX behavior too. On the other hand, if you re-define a control sequence in the private namespace, the OpTeX behavior can be changed. You can do it but we suppose that you know what you are doing and what OpTeX behavior is changed.

All multiletter control sequences declared by OpTeX are defined in the private namespace first (`\_def\_macro{...}`). If the declared control sequences are intended for end users too then they are exported to the public namespace after that. It is done by the `\public` macro:

`\public <list of control sequences> ;`

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

There is an exception of the above mentioned principle. Control sequences which are alternatives to math characters (`\alpha`, `\forall`, `\subset` etc.) are declared only in public namespace if they are not used in any internal OpTeX macros.

<sup>1</sup> This is only singular exception from category codes given by plain TeX.

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 unprefix 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 unprefix), then the accompanying text mentions only the unprefix variant. The code typically defines the prefixed variant and then the `\public` (or `\_public`) macro is used.

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).

It is very tempting to use control sequence names with `_` in order to distinguish more words in the sequence name. If the first character isn't `_` then such a name is outside private and package namespaces, so they can be used for various purposes. For example `\my_control_sequence`. But there is an exception: 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 feature is activated because we want to write math formulae as in plain TeX, 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}
```

It 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.

## 2.2.2 Macro files syntax

Segments of OpTeX macros or external macro packages are stored in files 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 {\<short title> <\<version>>}
```

If the mentioned `\sequence` is undefined then `\_codedec1` prints the message

```
@:[\<file name>] \<short title> <\<version>>
```

to the log file and TeX continues with reading the following macros. If the `\sequence` is defined, then `\_codedec1` acts like `\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 definitions. The documentation text should be here. It is ignored when macros are read.

The `\_doc ... \_cod` parts can be printed after `\load[doc]` using `\printdoc` macro, see section 2.40. If you have created a documented macro file `pkgname.opm` then you can put macros for creating your documentation between first pair of `\_doc ... \_cod` used after `\_endcode`. These macros should `\load[doc]` and must be finished by `\bye`. Then you have code+documentation together in a single file and user can generate the documentation of your package by `\docgen` used at command line:

```
optex -jobname pkgname-doc '\docgen pkgname'
```

Example of a `\_doc ... \_cod` code used for creating the documentation using `\docgen` can be found in the `math.opm` file. You can see [its documentation](#), especially [section about creating packages](#).

## 2.2.3 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>2</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

<sup>2</sup> We have not adopted the idea from expl3 language:)

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.

`\_nspublic <list of sequences>` ; does `\let\foo = \_⟨pkg⟩_foo` for each given sequence 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 public namespace by `\_nspublic \macro;`.

It could happen that a package writer needs to declare a control sequence (say `\foo`) directly without setting it in `\_⟨pkg⟩_foo` namespace followed by using `\_nspublic`. The `\_newpublic` prefix should be used in this case, for example `\_newpublic\def\foo` or `\_newpublic\chardef\foo` or `\_newpublic{\_long\def}\foo`. The `\_newpublic⟨do⟩\⟨sequence⟩` prints a warning if the declared `\⟨sequence⟩` is defined already and then runs `⟨do⟩\⟨sequence⟩`. The reason of the warning is the same as when `\_nspublic` warns about doing re-declaration of control sequences already declared.

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).

## 2.2.4 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 public namespace in the macro bodies if there is no explicit and documented reason to do this.
- Don't declare control sequences in the public namespace if there are no explicit and documented reasons to do this.
- 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 2.2.3.
- 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 `math.opm` is a good example of how an external macro file for OpTeX can look like. Another good and short example is [here](#).

## 2.2.5 The implementation of the namespaces and macros for macro-files

```
3 \_codedecl \public {Prefixing and code syntax <2024-02-02>} % preloaded in format
```

prefixed.opm

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



```

9 \let\directlua = \directlua
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.

`\newpublic<do>\<sequence>` prints warning if `\<sequence>` is declared already. Then runs `<do>\<sequence>`.

`\_checkexists <where> {\<sequence-string>}` prints error if the control sequence given by its name `<sequence-string>` is not declared. This check is used in `\public`, `\private`, `\_nspublic` and `\_nsprivate` macros in order to avoid mistakes in names when declaring new control sequences.

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

```

42 \_let\_ea =\_expandafter % useful shortcut
43
44 \_long\_def \_xargs #1#2{\_ifx #2;\_else \_ea#1\_ea#2\_ea\_xargs \_ea #1\_fi}
45
46 \_def \_pkglabel{}
47 \_def \_public {\_xargs \_publicA}
48 \_def \_publicA #1{%
49   \_checkexists \_public {\_csstring#1}%
50   \_ea\_let \_ea#1\_csname \_csstring #1\_endcsname
51 }
52 \_def \_private {\_xargs \_privateA}
53 \_def \_privateA #1{%
54   \_checkexists \_private {\_csstring #1}%
55   \_ea\_let \_csname \_csstring #1\_endcsname =#1%
56 }
57 \_def\_checkexists #1#2{\_unless \_ifcsname #2\_endcsname
58   \_errmessage {\_string#1: \_bslash#2 must be declared}\_fi
59 }
60 \_def\_newpublic #1#2{\_unless\_ifx #2\_undefined
61   \_opwarning{\_string#2 is redefined%
62     \_ifx\_pkglabel\_empty \_else\_space by the \_ea\_ignoreit\_pkglabel\_space package\_fi}\_fi
63   #1#2%
64 }
65 \_public \_public \_private \_newpublic \_xargs \_ea ;

```

We define the macros `\_namespace {\<pkg label>}`, `\_resetnamespace {\<pkg label>}`, `\_endnamespace`, `\_pkglabel`, `\_nspublic`, and `\_nsprivate` for package writers, see section 2.2.3.

```

74 \_def \_pkglabel{}
75 \_def\_namespace #1{%
76   \_ifcsname \_namesp:#1\_endcsname \_errmessage
77     {The name space "#1" is used already, it cannot be used twice}%
78   \_endinput
79   \_else
80     \_ea \_gdef \_csname \_namesp:#1\_endcsname {}%
81     \_resetnamespace{#1}\_fi
82 }
83 \_def\_resetnamespace #1{%
84   \_unless \_ifx \_pkglabel\_empty \_endnamespace \_fi
85   \_gdef \_pkglabel{#1}%
86   \_directlua{
87     callback.add_to_callback("process_input_buffer",
88       function (str)
89         return string.gsub(str, "\_nbb[.][a-zA-Z]", "\_nbb \_#1\_pcent 1")
90       end, "\_namespace")
91   }%
92 }
93 \_def\_endnamespace {%

```

```

94 \directlua{ callback.remove_from_callback("process_input_buffer", "_namespace") }%
95 \gdef \pkglabel{}%
96 }
97 \def \nspublic {\xargs \nspublicA}
98 \def \nspublicA #1{%
99 \checkexists \nspublic {\pkglabel \_csstring #1}%
100 \ifcsname _eol:\_ea\ignoreit\pkglabel \_csstring #1\endcsname % defined by \eoldef
101 \slet {_eol:\_csstring #1}{_eol:\_ea\ignoreit\pkglabel \_csstring #1}\_fi
102 \ea\_newpublic \ea\_let \ea#1\_csname \pkglabel \_csstring #1\endcsname
103 }
104 \def \nsprivate {\xargs \nsprivateA}
105 \def \nsprivateA #1{%
106 \checkexists \nsprivate {\_csstring #1}%
107 \ea\_let \_csname \pkglabel \_csstring #1\endcsname =#1%
108 }

```

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 the `<text>` to the terminal and to the `.log` file, `\wlog {<text>}` prints the `<text>` only to the `.log` file (as in plain T<sub>E</sub>X)

```

120 \def \_codedecl #1#2{%
121 \ifx #1\undefined \wlog{@:[\_basefilename\_currfile] #2}%
122 \else \ea\_endinput \_fi
123 }
124 \def \wterm {\_immediate \_write16 }
125 \def \wlog {\_immediate\_write-1 } % write on log file (only)
126
127 \_public \wterm \wlog ;

```

prefixed.opm

`\currfile` returns the name of the current input file including its path.

`\basefilename\currfile` returns base name of the current file, without its path and extension.

`\_nofilepath <text>/<with>/<slashes>/\_fin` expands to the last segment separated by slashes.

`\_nofileext <filename>.\_fin` expands to the file name without extension.

```

138 \def\_currfile{\directlua{tex.print(status.filename)}}
139 \def\_basefilename #1{\_ea\_nofileext\_expanded{\_ea\_ea\_ea\_nofilepath#1/\_fin}.\_fin}
140 \def\_nofilepath #1/#2{\_ifx#2\_fin #1\_else \_ea\_nofilepath \_ea#2\_fi}
141 \def\_nofileext #1.#2\_fin{#1}
142
143 \_public \currfile \basefilename ;

```

prefixed.opm

We define `\_fin` as a useless macro. Suppose that its meaning will be never used for another control sequence. You can use `\_fin` as a final delimiter of a list of tokens and your macro can ask `\ifx\_fin#1` in order to decide that the list of tokens is finalized.

```

152 \_protected\_long \def \_fin \_fin {}

```

prefixed.opm

## 2.3 pdfT<sub>E</sub>X initialization

Common pdfT<sub>E</sub>X primitives equivalents are declared here. Initial values are set.

```

3 \_codedecl \pdfprimitive {LuaTEX initialization code <2024-02-29>} % 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

```

luatex-ini.opm

```

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\_pdfinsetht \insetht
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 \pdfinsetht \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 \pdfglyptounicode {\_pdfextension glyptounicode }
82 \_protected\_def \pdfrunninglinkoff {\_pdfextension linkstate 1 }
83 \_protected\_def \pdfrunninglinkon {\_pdfextension linkstate 0 }
84
85 \_protected\_edef\_pdfcompresslevel {\_pdfvariable compresslevel}

```

```

86 \_protected\_edef\_pdfobjcompresslevel {\_pdfvariable objcompresslevel}
87 \_protected\_edef\_pdfdecimaldigits {\_pdfvariable decimaldigits}
88 \_protected\_edef\_pdfgamma {\_pdfvariable gamma}
89 \_protected\_edef\_pdfimageresolution {\_pdfvariable imageresolution}
90 \_protected\_edef\_pdfimageapplygamma {\_pdfvariable imageapplygamma}
91 \_protected\_edef\_pdfimagegamma {\_pdfvariable imagegamma}
92 \_protected\_edef\_pdfimagehicolor {\_pdfvariable imagehicolor}
93 \_protected\_edef\_pdfimageaddfilename {\_pdfvariable imageaddfilename}
94 \_protected\_edef\_pdfpkresolution {\_pdfvariable pkresolution}
95 \_protected\_edef\_pdfinclusioncopyfonts {\_pdfvariable inclusioncopyfonts}
96 \_protected\_edef\_pdfinclusionerrorlevel {\_pdfvariable inclusionerrorlevel}
97 \_protected\_edef\_pdfgentounicode {\_pdfvariable gentounicode}
98 \_protected\_edef\_pdfpagebox {\_pdfvariable pagebox}
99 \_protected\_edef\_pdfmajorversion {\_pdfvariable majorversion}
100 \_protected\_edef\_pdfminorversion {\_pdfvariable minorversion}
101 \_protected\_edef\_pdfuniqueresname {\_pdfvariable uniqueresname}
102 \_protected\_edef\_pdfhorigin {\_pdfvariable horigin}
103 \_protected\_edef\_pdfvorigin {\_pdfvariable vorigin}
104 \_protected\_edef\_pdflinkmargin {\_pdfvariable linkmargin}
105 \_protected\_edef\_pdfdestmargin {\_pdfvariable destmargin}
106 \_protected\_edef\_pdfthreadmargin {\_pdfvariable threadmargin}
107 \_protected\_edef\_pdfpagesattr {\_pdfvariable pagesattr}
108 \_protected\_edef\_pdfpageattr {\_pdfvariable pageattr}
109 \_protected\_edef\_pdfpageresources {\_pdfvariable pageresources}
110 \_protected\_edef\_pdfxformattr {\_pdfvariable xformattr}
111 \_protected\_edef\_pdfxformresources {\_pdfvariable xformresources}
112 \_protected\_edef\_pdfpkmode {\_pdfvariable pkmode}
113
114 \_public
115 \pdftexversion \pdftexrevision \pdflastlink \pdfretval \pdflastobj
116 \pdflastannot \pdfxformname \pdfcreationdate \pdffontname \pdffontobjnum
117 \pdffontsize \pdfpageref \pdfcolorstackinit \pdfliteral \pdfcolorstack
118 \pdfsetmatrix \pdfsave \pdfrestore \pdfobj \pdfrefobj \pdfannot
119 \pdfstartlink \pdfendlink \pdfoutline \pdfdest \pdfthread \pdfstartthread
120 \pdfendthread \pdfinfo \pdfcatalog \pdfnames \pdfincludechars \pdffontattr
121 \pdfmapfile \pdfmapline \pdftrailer \pdfglyphtounicode
122 \pdfcompresslevel \pdfrunninglinkoff \pdfrunninglinkon
123 \pdfobjcompresslevel \pdfdecimaldigits \pdfgamma \pdfimageresolution
124 \pdfimageapplygamma \pdfimagegamma \pdfimagehicolor \pdfimageaddfilename
125 \pdfpkresolution \pdfinclusioncopyfonts \pdfinclusionerrorlevel
126 \pdfgentounicode \pdfpagebox \pdfmajorversion \pdfminorversion \pdfuniqueresname
127 \pdfhorigin \pdfvorigin \pdflinkmargin \pdfdestmargin \pdfthreadmargin \pdfpagesattr
128 \pdfpageattr \pdfpageresources \pdfxformattr \pdfxformresources \pdfpkmode ;
129
130 \_pdfmajorversion = 1
131 \_pdfminorversion = 7 % was 5, see https://tug.org/pipermail/tex-live/2025-February/
132 \_pdfobjcompresslevel = 2
133 \_pdfcompresslevel = 9
134 \_pdfdecimaldigits = 3
135 \_pdfpkresolution = 600

```

## 2.4 Basic macros

We define first bundle of basic macros.

```

3 \_codedecl \sdef {Basic macros for OpTeX <2026-04-14>} % preloaded in format

```

basic-macros.opm

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

basic-macros.opm

```

10 \_let\bgroup={ \_let\egroup=}
11 \_def \_empty {}
12 \_def \_space { }
13 \_def \_null {\_hbox{}}
14 \_public \bgroup \egroup \empty \space \null ;

```

`\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.

```

23 \_long\_def \_ignoreit #1{
24 \_long\_def \_useit #1{#1}
25 \_long\_def \_ignoresecond #1#2{#1}
26 \_long\_def \_usesecsecond #1#2{#2}
27 \_public \_ignoreit \_useit \_ignoresecond \_usesecsecond ;

```

`\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.

```

36 \_edef \bslash {\_csstring\\}
37 \_edef \nbb {\_bslash\_bslash}
38 \_edef \pcent{\_csstring\%}
39 \_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 }.

`\sxddef {⟨text⟩}` is equivalent to `\xdef\⟨text⟩`.

`\slet {⟨textA⟩}{⟨textB⟩}` is equivalent to `\let \⟨textA⟩ = \⟨textB⟩`.

```

51 \_def \_sdef #1{\_ea\_def \_csname#1\_endcsname}
52 \_def \_sxddef #1{\_ea\_xdef \_csname#1\_endcsname}
53 \_def \_slet #1#2{\_ea\_let \_csname#1\_ea\_endcsname
54 \_ifcsname#2\_ea\_endcsname \_begincsname#2\_endcsname \_else \_undefined \_fi
55 }
56 \_public \_sdef \_sxddef \_slet ;

```

`\adef ⟨char⟩{⟨body⟩}` defines active `⟨char⟩` as `⟨body⟩` and then puts the `⟨char⟩` as active character. I.e. the `⟨body⟩` can include the `⟨char⟩` as non-active character (if it is non-active before `\adef`). For example `\adef ?{\,?}`. If the character is special, you can escape it, for example `\adef \%{...}`. The space can be declared by `\adef { }{⟨body⟩}`. You can declare a macro with parameters too, for example `\adef @#1{...#1...}`. You can use prefixes `\protected`, `\global`, `\long` before `\adef`, they behave like prefixes before `\def`.

`\mathadef ⟨char⟩{⟨body⟩}` defines active `⟨char⟩` only in math mode. The catcode of the `⟨char⟩` is kept.

```

72 \_def\_adef#1#2#{\_adefA{#1}{#2}}
73 \_def\_adefA#1#2#3{\_ea\_def\_directlua{tex.cprint(13, "\_luaescapestring{\_csstring#1}")}}#2{#3}%
74 \_catcode`#1=13 }
75 \_def\_mathadef#1{\_mathcode`#1="8000 \_bgroup \_lccode`~="#1 \_lowercase{\_egroup \_def~}}
76 \_public \_adef \_mathadef ;

```

`\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⟩`.

```

86 \_def \_cs #1{\_csname#1\_endcsname}
87 \_def \_trycs#1#2{\_ifcsname #1\_endcsname \_csname #1\_ea\_endcsname \_else \_afterfi{#2}\_fi}
88 \_public \_cs \_trycs ;

```

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

`\aheadto \macro{⟨text⟩}` defines `\macro` as `⟨text⟩` followed by the original `\macro` body.

```

96 \_long\_def \_addto #1#2{\_ea\_def\_ea#1\_ea{#1#2}}
97 \_long\_def \_aheadto #1#2{\_edef#1{\_unexpanded{#2}\_unexpanded\_ea{#1}}}
98
99 \_public \_addto \_aheadto ;

```

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

```

106 \_def\_incr #1{\_global\_advance#1by1 }
107 \_def\_decr #1{\_global\_advance#1by-1 }
108 \_public \_incr \_decr ;

```

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

```

114 \_def \_opwarning #1{\_wterm{WARNING 1.\_the\_inputlineno: #1.}}
115 \_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, because eTeX and pdfTeX extensions are used.

`\tracingnone` sets the tracing registers to their default zero value.

basic-macros.opm

```
125 \def\loggingall{%
126   \tracingstats=2 \tracingpages=1
127   \tracingoutput=1 \tracingmacros=3 % \tracinglostchars=2 is already set
128   \tracingparagraphs=1 \tracingscantokens=1 \tracingifs=1 \tracinggroups=1
129   \tracingcommands=3 \tracingrestores=1 \tracingassigns=1 }
130 \def\tracingall{\tracingonline=1 \loggingall}
131 \def\tracingnone{\tracingassigns=0 \tracingcommands=0 \tracingmacros=0 \tracingifs=0
132   \ea \settracingzero \loggingall \tracingonline=0 }
133 \def\settracingzero #1=#2 {#1=0 \ifnum#2>0 \ea \settracingzero \fi}
134 \public \loggingall \tracingall \tracingnone ;
```

The `\optexversion` and `\fmtname` are defined in the `optex.ini` file. Maybe, somebody will need a private version of these macros. We add `\_banner` used in `\everyjob` and in `\docgen`

basic-macros.opm

```
142 \def\_banner {This is OpTeX (Olsak's Plain TeX), version <\optexversion>}%
143 \private \optexversion \fmtname ;
```

`\_byehook` is used in the `\bye` macro. It writes a warning if the user did not load a Unicode Font and writes a “rerun” warning if the `.ref` file was newly created or it was changed (compared to the previous TeX run). The last `\message{^^J}` separates the usable log info from the following unusable font-loading info, which (unfortunately) cannot be deactivated on the terminal.

basic-macros.opm

```
154 \def\_byehook{%
155   \ifx\_initunifonts\_relax \_relax\_else \_opwarning{Unicode font was not loaded}\_fi
156   \immediate\_closeout\_reffile
157   \edef\_tmp{\mdfive{\_jobname.ref}}%
158   \ifx\_tmp\_prevrefhash\_else \_opwarning{Try to rerun,
159     \_jobname.ref file was \_ifx\_prevrefhash\_empty created\_else changed\_fi}\_fi
160   \message{^^J}%
161 }
```

## 2.5 Allocators for TeX registers

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

Unlike in PlainTeX, 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 plainTeX, 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 \codedec1 \newdimen {Allocators for registers <2023-02-03>} % preloaded in format
```



The limits are set first.

```

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
18 \_chardef\_mailanguage = 16380 % In fact 16383, but we reserve next numbers for dummy patterns

```

alloc.opm

Each allocation macro needs its own counter.

```

24 \_countdef\_countalloc=10    \_countalloc=255
25 \_countdef\_dimenalloc=11    \_dimenalloc=255
26 \_countdef\_skipalloc=12     \_skipalloc=255
27 \_countdef\_muskipalloc=13   \_muskipalloc=255
28 \_countdef\_boxalloc=14      \_boxalloc=255
29 \_countdef\_toksalloc=15     \_toksalloc=255
30 \_countdef\_readalloc=16     \_readalloc=-1
31 \_countdef\_writealloc=17    \_writealloc=0 % should be -1 but there is bug in new luatex
32 \_countdef\_famalloc=18      \_famalloc=42  % \_newfam are 43, 44, 45, ...
33 \_countdef\_languagealloc=19 \_languagealloc=0

```

alloc.opm

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

```

43 \_def\_allocator #1#2#3{%
44   \_incr{\_cs{#2alloc}}%
45   \_ifnum\_cs{#2alloc}>\_cs{\_mai#2}%
46     \_errmessage{No room for a new \_ea\_string\_csname #2\_endcsname}%
47   \_else
48     \_global#3#1=\_cs{#2alloc}%
49     \_wloga{\_string#1=\_ea\_string\_csname #2\_endcsname\_the\_cs{#2alloc}}%
50   \_fi
51 }
52 \_let\_wloga=\_wlog % you can suppress the logging by \_let\_wloga=\_ignoreit

```

alloc.opm

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

```

61 \_def\_newcount #1{\_allocator #1{count}\_countdef}
62 \_def\_newdimen #1{\_allocator #1{dimen}\_dimendef}
63 \_def\_newskip #1{\_allocator #1{skip}\_skipdef}
64 \_def\_newmuskip #1{\_allocator #1{muskip}\_muskipdef}
65 \_def\_newbox #1{\_allocator #1{box}\_chardef}
66 \_def\_newtoks #1{\_allocator #1{toks}\_toksdef}
67 \_def\_newread #1{\_allocator #1{read}\_chardef}
68 \_def\_newwrite #1{\_allocator #1{write}\_chardef}
69 \_def\_newfam #1{\_allocator #1{fam}\_chardef}
70 \_def\_newlanguage #1{\_allocator #1{language}\_chardef}
71
72 \_public \newcount \newdimen \newskip \newmuskip \newbox \newtoks
73         \newread \newwrite \newfam \newlanguage ;

```

alloc.opm

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

```

79 \_newcount\_insertalloc    \_insertalloc=255
80 \_chardef\_insertmin = 201
81
82 \_def\_newinsert #1{%
83   \_decr\_insertalloc
84   \_ifnum\_insertalloc < \_insertmin
85     \_errmessage {No room for a new \_string\_insert}%
86   \_else
87     \_global\_chardef#1=\_insertalloc
88     \_wlog {\_string#1=\_string\_insert\_the\_insertalloc}%

```

alloc.opm

```

89  \_fi
90  }
91  \_public \newinsert ;

```

Other allocation macros `\newmarks`, `\newattribute` and `\newcatcodetable` have their counter allocated by the `\newcount` macro. `\_noattr` is constant `-"7FFFFFFF`, i.e. unused attribute

alloc.opm

```

99  \_newcount \marksalloc \marksalloc=0 % start at 1, 0 is \mark
100 \_chardef \maimarks=\maicount
101 \_def \newmarks #1{\_allocator #1{marks}\_chardef}
102
103 \_newcount \attributealloc \attributealloc=0
104 \_chardef \maiattribute=\_numexpr \maicount -1\_relax
105 \_attributedef \_noattr \maicount
106 \_def \newattribute #1{\_allocator #1{attribute}\_attributedef}
107
108 \_newcount \catcodetablealloc \catcodetablealloc=10
109 \_chardef \maicatcodetable=32767
110 \_def \newcatcodetable #1{\_allocator #1{catcodetable}\_chardef}
111
112 \_public \newmarks \newattribute \newcatcodetable ;

```

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

alloc.opm

```

119 \_newcount \tmpnum \_newcount \_tmpnum
120 \_newdimen \tmpdim \_newdimen \_tmpdim

```

A few registers: `\maxdimen`, `\hideskip` and `\centering` are initialized like in plain $\TeX$ . We absolutely don't support the `@category` dance, so `\z@skip` `\z@`, `\p@` etc. are defined but not recommended. The `\_zo`, `\_zoskip` and `\voidbox` (equivalents to `\z@`, `\z@skip` and `\voidb@x`) are preferred in Op $\TeX$ .

alloc.opm

```

131 \_newdimen \_maxdimen \_maxdimen=16383.999999pt % the largest legal <dimen>
132 \_newskip \_hideskip \_hideskip=-1000pt plus 1fill % negative but can grow
133 \_newskip \_centering \_centering=0pt plus 1000pt minus 1000pt
134 \_newdimen \_zo \_zo=0pt
135 \_newskip \_zoskip \_zoskip=0pt plus 0pt minus 0pt
136 \_newbox \_voidbox % permanently void box register
137
138 \_public \_maxdimen \_hideskip \_centering \_voidbox ;

```

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

if-macros.opm

```

3 \_codedecl \newif {Special if-macros, is-macros and loops <2025-03-01>} % preloaded in format

```

### 2.6.1 Classical `\newif`

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

The macro `\_newif` enables to declare `\_ifxxx` and to use `\_xxxtrue` and `\_xxxfalse`. This means that it is usable for the internal namespace (`\_prefixed` macros).

if-macros.opm

```

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`  $\langle\textit{what to do}\rangle\langle\textit{ignored}\rangle\backslash\textit{fi}$  closes condition by `\fi` and processes  $\langle\textit{what to do}\rangle$ . Usage:

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

`\afterxfi`  $\langle\textit{what to do}\rangle\langle\textit{ignored}\rangle\backslash\textit{xfi}$  closes all opened `\if.. \fi` conditionals until `\xfi` is found and then processes  $\langle\textit{what to do}\rangle$ . The  $\langle\textit{ignored}\rangle$  text must expand to nothing (or spaces only); arbitrary number of opened conditionals can be closed here. Example:

```
\ifodd#1 \ifodd#2 \afterxfi{odd-odd} \else \afterxfi{odd-even} \fi
\else \ifodd#2 \afterxfi{even-odd} \else \afterxfi{even-even} \fi
\fi \xfi
```

Nested `\if.. \afterfi{ \if.. \afterfi{...} \fi} \fi` are possible. Another approach is mentioned in [OpTeX trick 0098](#) which also solves the `\fi` in `\if` problem.

`if-macros.opm`

```
52 \_long\_def \_afterfi#1#2\_fi{\_fi#1}
53 \_long\_def \_afterfi#1#2\_fi{\_fi#1}
54 \_let\_xfi=\_empty \_let\_xfi=\_empty % \xfi alone does nothing
55 \_long\_def \_afterxfi#1#2\_xfi{\_romannumeral#20 #1}
56 \_long\_def \_afterxfi#1#2\_xfi{\_romannumeral#20 #1}
```

## 2.6.2 Loops

The `\loop`  $\langle\textit{codeA}\rangle$  `\ifsomething`  $\langle\textit{codeB}\rangle$  `\repeat` loops  $\langle\textit{codeA}\rangle\langle\textit{codeB}\rangle$  until `\ifsomething` is false. Then  $\langle\textit{codeB}\rangle$  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.

`if-macros.opm`

```
71 \_long\_def \_loop #1\_repeat{\_def\_body{#1}\_iterate}
72 \_long\_def \_loop #1\_repeat{\_def\_body{#1}\_iterate}
73 \_let \_repeat=\_fi % this makes \loop... \if... \repeat skippable
74 \_let \_repeat=\_fi
75 \_def \_iterate {\_body \_ea \_iterate \_fi}
```

`\xloop`  $\langle\textit{parameters}\rangle\backslash\textit{do}$   $\langle\textit{codeA}\rangle$  `\ifsomething`  $\langle\textit{codeB}\rangle$  `\repeat` or `\_xloop`  $\langle\textit{parameters}\rangle\backslash\textit{do}$   $\langle\textit{codeA}\rangle$  `\_ifsomething`  $\langle\textit{codeB}\rangle$  `\_repeat` behave analogically like `\loop`, but moreover: you can specify a rule of parameters scanning (like in `\def` primitive) and you can use these parameters (i.e. #1, #2, etc.) in the  $\langle\textit{codeA}\rangle$  or  $\langle\textit{codeB}\rangle$ . Each iteration of this loop reads next parameters from the input queue (just after `\repeat`) using the  $\langle\textit{parameters}\rangle$  rule. Example:

```
\def\countparams{\tmpnum=0
\loop ##1,\do \ifx\end##1\empty \else \incr\tmpnum \repeat}
\def\list{A, B, C}
\ea \countparams \list,\end, % \tmpnum = the number of comma separated parameters
```

Second new feature: the `\xloop` macro is expandable, if  $\langle\textit{codeA}\rangle$  and  $\langle\textit{codeB}\rangle$  include only expandable macros. This is not the case of previous example but you can prefix `\tmpnum=0` and `\incr` by the `\immediateassignment` LuaTeX primitive and you get expandable macro `\countparams`.

`if-macros.opm`

```
98 \_long\_def\xloop#1\do#2\_repeat{\_immediateassignment\_long\_def\_body#1{#2\_ea\_body\_fi}\_body}
99 \_long\_def\_xloop#1\_do#2\_repeat{\_immediateassignment\_long\_def\_body#1{#2\_ea\_body\_fi}\_body}
```

`\foreach`  $\langle\textit{list}\rangle\backslash\textit{do}$   $\langle\textit{what}\rangle$  repeats  $\langle\textit{what}\rangle$  for each element of the  $\langle\textit{list}\rangle$ . The  $\langle\textit{what}\rangle$  can include #1 which is substituted by each element of the  $\langle\textit{list}\rangle$ . The macro is expandable.

`\foreach`  $\langle\textit{list}\rangle\backslash\textit{do}$   $\langle\textit{parameter-mask}\rangle\langle\textit{what}\rangle$  reads parameters from  $\langle\textit{list}\rangle$  repeatedly and does  $\langle\textit{what}\rangle$  for each such reading. The parameters are declared by  $\langle\textit{parameter-mask}\rangle$ . 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`  $\langle\textit{list}\rangle\backslash\textit{do}$   $\langle\textit{what}\rangle$  is equivalent to `\foreach`  $\langle\textit{list}\rangle\backslash\textit{do}$  #1 $\langle\textit{what}\rangle$ .

Recommendation: it is better to use private variants of `\_foreach`. When the user writes `\load[tikz]` then `\foreach` macro is redefined in each TikZ environment. The private variants use

`\do` separator instead `\do` separator and it isn't redefined.

`\foreachx` `\list` behaves like `\ea` `\foreach` `\list` but the `\list` is expanded later, when its data are actually needed. This is more effective because the macro needn't skip a potentially huge amount of data in order to scan `\do{<what>}`.

if-macros.opm

```

130 \_newcount\_frnum      % the numeric variable used in \fornum
131 \_def\_do{\_doundefined} % we need to ask \_ifx#1\_do ...
132
133 \_def\_foreach{\_foreachx\_empty} \_def\_foreach{\_foreachAx\_empty}
134 \_long\_def\_foreachx #1\_do #2#{\_foreachAz{#1}{#2}}
135 \_long\_def\_foreachAx #1\_do #2#{\_foreachAz{#1}{#2}} \_let\_foreachx=\_foreachAx
136 \_long\_def\_foreachAz #1#2{\_isempty{#2}\_iftrue
137   \_afterfi{\_foreachA{#1}{#1}}\_else\_afterfi{\_foreachA{#1}{#2}}\_fi}
138 \_long\_def\_foreachA #1#2#3{\_putforstack
139   \_immediateassignment \_long\_gdef\_fbody#2{\_testparam#1..\_iftrue #3\_ea\_fbody\_fi}%
140   \_ea\_fbody #1#2\_finbody\_getforstack
141 }
142 \_long\_def\_testparam#1#2#3\_iftrue{\_ifx##1\_empty\_ea\_finbody\_else}
143 \_long\_def\_finbody#1\_finbody{}

```

`\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

```

159 \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
160 \_long\_def\_fornumstep#1:#2..#3\_do#4{\_putforstack
161   \_immediateassigned{%
162     \_gdef\_fbody##1{#4}%
163     \_global\_frnum=\_numexpr#2\_relax
164   }%
165   \_ea\_fornumB\_ea{\_the\_numexpr#3\_ea}\_ea{\_the\_numexpr#1}%
166 }
167 \_def\_fornumB #1#2{\_ifnum#1\_ifnum#2>0<\_else>\_fi \_frnum \_getforstack
168   \_else \_afterfi{\_ea\_fbody\_ea{\_the\_frnum}%
169     \_immediateassignment\_global\_advance\_frnum by#2
170     \_fornumB{#1}{#2}}\_fi
171 }
172 \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
173 \_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

```

191 \_newcount\_forlevel
192 \_def\_putforstack{\_immediateassigned{%
193   \_ifnum\_forlevel>0
194     \_sxdef\_frnum:\_the\_forlevel\_ea{\_the\_frnum}%
195     \_global\_slet{\_fbody:\_the\_forlevel}{\_fbody}%
196   \_fi
197   \_incr\_forlevel
198 }}
199 \_def\_getforstack{\_immediateassigned{%
200   \_decr\_forlevel
201   \_ifnum\_forlevel>0
202     \_global\_slet{\_fbody}{\_fbody:\_the\_forlevel}%

```

```

203 \global\frnum=\cs{frnum:\the\forlevel}\space
204 \fi
205 }}
206 \ifx\immediateassignment\undefined % for compatibility with older LuaTeX
207 \let\immediateassigned=\useit \let\immediateassignment=\empty
208 \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.

```

223 \def\foreachdef#1#2#{\toks0{#2}%
224 \long\edef#1##1{\ea\noexpand\csname _body:\csstring#1\endcsname
225 ##1\the\toks0 \noexpand\finbody}%
226 \foreachdefA#1{#2}}
227 \long\def\foreachdefA#1#2#3{%
228 \long\sdef{body:\csstring#1}#2{\testparam##1..\iftrue #3\cs{body:\csstring#1\ea}\fi}}
229
230 \public \foreachdef ;

```

if-macros.opm

### 2.6.3 Is-macros and selection of cases

There are a collection of macros `\isempty`, `\istokseempty`, `\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.

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

```

261 \long\def \isempty #1#2{\if\relax\detokenize{#1}\relax \else \ea\unless \fi#2}
262 \def \istokseempty #1#2{\ea\isempty\ea{\the#1}#2}
263 \public \isempty \istokseempty ;

```

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.

```

272 \long\def\isequal#1#2#3{\directlua{%
273 if "\luaescapestring{\detokenize{#1}}"=="\luaescapestring{\detokenize{#2}}"
274 then else tex.print("\nbb unless") end}#3}
275 \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.

```

282 \long\def\ismacro#1{\ea\isequal\ea{#1}}
283 \public \ismacro ;

```

if-macros.opm

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

```

290 \def\isdefined #1#2{\ifcsname #1\endcsname \else \ea\unless \fi #2}
291 \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 expandable.

if-macros.opm

```
299 \_long\_def\_isinlist#1#2#3{%
300   \_immediateassignment\_long\_def\_isinlistA##1#2##2\_end/_%
301   {\_if\_relax\_detokenize{##2}\_relax \_ea\_unless\_fi#3}%
302   \_ea\_isinlistA#1\_endlistsep#2\_end/_%
303 }
304 \_public \isinlist ;
```

`\isfile` `{⟨filename⟩}` `\iftrue` is true if the file `⟨filename⟩` exists and are readable by T<sub>E</sub>X.

if-macros.opm

```
311 \_newread \_testin
312 \_def\_isfile #1#2{%
313   \_openin\_testin =#{1}\_relax
314   \_ifeof\_testin \_ea\_unless
315   \_else \_closein\_testin
316   \_fi #2%
317 }
318 \_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

```
326 \_newifi \_ifexistfam
327 \_def\_isfont#1#2{%
328   \_begingroup
329     \_suppressfontnotfounderror=1
330     \_font\_testfont=#{1}\_relax
331     \_ifx\_testfont\_nullfont \_def\_tmp{\_existfamfalse \_unless}
332     \_else \_def\_tmp{\_existfamtrue}\_fi
333   \_ea \_endgroup \_tmp #2%
334 }
335 \_public \isfont ;
```

The macro `\isnextchar` `⟨char⟩{⟨codeA⟩}{⟨codeB⟩}` has a different syntax than all other is-macros. It executes `⟨codeA⟩` if a next character is equal to `⟨char⟩`. Else the `⟨codeB⟩` is executed. The macro is expandable. It checks the next character `⟨space⟩` too. If you want to check a next non-space character, use `\nospaceafter{\isnextchar` `⟨char⟩{⟨codeA⟩}{⟨codeB⟩}`.

if-macros.opm

```
347 \_long\_def\_isnextchar#1#2#3{\_immediateassignment
348   \_def\_isnextcharA{\_isnextcharB{#1}{#2}{#3}}%
349   \_immediateassignment\_futurelet \_next \_isnextcharA
350 }
351 \_long\_def\_isnextcharB#1{\_ifx\_next#1\_ea\_ignoresecond\_else\_ea\_usessecond\_fi}
352
353 \_public \isnextchar ;
```

`\casesof` `⟨token⟩` `⟨list of cases⟩` implements something similar to the `switch` command known from C language. It is expandable macro. The `⟨list of cases⟩` is a list of arbitrary number of pairs in the format `⟨token⟩{⟨what to do⟩}` which must be finalized by the pair `\_finc {⟨what to do else⟩}`. The optional spaces after `⟨token⟩`s and between listed cases are ignored. The usage of `\casesof` looks like:

```
\casesof ⟨token⟩
  ⟨token-1⟩ {⟨what to do if token=token-1⟩}
  ⟨token-2⟩ {⟨what to do if token=token-2⟩}
  ...
  ⟨token-n⟩ {⟨what to do if token=token-n⟩}
  \_finc    {⟨what to do in other cases⟩}
```

The meaning of tokens are compared by `\ifx` primitive. The parts `⟨what to do⟩` can be finalized by a macro which can read more data from the input stream as its parameters.

if-macros.opm

```
375 \_long\_def \_casesof #1#2#3{\_ifx #2\_finc \_ea\_ignoresecond \_else \_ea\_usessecond \_fi
376   {#3}{\_ifx#1#2\_ea\_ignoresecond \_else \_ea\_usessecond \_fi {\_finc{#3}}{\_casesof#1}}%
377 }
378 \_long\_def \_finc #1#2\_finc#3{#1}
379
380 \_public \casesof ;
```



`\qcasesof`  $\langle string \rangle$   $\langle list\ of\ cases \rangle$  behaves like `\casesof` but it compares phrases with the given  $\langle string \rangle$  using `\isequal`. The  $\langle list\ of\ cases \rangle$  includes pairs  $\langle phrases \rangle$   $\langle what\ to\ do\ if\ string=phrase \rangle$  finalized by a pair `\finc`  $\langle what\ to\ do\ else \rangle$ . The  $\langle phrases \rangle$  is a single phrase or phrases separated by `|` which means “or”. For example the pair `\ab|cde|f`  $\langle code \rangle$  runs  $\langle code \rangle$  if the given  $\langle string \rangle$  is `ab` or `cde` or `f`. The usage of `\qcasesof` can be found in [OpTeX trick 0132](#).

`if-macros.opm`

```
394 \long\def \qcasesof #1#2#3{\ifx\finc#2\ea\ignoresecond \else \ea\usesecond \fi
395   {#3}{\qcasesofA{#1}#2|\qcasesofA|\finc{#3}}{\qcasesof{#1}}}%
396 }
397 \long\def \qcasesofA#1#2|{\ifx\qcasesofA#2\ea\usesecond \else
398   \isequal{#1}{#2}\iftrue \qcasesofB \fi \afterfi{\qcasesofA{#1}}\fi
399 }
400 \long\def \qcasesofB #1\qcasesofA|#2#3{\fi\fi#2}
401
402 \public \qcasesof ;
```

`\xcasesof`  $\langle list\ of\ pairs \rangle$  extends the features of the macro `\casesof`. Each pair from the  $\langle list\ of\ pairs \rangle$  is in the format  $\langle if\ statement \rangle$   $\langle what\ to\ do \rangle$ , only the last pair must have the different format: `\finc`  $\langle what\ to\ do\ else \rangle$ . The  $\langle if\ statement \rangle$  can be arbitrary primitive `\if*` condition (optionally prefixed by `\unless`) and it must be closed in its expansion. It means that `\ifnum\mycount>0` is bad, `\ifnum\mycount>0 }` is correct. Optional spaces between parameters are ignored. Example:

```
\message {The \tmpnum has \xcasesof
          {\ifnum\tmpnum>0 } {positive}
          {\ifnum\tmpnum=0 } {zero}
          \finc                {negative} value}
```

The `\xcasesof` macro works with principle: first true condition wins, next conditions are not evaluated.

`if-macros.opm`

```
422 \def \xcasesof {\nospacefuturelet\next\xcasesofA}
423 \def \xcasesofA {\ifx\next\finc \ea\usesecond \else \ea \xcasesofB \fi}
424 \long\def \xcasesofB #1#2{%
425   #1\ea\ignoresecond\else \ea\usesecond\fi {\finc{#2}}{\xcasesof}%
426 }
427 \public \xcasesof ;
```

## 2.7 Setting parameters

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

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

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

The registers are represented by their names (control sequences). If the user re-defines this control sequence then the appropriate register exists steadily and built-in algorithms are using it without change. But user cannot access its value in this case. OpTeX declares two control sequences for each register: prefixed (private) and unprefixed (public). OpTeX 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 OpTeX macros still work without change.

`parameters.opm`

```
3 \codedec1 \normalbaselineskip {Parameter settings <2026-04-20>} % 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 \doublehyphemerits=10000
16 \finalhyphemerits=5000
```

```

17 \_adjdemerits=10000
18 \_uchyph=1
19 \_defaultshyphenchar=-
20 \_defaultskewchar=-1
21 \_hfuzz=0.1pt
22 \_vfuzz=0.1pt
23 \_overfullrule=5pt
24 \_linepenalty=10 % penalty between lines inside the paragraph
25 \_hyphenpenalty=50 % when a word is broken
26 \_exhyphenpenalty=50 % when the hyphenmark is used explicitly
27 \_binoppenalty=700 % between binary operators in math
28 \_relpenalty=500 % between relations in math
29 \_brokenpenalty=100 % after lines if they end by a broken word.
30 \_displaywidowpenalty=50 % before last line of paragraph if display math follows
31 \_predisplaypenalty=10000 % above display math
32 \_postdisplaypenalty=0 % below display math
33 \_delimiterfactor=901 % parameter for scaling delimiters
34 \_delimitershortfall=5pt
35 \_nulldelimiterspace=1.2pt
36 %\_scriptspace=0.5pt % \Umathspaceafterscript used in \_setmathdimens, \_setunimathdimens instead
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

```

The following four registers were introduced to fix some bugs in the LuaTeX engine (see also [chapter 5 of luabidi documentation](#)). When `\matheqdirmode` is positive short skip detection around display equations will work with right to left typesetting. When `\breakafterdirmode` is set to 1 a glue after a dir node will not be ignored. When `\mathemptydisplaymode` is set to 1 paragraphs before display equation containing only local `_par` and `dir` nodes will be discarded (other wise dir nodes are considered actual material in this case). When `\fixupboxesmode` is set to 1 dir nodes will be balanced at the end of hboxes (like in normal groups). This is important for unboxing:

```
\setbox0\hbox{\textdirection=1 text}\unhbox0 more text
```

With `\fixupboxesmode` set to 0 you have

```
\beginDirTRT text more text\endDirTRT
```

where the `\endDirTRT` is added from LuaTeX's par builder. While with `\fixupboxesmode` set to 1 you have

```
\beginDirTRT text\endDirTRT more text
```

parameters.opm

```

87 \_ifx\_mathemptydisplaymode\_undefined \_else
88 \_matheqdirmode=1
89 \_breakafterdirmode=1
90 \_mathemptydisplaymode=1
91 \_fixupboxesmode=1
92 \_fi

```

The `\tracingfonts` register is equivalent to `\pdftracingfonts` from pdfTeX. We set it to 1 in order to enable the full font names in tracing information.

parameters.opm

```
99 \_tracingfonts=1
```

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

Allocate registers that are used just like in plain T<sub>E</sub>X.

`\smallskipamount`, `\medskipamount`, `\bigskipamount`, `\normalbaselineskip`, `\normallineskip`, `\normallineskiplimit`, `\jot`, `\interdisplaylinepenalty`, `\interfootnotelinepenalty`.

parameters.opm

```
109 % We also define special registers that function like parameters:
110 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
111 \newskip\medskipamount \medskipamount=6pt plus 2pt minus 2pt
112 \newskip\bigskipamount \bigskipamount=12pt plus 4pt minus 4pt
113 \newskip\normalbaselineskip \normalbaselineskip=12pt
114 \newskip\normallineskip \normallineskip=1pt
115 \newdimen\normallineskiplimit \normallineskiplimit=0pt
116 \newdimen\jot \jot=3pt
117 \newcount\interdisplaylinepenalty \interdisplaylinepenalty=100
118 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100
119
120 \public \smallskipamount \medskipamount \bigskipamount
121 \normalbaselineskip \normallineskip \normallineskiplimit
122 \jot \interdisplaylinepenalty \interfootnotelinepenalty ;
```

Plain T<sub>E</sub>X macros for setting parameters. `\normalbaselines`, `\frenchspacing`, `\nonfrenchspacing`.

parameters.opm

```
129 \def\normalbaselines{\lineskip=\normallineskip
130 \baselineskip=\normalbaselineskip \lineskiplimit=\normallineskiplimit}
131
132 \def\frenchspacing{\sfcode\.=1000 \sfcode\?=1000 \sfcode\!=1000
133 \sfcode\:=1000 \sfcode\:=1000 \sfcode\:=1000 }
134 \def\nonfrenchspacing{\sfcode\.=3000 \sfcode\?=3000 \sfcode\!=3000
135 \sfcode\:=2000 \sfcode\:=1500 \sfcode\:=1250 }
136
137 \public \normalbaselines \frenchspacing \nonfrenchspacing ;
```

## 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 `\typothesize` 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.

parameters.opm

```
151 \normalbaselines % baseline setting, 10 pt font size
```

The following primitive registers have different values than in plain T<sub>E</sub>X. 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, not letter.

parameters.opm

```
160 \emergencystretch=20pt % we want to use third pass of paragraph building algorithm
161 % we don't need compatibility with old documents
162
163 \clubpenalty=10000 % after first line of paragraph
164 \widowpenalty=10000 % before last line of paragraph
165
166 \showboxbreadth=150 % for tracing boxes
167 \showboxdepth=7
168 \errorcontextlines=15
169 \tracinglostchars=2 % missing character warnings on terminal too
170
171 \outputmode=1 % PDF output
172 \pdfvorigin=0pt % origin is exactly at upper left corner
173 \pdfhorigin=0pt
174 \hoffset=25mm % margins are 2.5cm, no 1in
175 \voffset=25mm
176 \hsize=160mm % 210mm (from A4 size) - 2*25mm (default margins)
177 \vsize=244mm % 297mm (from A4 size) - 2*25mm (default margins) -3mm baseline correction
178 \pdfpagewidth=210 true mm
179 \pdfpageheight=297 true mm
```

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

```

186 \_def\_plaintexsetting{%
187   \_emergencystretch=0pt
188   \_clubpenalty=150
189   \_widowpenalty=150
190   \_pdfvorigin=1in
191   \_pdfhorigin=1in
192   \_hoffset=0pt
193   \_voffset=0pt
194   \_hsize=6.5in
195   \_vsize=8.9in
196   \_pdfpagewidth=8.5 true in
197   \_pdfpageheight=11 true in
198   \_nonfrenchspacing
199 }
200 \_public \plaintexsetting ;

```

## 2.7.4 OpTeX parameters

The main principle of how to configure OpTeX is not to use only parameters. A designer can copy macros from OpTeX and re-define them as required. This is a reason why we don't implement dozens of parameters, but we keep OpTeX 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.

```

226 \_newtoks\_picdir
227 \_public \_picdir ;

```

You can control the dimensions of included images by the parameters `\_picwidth` (which is equivalent to `\_picw`) and `\_picheight` (`\_pich`). 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.

More parameters accepted by `\_pdfximage` primitive can be set in the `\_picparams` tokens list. For example `\_picparams={page3}` selects page 3 from included PDF file.

```

248 \_newdimen\_picwidth \_picwidth=0pt \_let\_picw=\_picwidth
249 \_newdimen\_picheight \_picheight=0pt \_let\_pich=\_picheight
250 \_newtoks\_picparams
251 \_public \_picwidth \_picheight \_picparams ;

```

`\_kvdict` is dictionary name when `\_readkv`, `\_kvx`, `\_kv`, and `\_iskv` are processed. The default is empty.

```

258 \_newtoks \_kvdict
259 \_public \_kvdict ;

```

The `\_everytt` is the token list used in `\_begtt...\_endtt` environment and in the verbatim group opened by `\_verbinp` 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.

```
282 \newtoks\everytt
283 \newtoks\everyintt
284 \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.

```
298 \newcount\ttline \ttline=-1 % last line number in \begtt... \endtt
299 \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 T<sub>E</sub>X default 20pt. A well-typeset document should have the same dimension for all indentations, so you should say `\ttindent=\parindent` and `\iindent=\parindent`.

```
319 \newdimen\ttindent \ttindent=\parindent % indentation in verbatim
320 \newdimen\ttshift
321 \newdimen\iindent \iindent=\parindent
322 \public \ttindent \ttshift \iindent ;
```

parameters.opm

The tabulator `^^I` has its category code like space: it behaves as a space in normal text. This is a common plain T<sub>E</sub>X 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.

```
334 \newcount\tabspaces \tabspaces=3
335 \public \tabspaces ;
```

parameters.opm

`\hicolors` can include a list of `\hicolor` commands with re-declarations of default colors mentioned in the `\hicolors<name>` from `hisyntax-<name>.opm` file. The user can give his/her preferences about colors for syntax highlighting by this tokens list.

```
345 \newtoks\hicolors
346 \public \hicolors ;
```

parameters.opm

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 `\olistskipamount` is vertical skip above and below the items list if `\ilevel=1`.

The `\ilistskipamount` is vertical skip above and below the items list if `\ilevel>1`.

The `\itemskipamount` is vertical skip between list items, but not above the first and below the last.

```
367 \newtoks\defaultitem \defaultitem={\_\bullet$\_\enspace}
368 \newtoks\everyitem
369 \newtoks\everylist
370 \newcount \ilevel
371 \newskip\olistskipamount \olistskipamount=\medskipamount
372 \newskip\ilistskipamount \ilistskipamount=0pt plus.5\smallskipamount
373 \newskip\itemskipamount \itemskipamount=0pt
```

parameters.opm

```

374
375 \_public \defaultitem \everyitem \everylist \ilevel
376 \olistskipamount \ilistskipamount \itemskipamount ;
377 \_let \listskipamount = \olistskipamount % for backward compatibility

```

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

parameters.opm

```

383 \_newskip\_titskip \_titskip=40pt \_relax % \vglue above title printed by \tit
384 \_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

```

393 \_newdimen\_colsep \_colsep=20pt % space between columns
394 \_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

```

402 \_newtoks \_everytocline
403 \_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 2.32.6.

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

parameters.opm

```

417 \_newtoks\_bibtexhook
418 \_newtoks\_biboptions
419 \_newtoks\_bibpart
420 \_public \bibtexhook \biboptions \bibpart ;

```

`\everycaptionf` is used before printing caption in figures and `\everycaptiont` is used before printing caption in tables.

parameters.opm

```

427 \_newtoks\_everycaptiont \_newtoks\_everycaptionf
428 \_public \everycaptiont \everycaptionf ;

```

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

parameters.opm

```

435 \_newtoks\_everyii
436 \_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

```

447 \_newtoks\_everymnote
448 \_newdimen\_mnotesize \_mnotesize=20mm % the width of the mnote paragraph
449 \_newdimen\_mnoteindent \_mnoteindent=10pt % distance between mnote and text
450 \_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.

`\tabskip1` 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 pxtosize` is used.

parameters.opm

```

484 \newtoks\everytable \newtoks\thistable
485 \newtoks\tabiteml \newtoks\tabitemr \newtoks\tabstrut
486 \newdimen\tablinespace \newdimen\vvkern \newdimen\hhkern \newdimen\tsize
487 \newskip\tabskip1 \newskip\tabskipr
488 \everytable={} % code used after settings in \vbox before table processing
489 \thistable={} % code used when \vbox starts, is removed after using it
490 \tabstrut={\strut}
491 \tabiteml={\enspace} % left material in each column
492 \tabitemr={\enspace} % right material in each column
493 \tablinespace=2pt % additional vertical space before/after horizontal rules
494 \vvkern=1pt % space between double vertical line and used in \frame
495 \hhkern=1pt % space between double horizontal line and used in \frame
496 \tabskip1=0pt\relax % \tabskip used before first column
497 \tabskipr=0pt\relax % \tabskip used after the last column
498 \public \everytable \thistable \tabiteml \tabitemr \tabstrut \tablinespace
499 \vvkern \hhkern \tsize \tabskip1 \tabskipr ;

```

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

parameters.opm

```

508 \newtoks \eqlines \eqlines={\openup\jot}
509 \newtoks \eqstyle \eqstyle={\strut\displaystyle}
510 \newdimen \eqspace \eqspace=20pt
511 \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

```

518 \newtoks \lmfil \lmfil={\hfil}
519 \public \lmfil ;

```

The output routine uses token lists `\headline` and `\footline` in the same sense as plain  $\TeX$  does. 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

```

537 \newtoks\headline \headline={}
538 \newtoks\footline \footline={\hss\rmfixed \numprint\folio \hss}
539 \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

```

553 \newdimen \headlinedist \headlinedist=14pt
554 \newdimen \footlinedist \footlinedist=24pt
555 \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

```

563 \newskip \pgbottomskip \pgbottomskip=0pt \relax
564 \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.

```
578 \newtoks \nextpages
579 \public \nextpages ;
```

parameters.opm

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

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

```
591 \newtoks \pgbackground \pgbackground={} % for page background
592 \public \pgbackground ;
```

parameters.opm

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.

```
600 \newtoks \ovalparams
601 \newtoks \circleparams
602 %\ovalparams={\_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
603 % \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt }
604 %\circleparams={\_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
605 % \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt}
606
607 \newdimen \roundness \roundness=5mm % used in \clippingoval macro
608 \public \ovalparams \circleparams \roundness ;
```

parameters.opm

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

```
628 \let\cnvinfo=\ignoreit
```

parameters.opm

In addition to the LuaTeX primitive registers `\textdirection`, `\bodydirection` etc. we introduce the register `\inboxdirection` with the same meaning of its four values 0, 1, 2, 3. It affects the texts in internal boxes of the macros `\mathbox`, `\puttext`, `\transformbox`, `\rotbox`, `\inoval`, `\incircle`. Dirmode 0 is L→R (Latin), dirmode 1 (R→L, Hebrew) was tested, see `op-letter-he.tex`. Dirmode 2 is T→B, L→R, and Dirmode 3 is T→B, R→L.

```
640 \newcount\inboxdirection \inboxdirection=0
641 \public \inboxdirection ;
```

parameters.opm

## 2.8 More OpTeX macros

The second bundle of OpTeX macros is here.

```
3 \codedec1 \eoldef {OpTeX useful macros <2026-03-23>} % 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 `\setcatcode` *<catcode table>* pushes current catcode table to the stack and activates catcodes from the *<catcode table>*. The `\restorecatcode` 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, `\restorecatcode` pops the stacks and returns to the catcodes used before `\opinput` is run.

```

29 \def\opinput #1{\setcatcode\optexcatcodes
30   \savecatcodetable\tmpcatcodes \catcodetable\tmpcatcodes
31   \input {#1}\relax\restorecatcode}
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 `\setcatcode` 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 `\restorecatcode` reads the last saved catcode table number from the `\ctablelist` and uses it.

```

54 \catcodetable0
55
56 \def\setcatcode#1{\edef\ctablelist{{\the\catcodetable}\ctablelist}%
57   \catcodetable#1\relax
58 }
59 \def\restorecatcode{\ea\restorecatcodeA\ctablelist\relax}
60 \def\restorecatcodeA#1#2\relax{%
61   \ifx^#2\opwarning
62     {You can't use \noindent\restorecatcode without previous \string\setcatcode}%
63   \else \def\ctablelist{#2}\catcodetable#1\relax \fi
64 }
65 \def\ctablelist{.}
66
67 \public \setcatcode \restorecatcode ;

```

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 `\optecatcodes` table and sets it to the main catcode table 0.

```

77 \def\normalcatcodes {\catcodetable\optecatcodes \savecatcodetable0 \catcodetable0 }
78 \public \normalcatcodes ;

```

more-macros.opm

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]{\savemathsb \loadA #1,,,\end \restoremathsb}
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     \sxddef{\load:#1}{}%
98     \trycs{\afterload}{}\let\afterload=\undefined
99   \fi
100 }
101 \def\loadE #1\end{}
102 \public \load ;

```

more-macros.opm

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 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` *<token>* or `\nospaceafter{<text>}` ignores the following optional space at expand processor level using the negative `\romannumeral` trick. Then returns to process *<token>* or *<text>*.

The `\nospacefuturelet` behaves like `\futurelet` primitive, but it ignores the following optional space and works at expand processor level.

more-macros.opm

```

123 \newtoks\opt
124 \def\optdef#1[#2]{%
125   \def#1{\_isnextchar[{\_cs\_oA:\_csstring#1}{\_cs\_oA:\_csstring#1}[#2]}}%
126   \sdef\_oA:\_csstring#1{##1}{%
127     \immediateassignment\_opt={##1}\_cs\_oB:\_csstring#1\nospaceafter}%
128   \sdef\_oB:\_csstring#1\nospaceafter}%
129 }
130 \def\nospaceafter#1{\_expanded{\_unexpanded{#1}\_ea}\_romannumeral-`.\_noexpand}
131 \def\nospacefuturelet#1#2{\_ea\_immediateassignment
132   \_ea\_futurelet\_ea#1\_ea#2\_romannumeral-`.\_noexpand}
133
134 \_public \opt \optdef \nospaceafter \nospacefuturelet ;

```

`\_noprefix` *<cs>* works like `\csstring` *<cs>*, but ignores not only the first backslash but the second “\_” ignores too (if it follows the backslash).

more-macros.opm

```

142 \def\_noprefix#1{\_ea\_noprefixA \_csstring#1\_empty\_fin}
143 \def\_noprefixA #1#2\_fin{\_if \_#1\_else #1\_fi #2}

```

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.

more-macros.opm

```

161 \def\_eoldef #1{\_def #1{\_begingroup \_catcode\`^M=12 \_eoldefA #1}%
162   \_ea\_def\_csname \_eol:\_noprefix #1\_endcsname}
163 \_catcode\`^M=12 %
164 \_def\_eoldefA #1#2^M{\_endgroup\_csname \_eol:\_noprefix #1\_endcsname{#2}}%
165 \_normalcatcodes %
166
167 \_eoldef\_skiptoel#1{}
168
169 \_def\_bracedparam#1{%
170   \_trycs{\_eol:\_noprefix#1}%
171   {\_errmessage{\_string\bracedparam: \_string#1 isn't defined by \_string\eoldef}}%
172 }
173 \_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 `\scantoel` keeps category 7 for ^ in order to be able to use ^^J as comment character which means that the next line continues.

more-macros.opm

```

191 \def\_scantoel#1{\_begingroup \_setscancatcodes \_scantoelA#1}
192 \_def\_setscancatcodes{\_setverb \_catcode\`^M=12\_catcode\`^=7\_catcode\`^=10\_catcode\`^J=14 }
193 \_catcode\`^M=12 %
194 \_def\_scantoelA#1#2^M{\_endgroup #1{#2}}%
195 \_normalcatcodes %
196
197 \_public \scantoel ;

```

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 `\_repltoks` and `\_replacestringsB` is run with `\_replacestringsC` in a loop. It finishes processing (if final `!` is scanned) or appends the next part to `\_repltoks` separated by `<textB>` and continues loop. The final part of the `\replstring` macro removes the last `?` from resulting `\_repltoks` and defines a new version of the `\foo`.

more-macros.opm

```
217 \_newtoks\_repltoks
218 \_catcode`!=3 \_catcode`?=3
219 \_long\_def\_replstring #1#2#3{% \replstring #1{stringA}{stringB}
220 \_long\_def\_replacestringsA##1#2{\_repltoks\_ea{##1}\_replacestringsB\_empty}%
221 \_long\_def\_replacestringsB##1#2{\_ea\_replacestringsC\_ea{##1}}%
222 \_long\_def\_replacestringsC##1{\_ifx!##1\_empty\_else
223 \_toksapp\_repltoks{#3##1}\_ea\_replacestringsB\_ea\_empty\_fi}%
224 \_ea\_replacestringsA \_ea\_empty#1?#2!#2%
225 \_ea\_replacestringsD \_ea\_empty\_the\_repltoks#1}
226 \_long\_def\_replacestringsD #1?#2{\_repltoks\_ea{#1}\_edef#2{\_the\_repltoks}}%
227 \_normalcatcodes
228
229 \_public \replstring ;
```

You can find three special tricks in the `\replstring` code:

- Adding and removing `?` character: without this the `\def\b{xxxA}\replstring\b{AA}{BB}` will cause an error.
- Using `\_toksapp` instead `\addto`: without this you cannot have `#6` in the string. Moreover, it significantly reduces the calculation time. Unfortunately, it does `\replstring` unexpandable.
- Using `\_empty` before scanning separated parameters and removing it after scanning by `\_ea`: without this `TEX` removes braces from `<textA>{xxx}<textA>`.

The `\replstring` macro isn't expandable. But you can see [OpT<sub>E</sub>X trick 0136](#) and [OpT<sub>E</sub>X trick 0137](#). These tricks implement expandable alternatives and enable more general modifications of macros by regular expressions.

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`.

more-macros.opm

```
260 \_def\_catcode#1{\_catcode \_if'\_noexpand#1\_ea\_else\_if"\_noexpand#1\_else
261 \_if'\_noexpand#1\_else \_ea\_ea\_ea\_ea\_ea\_ea\_ea#1\_fi\_fi\_fi}
```

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

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

more-macros.opm

```
270 \_def\_removespaces #1 {\_isempty{#1}\_iffalse #1\_ea\_removespaces\_fi}
271 \_ea\_def \_ea\ignorept \_ea#\_ea1\_detokenize{pt}{#1}
272
273 \_public \removespaces \ignorept ;
```

If you do `\let\foo=a` then it is not simple to return from `\foo` to the original character code of `a`. You can write ``a` but you cannot write ``\foo`. The macro `\cstochar<sequence>` solves this problem. If the sequence is equal to a character then it expands to this character (always with `catcode 12`). If it isn't equal to a character then it expands to nothing. You can say `\expanded{\cstochar\foo}` if you want to extract the character code.

more-macros.opm

```
285 \_def\_cstochar#1{\_ea\_cstocharA\_meaning#1 {} {} \_fin}
286 \_def\_cstocharA#1 #2 #3 #4\_fin{\_isinlist{#1#2}-\_iffalse #3\_fi}
287
288 \_public \cstochar ;
```



You can use expandable `\bp{⟨dimen⟩}` converter 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. The `⟨expression⟩` can include `+-*/^()` and decimal numbers in common syntax. Moreover, `a//b` means integer division and `a%b` is remainder. The math functions (and pi constant) have to be prefixed by `math.`, because it is processed by Lua interpreter. For example `\expr{math.pi*math.sqrt(2)}`. The list of available functions is in [Lua manual](#).

You can set the number of decimal digits after decimal point of the results of `\bp` and `\expr` by optional syntax `\bp[⟨digits⟩]{⟨dimen⟩}` and `\expr[⟨digits⟩]{⟨expression⟩}`. Default is `\decdigits`.

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

more-macros.opm

```
318 \def\_decdigits{3} % digits after decimal point in \_bp and \_expr outputs.
319 \def\_pttopb#1{%
320   \directlua{tex.print(string.format('\_pcent.#1f',
321     token.scan_dimen()/65781.76))}% pt to bp conversion
322 }
323 \def\_bp{\_isnextchar[{\_bpA}{\_bpA[\_decdigits]}}
324 \def\_bpA[#1]#2{\_pttopb{#1}\_dimexpr#2\_relax}
325 \def\_expr{\_isnextchar[{\_exprA}{\_exprA[\_decdigits]}}
326 \def\_exprA[#1]#2{\directlua{tex.print(string.format('\_pcent.#1f',#2))}}
327
328 \_public \expr \bp ;
```

The `\expr` and `\bp` macros return their results with given number of decimal digits even if there are trailing zeros. There is the `\nnum` macro to “normalize” such decimal numbers. `\nnum{⟨number⟩}` expands its parameter and removes trailing zeros after decimal point and removes the decimal point if nothing follows. For example, use `\nnum{\expr[10]{⟨expression⟩}}`. The `\nnum` macro is fully expandable.

more-macros.opm

```
339 \def\_nnum #1{\_ea\_nnumA\_expanded{#1}.\_fin}
340 \def\_nnumA #1.#2\_fin{#1\_ifx~#2~\_else \_nnumB #20.\_fin \_fi}
341 \def\_nnumB #10.#2\_fin{#1\_ifx~#2~\_nnumC#1\_else \_nnumB #1.0.\_fin \_fi}
342 \def\_nnumC #1.#2\_ifx~#1~\_else .#1\_fi}
343 \_public \nnum ;
```

You can write `\setpos[⟨label⟩]` somewhere and the position of such `\setpos[⟨label⟩]` can be referenced by `\posx[⟨label⟩]`, `\posy[⟨label⟩]` and `\pospg[⟨label⟩]`. The first two macros expand to  $x$  and  $y$  position measured from left-bottom corner of the page (dimen values) and `\pospg[⟨label⟩]` expands to the `⟨pageno⟩`, i.e. to the page number counted from one at beginning of the document. These values are available in the second (and more) T<sub>E</sub>X run, because the information is saved to `.ref` file and restored from it at the beginning of the T<sub>E</sub>X job. If these values are not known then mentioned macros expand to 0sp, 0sp and 0. The following example implements `\linefrom[⟨label⟩]` and `\lineto[⟨label⟩]` macros. The line connecting these two points is drawn (after second T<sub>E</sub>X run):

```
\def\linefrom[#1]{\setpos[#1:f]\drawlinefromto[#1]}
\def\lineto [#1]{\setpos[#1:t]}
\def\drawlinefromto[#1]{\ifnum\pospg[#1:f]>0 \ifnum\pospg[#1:f]=\pospg[#1:t]
  \pdfliteral{q 0 0 m 1 0 0 RG % << red color
    \expr{\bp{\posx[#1:t]}-\bp{\posx[#1:f]}}
    \expr{\bp{\posy[#1:t]}-\bp{\posy[#1:f]}} 1 S Q}\fi\fi
}
This is a text.\linefrom[A]\par
This is second paragraph with a text.\lineto[A]
Try to reverse from-to and watch the changes.
```

The coordinates are saved to the `.ref` file in the format `\_Xpos{⟨label⟩}{⟨x-pos⟩}{⟨y-pos⟩}`. The `\_Xpos` macro defines `\_pos:⟨label⟩` as `{⟨x-pos⟩}{⟨y-pos⟩}{⟨total-pg⟩}{⟨rel-pg⟩}`. We need to read only given parameter by `\_posi`, `\_posii` or `\_posiii` auxiliary macros. The implementation of `\setpos`, `\posx` and `\posy` macros are based on `\padsavepos` `\pdflastxpos` and `\pdflastypos` pdfT<sub>E</sub>X primitives. The `\pospg` simply reads the data from the `\_currpage` macro.



```

380 \_def\_Xpos#1#2#3{\_sxdef{\_pos:#1}{#2}{#3}\_currpage}}
381 \_def\_setpos[#1]{\_openref\_pdfsavepos
382 \_ewref\_Xpos{#1}\_unexpanded{\\_the\_pdflastxpos}{\_the\_pdflastypos}}}}
383
384 \_def\_posx [#1]{\_ea \_posi \_expanded {\_trycs{\_pos:#1}{0}{0}{0}{0}}sp}}
385 \_def\_posy [#1]{\_ea \_posii \_expanded {\_trycs{\_pos:#1}{0}{0}{0}{0}}sp}}
386 \_def\_pospg[#1]{\_ea \_posiii \_expanded {\_trycs{\_pos:#1}{0}{0}{0}{0}}}
387
388 \_def\_posi #1#2#3#4{#1} \_def\_posii #1#2#3#4{#2} \_def\_posiii #1#2#3#4{#3}
389
390 \_public \setpos \posx \posy \pospg ;

```

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.

```

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

```

`\docgen` processes lines before `\_codedecl` because the version text in the macro `\_<pkg>_version` can be defined here. The package documentation can print it. `\docgen` prints banner to log because TeX doesn't do it when command line doesn't begin with the main file name after parameters.

```

414 \_def\_docgen #1 {\_ea \_docgenA \_input{#1.opm}}
415 \_long \_def\_docgenA #1\_codedecl#2\_endcode #3\_doc {#1\_wlog{\_banner}\_skiptoeol}
416
417 \_public \docgen ;

```

## 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

```

or simply `\myframe {text3}`. 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
  \readkv\myframedefaults \readkv{\the\opt}%
  \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 read default parameters by `\readkv\myframedefaults` and secondly the actual parameters are read by `\readkv{\the\opt}`. The last setting wins. Third: the values can be used by the expandable `\kv{<key>}` macro. The `\kv{<key>}`

returns ??? if such a  $\langle key \rangle$  isn't declared but if `.notdef` key is declared then its value is returned in this case.

The `\kv{key}` macro fully expands the value of key. If you don't want to expand the value, use `\trykv{key}{code}`. It returns unexpanded value of  $\langle key \rangle$  if declared else returns expanded  $\langle code \rangle$ . For example `\edef\macro{\trykv{key}{}}` defines `\macro` as the unexpanded value of the key.

You can use keys without values in the parameters list too. Then you can ask if the key is declared by `\iskv{key}\iftrue` or the key is undeclared by `\iskv{key}\iffalse`. For example, you write to your documentation of your code that user can set the `draft` option without the value. Then you can do

```
\optdef\myframe [] #1{...
  \readkv\myframedefaults \readkv{\the\opt}%
  \iskv{draft}\iftrue ...draft mode... \else ...final mode... \fi
...}
```

Maybe, you want to allow not only `draft` option but `final` option (which is opposite to `draft`) too and you want to apply the result from the last given option. Then `\iskv` doesn't work because you can only check if both options are declared but you don't know what one is given as last. But you can use `\kvx{key}{code}` to declare  $\langle code \rangle$  which is processed immediately when the  $\langle key \rangle$  is processed by `\readkv`. For example

```
\newcount\mydraftmode
\kvx{draft}{\mydraftmode=1 }
\kvx{final}{\mydraftmode=0 }
\optdef\myframe [] #1{...
  \readkv\myframedefaults \readkv{\the\opt}%
  \ifnum\mydraftmode=1 ...draft mode... \else ...final mode... \fi
...}
```

The syntax of `\kvx{key}{code}` allows to use `#1` inside the code. It is replaced by the actual  $\langle value \rangle$ . Example: `\kvx{opt}{\message{opt is #1}}`, then `\readkv{opt=HELLO}` prints "opt is HELLO".

The `\nokvx{code}` can declare a  $\langle code \rangle$  processed for all  $\langle keys \rangle$  undeclared by `\kvx`. The `#1` and `#2` can be used in the  $\langle code \rangle$ , `#1` is  $\langle key \rangle$ , `#2` is  $\langle value \rangle$ . If `\nokvx` is unused then nothing is done for undeclared  $\langle key \rangle$ . Example: `\nokvx{\opwarning{Unknown option "#1"}}`.

The default dictionary name (where key-value pairs are processed) is empty. You can use your specific dictionary by `\kvdict={name}`. Then `\redakv`, `\kv`, `\trykv`, `\iskv`, `\kvx` and `\nokvx` macros use this named dictionary of  $\langle key \rangle/\langle value \rangle$  pairs. Package options can be processed when `\kvdict={pkg:pkg}`, example is the `\mathset` macro in [math.opm package](#).

Recommendation: If the value of the key-value pair includes `=` or `,` or `]`, then use the syntax  $\langle key \rangle=\langle value \rangle$ .

A more extensive example can be found in [OpTeX trick 0073](#).

keyval.opm

```
3 \_codedecl \readkv {Key-value dictionaries <2023-11-24>} % preloaded in format
```

## Implementation.

The `\readkv{list}` expands its parameter and does replace-strings in order to remove spaces around equal signs and commas. Then `\_kvscan` reads the parameters list finished by `,\fin` and saves values to `\_kv:dict:key` macros. The `\_kvx:dict:key` is processed (if it is defined) with parameter  $\langle value \rangle$  after it.

The `\kvx{key}{code}` defines the `\_kvx:dict:key#1` macro and `\nokvx{code}` defines the `\_nokvx:dict:key` macro.

The `\trykv{key}{code}` returns unexpanded value of  $\langle key \rangle$  if declared, else it runs  $\langle code \rangle$ .

The `\kv{key}` returns expanded value of  $\langle key \rangle$  if declared, else returns expanded value of `.notdef` key else expands `\_kvunknown`.

The `\iskv{key}\iftrue` (or `\iffalse`) is the test, if the  $\langle key \rangle$  is defined in current  $\langle dict \rangle$ .

keyval.opm

```
23 \_def\readkv#1{\_ea\_def\_ea\_tmpb\_ea{#1,}%
24   \_replstring\_tmpb{= }{=}\_replstring\_tmpb{,}{,}\_replstring\_tmpb{ },{ },}%
25   \_ea \_nospaceafter \_ea\_kvscan\_tmpb\_fin}
26 \_def\_kvscan#1,#2{\_ifx^#1\_else \_kvscd #1==\_fin \_fi
27   \_ifx\_fin#2\_empty \_ea\_ignoreit \_else\_ea\_useit \_fi {\_kvscan#2}}
28 \_def\_kvscd#1=#2=#3\_fin{\_sdef{\_kvcs#1}{#2}%
29   \_trycs{\_kvx:\_the\_kvdict:#1}%
```

```

30      {\_trycs{\nokvx:\_the\_kvdict}{\_ea\_ignoreit}{#1}\_ea\_ignoreit}{#2}}
31 \_def\_kvx#1#2{\_sdef{\_kvx:\_the\_kvdict:#1}##1{#2}}
32 \_def\_nokvx#1{\_sdef{\_nokvx:\_the\_kvdict}##1\_ea\_ignoreit##2{#1}}
33 \_def\_trykv#1{\_ea\_trykvA \_begincsname\_kvcs#1\_endcsname \_ignoreit}
34 \_def\_trykvA#1{\_ifx #1\_ignoreit \_ea\_useit\_else \_unexpanded\_ea{#1}\_fi}
35 \_def\_kv#1{\_expanded{\_trykv{#1}{\_trykv{.\_csstring\notdef}{\_kvunknown}}}}
36 \_def\_iskv#1#2{#2\_else\_ea\_unless\_fi \_ifcsname\_kvcs#1\_endcsname}
37 \_def\_kvcs{\_kv:\_the\_kvdict:}
38 \_def\_kvunknown{???}
39
40 \public \readkv \kvx \nokvx \kv \trykv \iskv ;

```

## 2.10 Plain TeX macros

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

```

3 \_codedecl \magstep {Macros from plain TeX <2022-10-11>} % preloaded in format

```

plain-macros.opm

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

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

`\active` is character constant 13, we can use it in the context `\catcode`<character>=\active`.

plain-macros.opm

```

15 \_def\_dospecials {\do\ \do\\\do{\do\}\do\$\do\&%
16 \do\#\do\^{\do\^K\do\^A\do\%\do\~}
17 \_chardef\_active = 13
18
19 \_public \dospecials \active ;

```

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

plain-macros.opm

```

29 \_def \_magstephalf{1095 }
30 \_def \_magstep#1{\_ifcase#1 1000\_or 1200\_or 1440\_or 1728\_or 2074\_or 2488\_fi\_space}
31 \_public \magstephalf \magstep ;

```

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

plain-macros.opm

```

39 \_def\^M{\ } % control <return> = control <space>
40 \_def\^I{\ } % same for <tab>
41
42 \_def\lq{' } \_def\rq{' }
43 \_def\lbrack[{} \_def\rbrack[{} % They are only public versions.
44 % \_catcode\^L=\active \_outer\def^^L{\par} % ascii form-feed is "\outer\par" % obsolete
45
46 \_let\_endgraf=\_par \_let\_endline=\_cr
47 \_public \endgraf \endline ;

```

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

plain-macros.opm

```

53 % In \obeylines, we say '\let^M=\_par' instead of '\def^M{\_par}'
54 % since this allows, for example, '\let\_par=\cr \obeylines \halign{...'
55 {\_catcode\^M=13 % these lines must end with %
56 \_gdef\_obeylines{\_catcode\^M=13\_let^M\_par}%
57 \_glet^M=\_par} % this is in case ^M appears in a \write
58 \_def\_obeyspaces{\_catcode\^ =13 }
59 {\_obeyspaces\_glet =\_space}
60 \_public \obeylines \obeyspaces ;

```

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

```

70 \protected\def\thinspace {\kern .16667em }
71 \protected\def\negthinspace {\kern-.16667em }
72 \protected\def\enspace {\kern.5em }
73 \protected\def\enskip {\hskip.5em\relax}
74 \protected\def\quad {\hskip1em\relax}
75 \protected\def\qqquad {\hskip2em\relax}
76 \protected\def\smallskip {\vskip\smallskipamount}
77 \protected\def\medskip {\vskip\medskipamount}
78 \protected\def\bigskip {\vskip\bigskipamount}
79 \def\nointerlineskip {\prevdepth=-1000pt }
80 \def\offinterlineskip {\baselineskip=-1000pt \lineskip=0pt \lineskiplimit=\maxdimen}
81
82 \public \thinspace \negthinspace \enspace \enskip \quad \qqquad \smallskip
83 \medskip \bigskip \nointerlineskip \offinterlineskip ;
84
85 \def\topglue {\nointerlineskip\vglue-\topskip\vglue} % for top of page
86 \def\vglue {\afterassignment\vglA \skip0=}
87 \def\vglA {\par \dimen0=\prevdepth \hrule height0pt
88 \nobreak\vskip\skip0 \prevdepth=\dimen0 }
89 \def\hg1ue {\afterassignment\hg1A \skip0=}
90 \def\hg1A {\leavevmode \count255=\spacefactor \vrule width0pt
91 \nobreak\hskip\skip0 \spacefactor=\count255 }
92 \protected\def~{\penalty10000 \ } % tie
93 \protected\def\slash {/\penalty\exhyphenpenalty} % a '/' that acts like a '-'
94
95 \public \topglue \vglue \hg1ue \slash ;

```

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

```

104 \protected\def \break {\penalty-10000 }
105 \protected\def \nobreak {\penalty10000 }
106 \protected\def \allowbreak {\penalty0 }
107 \protected\def \filbreak {\par\vfil\penalty-200\vfilneg}
108 \protected\def \goodbreak {\par\penalty-500 }
109 \protected\def \eject {\par\break}
110 \protected\def \supereject {\par\penalty-20000 }
111 \protected\def \dosupereject {\ifnum \insertpenalties>0 % something is being held over
112 \line{\kern-\topskip \nobreak \vfill \supereject \fi}
113 \def \removelastskip {\ifdim\lastskip=\zo _else \vskip-\lastskip \fi}
114 \def \smallbreak {\par\ifdim\lastskip<\smallskipamount
115 \removelastskip \penalty-50 \smallskip \fi}
116 \def \medbreak {\par\ifdim\lastskip<\medskipamount
117 \removelastskip \penalty-100 \medskip \fi}
118 \def \bigbreak {\par\ifdim\lastskip<\bigskipamount
119 \removelastskip \penalty-200 \bigskip \fi}
120
121 \public \break \nobreak \allowbreak \filbreak \goodbreak \eject \supereject \dosupereject
122 \removelastskip \smallbreak \medbreak \bigbreak ;

```

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

```

130 \def \line {\hbox to\hsize}
131 \def \leftline #1{\line{#1\hss}}
132 \def \rightline #1{\line{\hss#1}}
133 \def \centerline #1{\line{\hss#1\hss}}
134 \def \rlap #1{\hbox to\zo{#1\hss}}
135 \def \llap #1{\hbox to\zo{\hss#1}}
136 \def\_\underbar #1{\setbox0=\dirhbox{#1}\dp0=\zo \math \underline{\box0}$}
137
138 \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. The `\strut` macro puts the `\strutbox`.

```

146 \newbox\strutbox
147 \setbox\strutbox=\hbox{\vrule height8.5pt depth3.5pt width0pt}
148 \def \strut {\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
149
150 \public \strutbox \strut ;

```

Alignment. `\hidewidth \ialign \multispan`.

plain-macros.opm

```
156 \def \hidewidth {\hskip\hideskip} % for alignment entries that can stick out
157 \def \ialign{\everycr={}\_tabskip=\_zoskip \halign} % initialized \halign
158 \newcount\_mscount
159 \def \multispan #1{\omit \mscount=#1\_relax
160   \loop \ifnum\_mscount>1 \spanA \repeat}
161 \def \spanA {\span\omit \advance\_mscount by-1 }
162
163 \public \hidewidth \ialign \multispan ;
```

Tabbing macros are omitted because they are obsolete.

Indentation and similar macros are defined here: `\hang`, `\textindent`, `\item`, `\itemitem`, `\narrower`, `\raggedright`, `\ttraggedright`, `\leavevmode`.

plain-macros.opm

```
172 \def \hang {\hangindent\_parindent}
173 \def \textindent #1{\indent\_llap{#1\_enspace}\_ignorespaces}
174 \def \item {\_par\_hang\_textindent}
175 \def \itemitem {\_par\_indent \hangindent2\_parindent \textindent}
176 \def \narrower {\advance\_leftskip\_parindent
177   \advance\_rightskip\_parindent}
178 \def \raggedright {\_rightskip=0pt plus2em
179   \spaceskip=.3333em \xspaceskip=.5em\_relax}
180 \def \ttraggedright {\_tt\_rightskip=0pt plus2em\_relax} % for use with \tt only
181 \def \leavevmode {\_unhbox\_voidbox} % begins a paragraph, if necessary
182
183 \public \hang \textindent \item \itemitem \narrower \raggedright \ttraggedright \leavevmode ;
```

Few character codes are set for backward compatibility. But old obscurities (from plain T<sub>E</sub>X) 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

```
194 %\chardef\%=\%
195 \let\% =\_pcent % more natural, can be used in lua codes.
196 \chardef\&=\&
197 \chardef\#=\#
198 \chardef\$=\$
199 \chardef\ss="FF
200 \chardef\ae="E6
201 \chardef\oe="F7
202 \chardef\o="F8
203 \chardef\AE="C6
204 \chardef\OE="D7
205 \chardef\O="D8
206 \chardef\i="19 \chardef\j="1A % dotless letters
207 \chardef\aa="E5
208 \chardef\AA="C5
209 \chardef\S="9F
210 \def\l{\_errmessage{\_usedirectly l}}
211 \def\L{\_errmessage{\_usedirectly L}}
212 %\def\_{{\_ifmmode \kern.06em \vbox{\hrule width.3em}\_else\_fi} % obsolete
213 \_protected\_def\_{{\_relax \_ifmmode \_hbox{\_}\_else\_fi}
214 \_def\dag{\_errmessage{\_usedirectly †}}
215 \_def\ddag{\_errmessage{\_usedirectly ‡}}
216 \_def\copyright{\_errmessage{\_usedirectly ©}}
217 %\_def\Orb{\_mathhexbox20D} % obsolete (part of Copyright)
218 %\_def\P{\_mathhexbox27B} % obsolete
219
220 \def \usedirectly #1{Load Unicoded font by \string\fontfam\space and use directly #1}
221 \def \mathhexbox #1#2#3{\_leavevmode \_hbox{\_math \_mathchar"#1#2#3$}}
222 \public \mathhexbox ;
```

The `\_unichars` macro is run when Unicode font family is loaded, Unicodes are used instead old plain T<sub>E</sub>X settings.

plain-macros.opm

```
229 \def\_unichars{% Plain TeX character sequences with different codes in Unicode:
230   \_chardef\ss="ß
231   \_chardef\ae="æ \_chardef\AE="Æ
```

```

232 \_chardef\oe=`œ \_chardef\OE=`Ë
233 \_chardef\o=`ø \_chardef\O=`Ø
234 \_chardef\aa=`å \_chardef\AA=`Å
235 \_chardef\l=`ł \_chardef\L=`Ł
236 \_chardef\i=`ı \_chardef\j=`ĵ
237 \_chardef\S=`$ \_chardef\P=`¶
238 \_chardef\dag`†
239 \_chardef\ddag`‡
240 \_chardef\copyright`©
241 }

```

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

plain-macros.opm

```

249 \_def \_oalign #1{\_leavevmode\_vtop{\_baselineskip=\_zo \_lineskip=.25ex
250 \_ialign{##\_crr#1\_crr}}}
251 \_def \_oalignA {\_lineskiplimit=\_zo \_oalign}
252 \_def \_oalign {\_lineskiplimit=-\_maxdimen \_oalign} % chars over each other
253 \_def \_shiftx #1{\_dimen0=#1\_kern\_ea\_ignorept \_the\_fontdimen1\_font
254 \_dimen0 } % kern by #1 times the current slant
255 \_def \_d #1{\_oalignA{\_relax#1\_crr\_hidewidth\_shiftx{-1ex}.\_hidewidth}}
256 \_def \_b #1{\_oalignA{\_relax#1\_crr\_hidewidth\_shiftx{-3ex}%
257 \_vbox to.2ex{\_hbox{\_char\_macron}\_vss}\_hidewidth}}
258 \_def \_c #1{\_setbox0=\_hbox{#1}\_ifdim\_ht0=1ex\_accent\_cedilla #1%
259 \_else\_oalign{\_unhbox0\_crr\_hidewidth\_cedilla\_hidewidth}\_fi}
260 \_def \_dots{\_relax\_ifmode\_ldots\_else$\_math\_ldots\_thinsk$\_fi}
261 \_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.

The `\uniaccents` macro redeclares codes for accents and it is run when Unicode font family is loaded.

plain-macros.opm

```

274 \_def \_oldaccents {%
275 \_def\`##1{\\_accent\_tgrave ##1}}%
276 \_def\'##1{\\_accent\_tacute ##1}}%
277 \_def\v##1{\\_accent\_caron ##1}}%
278 \_def\u##1{\\_accent\_tbreve ##1}}%
279 \_def\=##1{\\_accent\_macron ##1}}%
280 \_def\^##1{\\_accent\_circumflex ##1}}%
281 \_def\.`##1{\\_accent\_dotaccent ##1}}%
282 \_def\H##1{\\_accent\_hungarumlaut ##1}}%
283 \_def\~##1{\\_accent\_ttilde ##1}}%
284 \_def\"##1{\\_accent\_dieresis ##1}}%
285 \_def\r##1{\\_accent\_ring ##1}}%
286 }
287 \_public \_oldaccents ;
288
289 % ec-lmr encoding (will be changed after \fontfam macro):
290 \_chardef\_tgrave=0
291 \_chardef\_tacute=1
292 \_chardef\_circumflex=2
293 \_chardef\_ttilde=3
294 \_chardef\_dieresis=4
295 \_chardef\_hungarumlaut=5
296 \_chardef\_ring=6
297 \_chardef\_caron=7
298 \_chardef\_tbreve=8
299 \_chardef\_macron=9
300 \_chardef\_dotaccent=10
301 \_chardef\_cedilla=11
302
303 \_def \_uniaccents {% accents with Unicode
304 \_chardef\_tgrave="0060
305 \_chardef\_tacute="00B4
306 \_chardef\_circumflex="005E
307 \_chardef\_ttilde="02DC
308 \_chardef\_dieresis="00A8
309 \_chardef\_hungarumlaut="02DD
310 \_chardef\_ring="02DA

```



```

311 \_chardef\_caron="02C7
312 \_chardef\_tbreve="02D8
313 \_chardef\_macron="00AF
314 \_chardef\_dotaccent="02D9
315 \_chardef\_cedilla="00B8
316 \_chardef\_ogonek="02DB
317 \_let \_uniaccents=\_relax
318 }

```

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.

plain-macros.opm

```

329 \_def \_hrulefill {\_leaders\_hrule\_hfill}
330 \_def \_dotfill {\_cleaders\_hbox{\_math\_mkern1.5mu.\_mkern1.5mu}\_hfill}
331 \_def \_rightarrowfill {\_math\_smash-\_mkern-7mu%
332 \_cleaders\_hbox{\_mkern-2mu\_smash-\_mkern-2mu}\_hfill
333 \_mkern-7mu\_mathord\_rightarrow$}
334 \_def \_leftarrowfill {\_math\_mathord\_leftarrow\_mkern-7mu%
335 \_cleaders\_hbox{\_mkern-2mu\_smash-\_mkern-2mu}\_hfill
336 \_mkern-7mu\_smash-$}
337
338 \_mathchardef \_braceld="37A \_mathchardef \_bracerd="37B
339 \_mathchardef \_bracelu="37C \_mathchardef \_braceru="37D
340 \_def \_downbracefill {\_math\_setbox0=\_hbox{\_braceld$}%
341 \_braceld \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_braceru
342 \_bracelu \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_bracerd$}
343 \_def \_upbracefill {\_math\_setbox0=\_hbox{\_braceld$}%
344 \_bracelu \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_bracerd
345 \_braceld \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_braceru$}
346
347 \_public \hrulefill \dotfill
348 \rightarrowfill \leftarrowfill \downbracefill \upbracefill ;

```

The last part of plain T<sub>E</sub>X macros: `\magnification`, `\showhyphens`, `\bye`. Note that math macros are defined in the `math-macros.opm` file (section 2.15).

plain-macros.opm

```

356 \_def \_magnification {\_afterassignment \_magA \_count255 }
357 \_def \_magA {\_mag=\_count255 \_truedimen\_hsize \_truedimen\_vsize
358 \_dimen\_footins=8truein
359 }
360 % only for backward compatibility, but \margins macro is preferred.
361 \_public \magnification ;
362
363 \_def \_showhyphens #1{\_setbox0=\_vbox{\_parfillskip=0pt \_hsize=\_maxdimen \_tenrm
364 \_pretolerance=-1 \_tolerance=-1 \_hbadness=0 \_showboxdepth=0 \_#1}}
365
366 \_def \_bye {\_par \_vfill \_supereject \_byehook \_end}
367 \_public \showhyphens \bye ;

```

Plain T<sub>E</sub>X reads `hyphen.tex` with patterns when `\language=0`. We do the same.

plain-macros.opm

```

373 \_lefthyphenmin=2 \_righthyphenmin=3 % disallow x- or -xx breaks
374 \_input hyphen % en(USenglish) patterns from TeX82

```

## 2.11 Preloaded fonts for text mode

The format in LuaT<sub>E</sub>X can download only non-Unicode fonts. Latin Modern EC is loaded here. These fonts are totally unusable in LuaT<sub>E</sub>X 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 of 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.

If the `*.tfm` files are missing during format generation then the format is successfully generated without any pre-loaded fonts. It doesn't matter if each document processed by OpTeX declares Unicode fonts. You can create such fonts-less format anyway if you set `\fontspreload` to `\relax` before `\input optex.ini`, i.e.: `luatex -ini '\let\fontspreload=\relax \input optex.ini'`

fonts-preload.opm

```

3 \codedec{\tenrm {Latin Modern fonts (EC) preloaded <2022-02-12>}} % preloaded in format
4
5 \ifx\fontspreload\relax
6   \let\tenrm=\nullfont \let\tenbf=\nullfont \let\tenit=\nullfont
7   \let\tenbi=\nullfont \let\tentt=\nullfont
8 \else
9   % Only few text fonts are preloaded:
10  % allow missing fonts during format generation
11  \suppressfontnotfounderror=1
12  \font\tenrm=ec-lmr10 % roman text
13  \font\tenbf=ec-lmbx10 % boldface extended
14  \font\tenit=ec-lmri10 % text italic
15  \font\tenbi=ec-lmbxi10 % bold italic
16  \font\tentt=ec-lmtt10 % typewriter
17  \suppressfontnotfounderror=0
18 \fi
19
20 \tenrm
21
22 \public \tenrm \tenbf \tenit \tenbi \tentt ;

```

## 2.12 Using \font primitive directly

You can declare a new *font switch* by `\font` primitive:

```

\font \<font switch> = <font file name> <size spec>
% for example:
\font \tipa = tipa10 at12pt % the font tipa10 at 12pt is loaded
% usage:
{\tipa TEXT} % the TEXT is printed in the loaded font.

```

The `<size spec>` can be empty or at `<dimen>` or scaled `<scale factor>`. The `<font file name>` must be terminated by space or surrounded in the braces.

OpTeX starts with `\font` primitive which is able to read only `tfm` files. i.e. the `<font file name>.tfm` (and additional data for glyphs) must be correctly installed in your system. If you want to load OpenType `otf` or `ttf` font files, use the declarator `\initunifonts` before first `\font` primitive. This command adds additional features to the `\font` primitive which gives the extended syntax:

```

\font \<font switch> = {[<font file name>]:<font features>}<size spec>
% or
\font \<font switch> = {<font name>:<font features>}<size spec>

```

where `<font file name>` is name of the OpenType font file with the extension `.otf` or `.ttf` or without it. The braces in the syntax are optional, use them when the `<font file name>` or `<font name>` includes spaces. The original syntax for `tfm` files is also available. Example:

```

\initunifonts
\font\crimson=[Crimson-Roman] at11pt % the font Crimson-Regular.otf is loaded
\font\crimsonff=[Crimson-Roman]:+smcp;+onum at11pt % The same font is re-loaded
% with font features
{\crimson Text 12345} % normal text in Crimson-Regular
{\crimsonff Text 12345} % Crimson-Regular with small capitals and old digits

```

`\initunifonts` loads the implementation of the `\font` primitive from `luaotfload` package. More information is available in the `luaotfload-latex.pdf` file.

You can use `\ufont` macro which runs `\initunifonts` followed by `\font` primitive. And `\fontfam` does (among other things) `\initunifonts` too. You need not to specify `\initunifonts` if `\fontfam` or `\ufont` is used.

When `\initunifonts` is declared then the `\font` primitive is ready to read Type1 fonts too. If you have `file.afm` and `file.pfb` then you can declare `\font\font=font.afm` and use `\font`. It means that you needn't to create `tfm` files nor `vf` files, you can use Type1 fonts directly. They behave as Unicode fonts if the `afm` metrics are implemented correctly (with correct names of all included glyphs). But we must to say that Type1 font format is old technology, the loading of Type1 fonts is not optimized. Use OpenType fonts (`otf` or `ttf`) if it is possible.

Let's sum it up. Suppose that `\initunifonts` was used. The `\font` primitive is able to load OpenType fonts (`otf` or `ttf`), Type1 fonts (`afm` and `pfb`) or classical `tfm` fonts. We strongly recommend to prefer OpenType format over Type1 format over `tfm` format. The last one doesn't support Unicode. If there is nothing else left and you must to use `tfm`, then you must to implement re-encoding from Unicode to the `tfm` encoding at macro level, see the [OpTeX trick 0018](#) for example.

### 2.12.1 The `\setfontsize` macro

It seems that you must decide about final size of the font before it is loaded by the `\font` primitive. It is not exactly true; OpTeX offers powerful possibility to resize the font already loaded on demand.

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

- `at<dimen>`, for example `\setfontsize{at12pt}`. It gives the desired font size directly.
- `scaled<scale factor>`, for example `\setfontsize{scaled1200}`. The font is scaled in respect to its native size (which is typically 10 pt). It behaves like `\font\... scaled<number>`.
- `mag<decimal number>`, 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 initial 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.
```

The `\resizethefont` works with arbitrary current font, for example with the font loaded directly by `\font` primitive. For example:

```

\ufont\tencrimson=[Crimson-Roman]:+onum % font Crimson-Regular at 10 pt is loaded
\def\crimson{\tencrimson\resizethefont} % \crimson uses the font size on demand

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

This is not only an academical example. The `\csrimson` command defined here behaves like variant selector in the Font Selection System (section 2.13). It takes only information about size from the font context, but it is sufficient. You can use it in titles, footnotes, etc. The font size depending on surrounding size is automatically selected. There is a shortcut `\sfont` with the same syntax like `\font` primitive, it declares a macro which selects the font and does resizing depending on the current size. So, the example above can be realized by `\sfont\crimson=[Crimson-Roman]:+onum`.

### 2.12.2 The `\font`-like commands summary

- `\font` is  $\TeX$  primitive. When  $\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$  starts, then it accepts only classical  $\mathcal{T}\mathcal{E}\mathcal{X}$  syntax and doesn't allow to load Unicode fonts. Once `\initunifonts` (or `\fontfam`) is used, the `\font` primitive is re-initialized: now it accepts extended syntax and it is able to load Unicode OpenType fonts.
- `\ufont` is a shortcut of `\initunifonts \font`. I.e. it behaves like `\font` and accepts extended syntax immediately.
- `\sfont` has syntax like extended `\font`. It declares a macro which selects the given font and resizes it to the current size (given by `\setfontsize`). In various part of document (text, footnotes, titles), the size of this font is selected by the declared macro properly.

### 2.12.3 The `\fontlet` declarator

We have another command for scaling: `\fontlet` which can resize arbitrary font given by its font switch.

```
\fontlet \langle new font switch \rangle = \langle given font switch \rangle \langle size spec \rangle
example:
\fontlet \bigfont = \_tenbf at15pt
```

The `\langle given font switch \rangle` must be declared previously by `\font` or `\fontlet` or `\fontdef`. The `\langle new font switch \rangle` is declared as the same font at given `\langle size spec \rangle`. The equal sign in the syntax is optional. You can declare `\langle new font switch \rangle` as the scaled current font by

```
\fontlet \langle new font switch \rangle = \font \langle size spec \rangle
```

### 2.12.4 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”. Each design size is implemented in its individual font file and  $\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$  is able to choose right file if various optical sizes and corresponding file names are declared for the font by `\_regtfm` or `\_regoptsizes` command. The command `\setfontsize` sets the internal requirements for optical size if the parameter is in the format `at<dimen>` or `mag<factor>`. Then the command `\resizethefont` or `\fontlet` or variant selectors try to choose the font suitable for the required optical size. For example

```
\fontfam[lm]
The text is printed in font [lmroman10-regular] at 10 pt.
\setfontsize{at13pt}\rm
Now, the text is printed in [lmroman12-regular] at 13 pt.
```

See also section 2.13.12.

### 2.12.5 Font rendering

If `\initunifonts` isn't declared then  $\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$  uses classical font renderer (like in `pdf $\mathcal{T}\mathcal{E}\mathcal{X}$` ). The extended font renderer implemented in the `Luaotfload` package is started after `\initunifonts`.

The  $\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$  format uses `lua $\mathcal{H}\mathcal{T}\mathcal{E}\mathcal{X}$`  engine by default (from 2025). It means that the `harfbuzz` library is ready to use for font rendering as an alternative to built-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 `lua $\mathcal{H}\mathcal{T}\mathcal{E}\mathcal{X}$`  engine is not used then `mode=harf` is ignored. See `Luaotfload` documentation for more information.

### 2.12.6 Implementation of resizing

Only “resizing” macros and `\initunifonts` are implemented here. Other aspects of Font Selection System and their implementation are described in section 2.13.14.

```
3 \_codedecl \setfontsize {Font resizing macros <2022-11-08>} % preloaded in format
```

fonts-resize.opm

`\initunifonts` macro extends  $\text{Lua}\mathcal{T}\mathcal{E}\mathcal{X}$ 's font capabilities, in order to be able to load Unicode fonts. Unfortunately, this part of  $\text{Op}\mathcal{T}\mathcal{E}\mathcal{X}$  depends on the `luaotfload` package, which adapts `Con $\mathcal{T}\mathcal{E}\mathcal{X}$` '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. `\initunifonts` sets itself to relax because we don't want to do this work twice. `\ufont` is a shortcut of `\initunifonts \font`.

```
fonts-resize.opm
```

```

16 \protected\def \initunifonts {%
17   \directlua{%
18     require('luaotfload-main')
19     luaotfload.main()
20     optex.hook_into_luaotfload()
21   }%
22   \glet \fmodtt=\unifmodtt % use \ttunifont for \tt
23   \glet \initunifonts=\relax % we need not to do this work twice
24   \glet \initunifonts=\relax
25 }
26 \protected\def \ufont {\initunifonts \font}
27
28 \public \initunifonts \ufont ;

```

The `\setfontsize {<size spec>}` saves the `<size spec>` to the `\sizespec` macro. The `\optsize` value is calculated from the `<size spec>`. If the `<size spec>` is in the format `scaled<factor>` then `\optsize` is set from `\defaultoptsize`. If the `<size spec>` is in the `mag<number>` format then the contents of the `\sizespec` macro is re-calculated to the `at<dimen>` format using previous `\optsize` value.

```
fonts-resize.opm
```

```

41 \newdimen \optsize \optsize=10pt
42 \newdimen \defaultoptsize \defaultoptsize=10pt
43 \newdimen \lastmagsize
44
45 \def\setfontsize #1{%
46   \edef\sizespec{#1}%
47   \ea \setoptsize \sizespec\relax
48 }
49 \def\setoptsize {\isnextchar a{\setoptsizeA}
50   {\isnextchar m{\setoptsizeC}{\setoptsizeB}}}
51 \def\setoptsizeA at#1\relax{\optsize=#1\relax\lastmagsize=\optsize} % at<dimen>
52 \def\setoptsizeB scaled#1\relax{\optsize=\defaultoptsize\relax} % scaled<scalenum>
53 \def\setoptsizeC mag#1\relax{%
54   \ifdim\lastmagsize>\zo \optsize=\lastmagsize \else \optsize=\pdffontsize\font \fi
55   \optsize=#1\optsize
56   \lastmagsize=\optsize
57   \edef\sizespec{at\the\optsize}%
58 }
59 \public \setfontsize \defaultoptsize ;

```

The `\fontname` primitive returns the `<font file name>` optionally followed by `<size spec>`. The `\xfontname` macro expands to `<font file name>` without `<size spec>`. We need to remove the part `<space>at<dimen>` from `\fontname` output. The letters `at` have category 12 in the `\stringat` macro.

```
fonts-resize.opm
```

```

69 \edef\stringat{\string a\string t}
70 \edef\xfontname#1{\unexpanded{\ea\xfontnameA\fontname}#1 \stringat\relax}
71 \expanded{\def\noexpand\xfontnameA#1 \stringat#2\relax}{#1}

```

`\fontlet <font switch A> <font switch B> <size spec>` does

`\font <font switch A> = {\<font file name>} <size spec>`

Note, that the `\xfontname` output is converted due to optical size data using `\optfn`.

```
fonts-resize.opm
```

```

81 \protected\def \fontlet #1#2{\ifx #2=\ea\fontlet \ea#1\else
82   \ea\font \ea#1\expanded{{\optfn\xfontname#2}}\fi}
83 \public \xfontname \fontlet ;

```

`\newcurrfontsize <size spec>` does `\fontlet <saved switch>=\font <size spec>\relax <saved switch>`. It changes the current font at the given `<size spec>`.

`\resizethefont` is implemented by `\newcurrfontsize` using data from the `\sizespec` macro.

`\sfont` has the same syntax like `\font` primitive, but declares a macro which selects the font and sets its size properly dependent on the current size.

```

97 % \newcurrfontsize{at25pt}
98 \def \newcurrfontsize {\_ea\_newcurrfontsizeA \_csname \_ea\_csstring \_the\_font \_endcsname}
99 \def \newcurrfontsizeA #1#2{\_fontlet #1\_font #2\_relax \_fontloaded#1#1}
100 \protected\_def \resizethefont {\_newcurrfontsize\_sizespec}
101 \protected\_def \_sfont #1{%
102   \protected\_edef #1{\_csname \_sfont:\_csstring#1\_endcsname \resizethefont}%
103   \initunifonts \_ea\_font \_csname \_sfont:\_csstring#1\_endcsname
104 }
105 \public \newcurrfontsize \resizethefont \_sfont ;

```

The `\_regtfm`  $\langle font id \rangle$   $\langle optical size data \rangle$  registers optical sizes data directly by the font file names. This can be used for tfm files or OpenType files without various font features. See also `\_regoptsizes` in section 2.13.12. The `\_regtfm` command saves the  $\langle optical size data \rangle$  concerned to the  $\langle font id \rangle$ . The  $\langle optical size data \rangle$  is in the form as shown below in the code where `\_regtfm` is used.

The `\_optfn`  $\langle fontname \rangle$  expands to the  $\langle fontname \rangle$  or to the corrected  $\langle fontname \rangle$  read from the  $\langle optical size data \rangle$  registered by `\_regtfm`. It is used in the `\fontlet` macro.

The implementation detail: The `\_reg:`  $\langle font id \rangle$  is defined as the  $\langle optical size data \rangle$  and all control sequences `\_reg:`  $\langle fontname \rangle$  from this data line have the same meaning because of the `\_reversetfm` macro. The `\_optfn` expands this data line and apply `\_runoptfn`. This macro selects the right result from the data line by testing with the current `\_optsize` value.

```

128 \def\_regtfm #1 0 #2 *{\_ea\_def \_csname \_reg:#1\_endcsname{#2 16380 \_relax}%
129   \def\_tmpa{#1}\_reversetfm #2 * %
130 }
131 \def\_reversetfm #1 #2 {% we need this data for \_setmathfamily
132   \_ea\_let\_csname \_reg:#1\_ea\_endcsname
133   \_csname \_reg:\_tmpa\_endcsname
134   \_if*#2\_else \_ea\_reversetfm \_fi
135 }
136 \def\_optfn #1{%
137   \_ifcsname \_reg:#1\_endcsname
138     \_ea\_ea\_ea \_runoptfn
139     \_csname \_reg:#1\_ea\_endcsname
140   \_else
141     #1%
142   \_fi
143 }
144 \def\_runoptfn #1 #2 {%
145   \_ifdim\_optsize<#2pt #1\_ea\_ignoretfm\_else \_ea\_runoptfn
146   \_fi
147 }
148 \def\_ignoretfm #1\_relax{}

```

Optical sizes data for preloaded 8bit Latin Modern fonts:

```

154 \_regtfm lmr 0 ec-lmr5 5.5 ec-lmr6 6.5 ec-lmr7 7.5 ec-lmr8 8.5 ec-lmr9 9.5
155   ec-lmr10 11.1 ec-lmr12 15 ec-lmr17 *
156 \_regtfm lmbx 0 ec-lmbx5 5.5 ec-lmbx6 6.5 ec-lmbx7 7.5 ec-lmbx8 8.5 ec-lmbx9 9.5
157   ec-lmbx10 11.1 ec-lmbx12 *
158 \_regtfm lmri 0 ec-lmri7 7.5 ec-lmri8 8.5 ec-lmri9 9.5 ec-lmri10 11.1 ec-lmri12 *
159 \_regtfm lmtt 0 ec-lmtt8 8.5 ec-lmtt9 9.5 ec-lmtt10 11.1 ec-lmtt12 *

```

## 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 switches*, 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 “built-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 name 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.

When `\fontfam [⟨Font Family⟩]` is used and the given family isn’t found in the current TeX system and the *⟨Font Family⟩* is previously declared by `\fontfamsub [⟨Font Family⟩] [⟨Other Family⟩]` then OpTeX does the given substitution and runs `\fontfam [⟨Other Family⟩]`.

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 {<size spec>}` command in the same sense as other font modifiers. It saves information about font size to the font context. See section 2.12.1. 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 {[\langle font file \rangle]}` or `\loadmath {\langle font name \rangle}` 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) or by `\fontlet` command (see section 2.12.3) or by `\fontdef` command (see sections 2.13.5). 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 “built-in” or declared by `\moddef` command. They do modifications in the font context but don't select any font.
  - “built-in” font modifiers are `\setfontsize` (see section 2.12.1), `\setff` (see section 2.13.9), `\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

You can declare `\langle font-switch \rangle` by the `\fontdef` command.

```
\fontdef\langle font-switch \rangle {\langle family selector \rangle \langle font modifiers \rangle \langle variant selector \rangle}
```

where `\langle family selector \rangle` and `\langle font modifiers \rangle` are optional and `\langle variant selector \rangle` is mandatory.

The resulting `\langle font-switch \rangle` declared by `\fontdef` is “fixed font switch” independent of the font context. 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 `\fontdef` does the following steps. It pushes the current font context to a stack, it does modifications of the font context by given `\langle family selector \rangle` and/or `\langle font modifiers \rangle` and it finds the real font by `\langle variant selector \rangle`. This font is not selected but it is assigned to the declared `\langle font switch \rangle` (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.

### 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\langle new variant selector \rangle {\langle family selector \rangle \langle font modifiers \rangle \langle variant selector \rangle}
```

where `\langle family selector \rangle` and `\langle font modifiers \rangle` are optional and `\langle variant selector \rangle` is mandatory. The `\langle new variant selector \rangle` declared by `\famvardef` 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 `\langle new variant selector \rangle` 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 the public namespace (`\mf`, `\caps`). They are most often declared in font family files and they are loaded by `\fontfam`. A conflict with such names in the public 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. When using this default `\tt` selector you need not to load the LatinModern family (by `\fontfam[lm]`) because the font file is explicitly specified in the macro `\_ttunifont`. The default `\tt` selector scales this font without respecting optical sizes.

The `\tt` variant selector is used in OpTeX internal macros `\_ttfont` (verbatim texts), `\_urlfont` (printing URL’s), `\_printii` (printing index entry when doc macros are activated) and `\ttraggedright` (plain TeX macro).

The `\tt` variant selector can be re-defined 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. It is recommended to use `\setff{-liga;-tlig}` to suppress the ligatures in typewriter fonts.

The `\famvardef\tt` behaves differently than `\famvardef\anythingelse`: it creates the `\tt` variant selector independent of current font family and it redefines both macros `\tt` and `\_tt` in order to all internal OpTeX macros based on this selector are working with the new selected font.

If you load `\fontfam[lm]` and declare `\famvardef\tt{\LMfonts\ttset\setff{-tlig}\rm}` then you get almost the same behavior as the default `\tt` selector does with one difference: a new `\tt` selector respects optical sizes declared in the LatinModern font family.

You can implement ex-height correction of `\tt` font with respect to your font family used for general text. For example:

```
\load[lm]
\famvardef\tt{\LMfonts\ttset\setfontsize{mag1.1}\setff{-tlig}\rm}
... or ...
\let\oritt=\tt
\famvardef\tt{\setfontsize{mag1.1}\oritt}
```

The second alternative uses the default font from `\_ttunifont` without respecting optical sizes.

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

```

The `\setfontsize` command is described in the section 2.12.1. The `\setff` command was described in previous subsection.

`\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=lowercase) 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   \addUmathfont \_xits {[XITSMath-Regular]}{} {[XITSMath-Bold]}{} {}
25   \addto\_frac{\_fam\_xits}\_addto\_cal{\_fam\_xits} \_public \frac \cal ;
26   % \bf, \bi from FiraMath:
27   \let\_bsansvariables=\_bfvariables
28   \let\_bsansGreek=\_bfGreek
29   \let\_bsansgreek=\_bfgreek

```

f-heros.opm

```

30 \let\bsansdigits=\bfdigits
31 \let\bisansvariables=\bivvariables
32 \let\bisansgreek=\bigreek
33 % \resetmathchars <fam-number> <list of \Umathchardef csnames> ;
34 \mathchars \xits {\bigtriangleup \bigblacktriangleup \blacktriangle
35 \vartriangle \smallblacktriangleright
36 \unicodevdots \unicodeadots \unicodeddots} % ... etc. you can add more
37 \_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 \<font-switch> {\_fontnamegen} \_sizespec
```

Note that the extended `\font` syntax `\font\<font-switch> {<font name>:<font features>} <size spec.>` or `\font\<font-switch> {[<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 11 pt 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  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  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\⟨Familyselector⟩ {%
  \_def\_currfamily {⟨Familyselector⟩}%
  \_def\_fontnamegen {...}% this is copied from 7-th parameter of \_famdecl
  \resetmod
  ⟨run all family-dependent font modifiers used before Familyselector without warnings⟩
```

### 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 in public 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  $\mathrm{OpT}_{\mathrm{E}}\mathrm{X}$  warns about it. See the end of section 2.13.4.

If you need to use an internal control sequence declared in your fontfile, use the reserved name space with names starting with two `_` followed by family identifier or by `vf` if it relates to variable fonts.

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

Please, look at [Op \$\mathrm{T}\_{\mathrm{E}}\mathrm{X}\$  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.

Several fonts don't switch to the font features if the features are specified directly as documented above. You must add the `script=latn` specification to the features string when using these fonts, see `f-baskerville.opm` for example. The reason: these fonts don't follow the OpenType specification and they don't set the DFLT script but only scripts with given names like `latn`. And the tables implementing all font features are included here. You can check the internals of the font by FontForge: View / Show ATT / OpenType Tables / GSUB. Do you see the DFLT script here?

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

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

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

In general:

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

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

If `\_optname` gets an argument which is not registered  $\langle internal-template \rangle$  then it expands to `\_failedoptname` which typically ends 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 Poltawskiego, 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-switches 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

```

### 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 can use the macros `\_faminfo`, `\_famalias` and `\_famtext`. See the example from `fams-ini.tex`:

```

3 % Version <2022-10-18>. 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 \_famsrc {CTAN}
11 \_famtext {Computer Modern like family:}
12
13 \_famfrom {GUST}
14 \_faminfo [Latin Modern] {TeX Gyre fonts based on Computer Modern} {f-lmfonts}
15 { -, \nbold, \sans, \sans\nbold, \slant, \ttset, \ttset\slant, \ttset\caps, %
16 \ttprop, \ttprop\bolder, \quotset: {\rm\bf\it\bi}
17 \caps: {\rm\it}
18 \ttlight, \ttcond, \dunhill: {\rm\it} \upital: {\rm} }
19 \_famalias [LMfonts] \_famalias [Latin Modern Fonts] \_famalias [lm]
20
21 \_famtext {TeX Gyre fonts based on Adobe 35:}
22
23 \_faminfo [Termes] {TeX Gyre Termes fonts based on Times} {f-termes}
24 { -, \caps: {\rm\bf\it\bi} }
25 \_famalias [Times]
26
27 \_faminfo [Heros] {TeX Gyre Heros fonts based on Helvetica} {f-heros}
28 { -, \caps, \cond, \caps\cond: {\rm\bf\it\bi} }
```

... 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 Implementation of the Font Selection System

```

3 \_codedecl \fontfam {Fonts selection system <2025-06-29>} % preloaded in format
```

The main principle of the Font Selection System is: run one or more modifiers followed by `\fontsel`. Modifiers save data and `\fontsel` selects the font considering saved data. Each basic variant selector `\rm`, `\bf`, `\it`, `\bi`, and `\tt` runs internal variant modifier `\_fmodrm`, `\_fmodbf`, `\_fmodit`, `\_fmodbi` and `\_fmodtt`. These modifiers save their data to the `\_famv` macro which is `rm` or `bf` or `it` or `bi` or `tt`. The `\currvar` selector is `\fontsel` by default, but variant selectors declared by `\famvardef` change it.

```

17 \_def\_famv{rm} % default value
18 \_protected\_def \_fmodrm {\_def\_famv{rm}}
19 \_protected\_def \_fmodbf {\_def\_famv{bf}}
20 \_protected\_def \_fmodit {\_def\_famv{it}}
21 \_protected\_def \_fmodbi {\_def\_famv{bi}}
22 \_protected\_def \_fmodtt {\_def\_famv{tt}}
23
24 \_protected\_def \_rm {\_fmodrm \fontsel \_marm}
25 \_protected\_def \_bf {\_fmodbf \fontsel \_mabf}
26 \_protected\_def \_it {\_fmodit \fontsel \_mait}
```

```

27 \protected\def \_bi {\_fmodbi \_fontsel \_mabi}
28 \protected\def \_tt {\_fmodtt \_fontsel \_matt}
29 \protected\def \_currvar {\_fontsel} \protected\def \currvar{\\_currvar}
30 \public \rm \bf \it \bi \tt ;

```

The `\fontsel` creates the  $\langle font switch \rangle$  in the format  $\_ten\langle famv \rangle$  and loads the font associated to the  $\langle font switch \rangle$ . The loading is done by:

- a) `\letfont \langle font switch \rangle = \savedswitch \_sizespec`
- b) `\font \langle font switch \rangle = \fontnamegen \_sizespec`

The a) variant is used when `\_fontnamegen` isn't defined, i.e. `\fontfam` wasn't used: only basic variant and `\_sizespec` is taken into account. The b) variant is processed when `\fontfam` was used: all data saved by all font modifiers are used during expansion of `\_fontnamegen`.

After the font is loaded, final job is done by `\_fontselA\langle font-switch \rangle`.

fonts-select.opm

```

47 \protected\def \_fontsel {%
48   \ifx\_fontnamegen\undefined % \fontfam was not used
49     \ea\_let \_ea\_tmpf \_csname _ten\_famv\_endcsname
50     \ea\_fontlet \_csname _ten\_xfamv\_endcsname \_tmpf \_sizespec
51   \else % \fontfam is used
52     \ea\_font \_csname _ten\_xfamv\_endcsname {\_fontnamegen}\_sizespec
53   \fi \relax
54   \ea \_fontselA \_csname _ten\_xfamv\_endcsname
55 }
56 \def\_fontselA #1{%
57   \protected\def \_currvar {\_fontsel}% default value of \_currvar
58   \logfont #1% font selecting should be logged.
59   \setwsp #1% wordspace setting
60   \fontloaded #1% initial settings if font is loaded firstly
61   #1% select the font
62 }
63 \def \_logfont #1{}
64 \def \_xfamv {\_famv}
65
66 \public \fontsel ;

```

If a font is loaded by macros `\fontsel` 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. The `mte.opm` package uses it. See also [OpTeX trick 0058](#).

fonts-select.opm

```

83 \def\_fontloaded #1{\_ifnum\_skewchar#1=-1 \_skewchar#1=-2 \_newfontloaded#1\_fi}
84 \def\_newfontloaded #1{}

```

`\_ttunifont` is default font for `\tt` variant when `\initunifonts` is declared. User can re-define it or use `\famvardef\tt`. The `\_unifmodtt` macro is used instead `\_fmodtt` after `\initunifonts`. It ignores the loading part of the following `\fontsel` and do loading itself.

fonts-select.opm

```

94 \def\_ttunifont{[lmmono10-regular]:\_fontfeatures-tlig;}
95 \def\_unifmodtt\_fontsel{% ignore following \_fontsel
96   \ea\_font \_csname _ten\_ttfamv\_endcsname {\_ttunifont}\_sizespec \_relax
97   \ea\_fontselA \_csname _ten\_ttfamv\_endcsname
98   \def \_currvar{\_tt}%
99 }
100 \def\_ttfamv{tt}

```

A large part of the Font Selection System was re-implemented in Feb. 2022. We want to keep backward compatibility:

fonts-select.opm

```

107 \def \_tryloadrm\_tenrm {\_fmodrm \_fontsel}
108 \def \_tryloadbf\_tenbf {\_fmodbf \_fontsel}
109 \def \_tryloadit\_tenit {\_fmodit \_fontsel}
110 \def \_tryloadbi\_tenbi {\_fmodbi \_fontsel}
111 \def \_tryloadtt\_tentt {\_fmodtt \_fontsel}
112 \def \_reloading {}

```

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 current meaning of the `\fontnamegen` macro. The `\_mathfaminfo` is saved for usage in the catalog.

fonts-select.opm

```

129 \def\famdecl [#1]#2#3#4#5#6#7#8{%
130   \_initunifonts \unichars \uniaccents
131   \_unless\_ifcsname \_f:\_csstring#2:main.fam\_endcsname
132     \_isfont{#7}\_iffalse
133     \_opwarning{Family [#1] skipped, font "#7" not found}\_endinput
134     \_ifcsname \_fams:\_famfile\_endcsname \_famsubstitute \_fi
135   \_else
136     \_edef\_currfamily {\_csstring #2}\_def\_mathfaminfo{#6}%
137     \_wterm {FONT: [#1] -- \_string#2 \_detokenize{(#3)^~J mods:{#4} vars:{#5} math:{#6}}}%
138     \_unless \_ifx #2\_undefined
139     \_opwarning{\_string#2 is redefined by \_string\_famdecl\_space[#1]}\_fi
140     \_protected\_edef#2{\_def\_noexpand\_currfamily{\_csstring #2}\_unexpanded{#8\_resetfam}}%
141     \_ea \_let \_csname \_f:\_currfamily:main.fam\_endcsname =#2%
142   \_fi
143   \_else \_csname \_f:\_csstring#2:main.fam\_endcsname \_rm \_endinput \_empty\_fi
144 }
145 \def\_initfontfamily{%
146   \_csname \_f:\_currfamily:main.fam\_endcsname \_rm
147 }

```

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

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

`\_fsetV` *<key>=<value>*, ..., *<key>=<value>* expands to `\def\_keyV{value}` in the loop.

`\_onlyif` *<key>=<value-a>*, *<value-b>*, ..., *<value-z>*: {*<what>*} expands *<what>* only if the `\_keyV` is defined as *<value-a>* or *<value-b>* or ... or *<value-z>*.

`\_prepcommalist` *ab,{}cd,fin*, 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`.

fonts-select.opm

```

173 \def\_fvars #1 #2 #3 #4 {%
174   \_sdef\_fvar:rm}{#1}%
175   \_sdef\_fvar:bf}{#2}%
176   \_ifx.#2\_slet\_fvar:bf}{\_fvar:rm}\_fi
177   \_sdef\_fvar:it}{#3}%
178   \_ifx.#3\_slet\_fvar:it}{\_fvar:rm}\_fi
179   \_sdef\_fvar:bi}{#4}%
180   \_ifx.#4\_slet\_fvar:bi}{\_fvar:it}\_fi
181 }
182 \def\_currV{\_trycs\_fvar:\_famv}{rm}}
183 \def\_V{ }
184 \def \_fsetV #1 {\_fsetVa #1,=}
185 \def \_fsetVa #1=#2,{\_isempty{#1}\_iffalse
186   \_ifx,#1\_else\_sdef{#1V}{#2}\_ea\_ea\_ea\_fsetVa\_fi\_fi
187 }
188 \def \_onlyif #1=#2:#3{%
189   \_expanded{\_noexpand\_isinlist{\_prepcommalist #2,\_fin},{\_cs{#1V}},}%
190   \_iftrue #3\_fi
191 }
192 \def\_prepcommalist#1,{\_ifx\_fin#1\_empty\_else #1,\_ea\_prepcommalist\_fi}
193 \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  $\langle modifier \rangle$  is defined as `\_famdepend\langle modifier \rangle{\_f:\_currfamily:\langle modifier \rangle}`. It expands to `\_f:\_currfamily:\langle modifier \rangle` value if it is defined or it prints the warning. When the `\_currfamily` value is changed then we can declare the same  $\langle modifier \rangle$  with a different meaning.

`\_setnewmeaning`  $\langle cs-name \rangle = \_tmpa$   $\langle by-what \rangle$  does exactly `\_let`  $\langle csname \rangle = \_tmpa$  but warning is printed if  $\langle cs-name \rangle$  is defined already and it is not a variant selector or font modifier.

`\_addtomodlist`  $\langle font modifier \rangle$  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
```

```

216 \_def \_moddef #1#2{%
217   \_edef\_tmp{\_csstring#1}%
218   \_sdef\_f:\_currfamily:\_tmp{\_addtomodlist#1#2}%
219   \_protected \_edef \_tmpa{\_noexpand\_famdepend\_noexpand#1\_f:\_noexpand\_currfamily:\_tmp}}%
220   \_setnewmeaning #1=\_tmpa \_moddef
221 }
222 \_protected \_def \_resetmod {\_cs{\_f:\_currfamily:resetmod}} % private variant of \resetmod
223 \_def \_resetfam{%
224   \_def\_addtomodlist##1{\_resetmod
225     \_edef \_modlist{\_ea}\_modlist
226     \_let\_addtomodlist=\_addtomodlistb
227     \_ifcsname \_f:\_currfamily:\_ea\_csstring \_currvar \_endcsname
228     \_else \_ea\_ifx\_currvar\_tt \_else \_def\_currvar{\_fontsel}\_fi
229     \_fi % corrected \_currvar in the new family
230 }
231 \_def \_currfamily{} % default current family is empty
232 \_def \_modlist{} % list of currently used modifiers
233
234 \_def \_addtomodlist#1{\_addto\_modlist#1}
235 \_let \_addtomodlistb=\_addtomodlist
236
237 \_def\_famdepend#1#2{\_ifcsname#2\_endcsname \_csname#2\_ea\_endcsname \_else
238   \_ifx\_addtomodlist\_addtomodlistb
239   \_opwarning{\_string#1 is undeclared in family "\_currfamily", ignored}\_fi\_fi
240 }
241 \_def\_setnewmeaning #1=\_tmpa#2{%
242   \_ifx #1\_undefined \_else \_ifx #1\_tmpa \_else
243     \_opwarning{\_string#1 is redefined by \_string#2}%
244     \_fi\_fi
245     \_let#1=\_tmpa
246 }
247 \_public \_moddef ;

```

`\fontdef`  $\langle font-switch \rangle$   $\{ \langle data \rangle \}$  does:

```
\begingroup  $\langle data \rangle$  \ea\endgroup \ea\let \ea\mathfont \the\font
```

It means that font modifiers used in  $\langle data \rangle$  are applied in the group and the resulting selected font (current at the end of the group) is set to the  $\langle font-switch \rangle$ . We want to declare  $\langle font-switch \rangle$  in its real name directly by `\font` primitive in order to save this name for reporting later (in overfull messages, for example). This is the reason why `\_xfamv` and `\_ttfamv` are re-defined locally here. They have precedence when `\fontsel` constructs the  $\langle font switch \rangle$  name.

```
fonts-select.opm
```

```

263 \_protected\_def \_fontdef #1#2{\_begingroup
264   \_edef\_xfamv{\_csstring#1}\_let\_ttfamv\_xfamv #2%
265   \_ea\_endgroup\_ea \_let\_ea #1\_the\_font
266 }
267 \_public \_fontdef ;

```

The `\famvardef`  $\backslash xxx$   $\{ \langle data \rangle \}$  does, roughly speaking:

```
\def \xxx { \mathfont \def\_currvar{\xxx} }
```

but the macro `\xxx` is declared as family-dependent. It is analogically as in `\moddef`. The `\xxx` is defined as `\_famdepend\xxx{\_f:\_currfamily:xxx}` and `\_f:\langle currfam \rangle:xxx` is defined as mentioned. `\famvardef` `\tt` behaves somewhat differently: it defines internal version `\_tt` (it is used in `\_ttfont` and `\_urlfont`) and set `\tt` to the same meaning.

```

283 \_protected\_def \_famvardef #1#2{%
284   \_sdef{f:\_currfamily:\_csstring#1}%
285   {{\_edef\_xfamv{\_csstring#1}\_let\_ttfamv\_xfamv #2\_ea}\_the\_font \_def\_currvar{#1}}%
286   \_protected\_edef\_tmpa {%
287     \_noexpand\_famdepend\_noexpand#1{f:\_noexpand\_currfamily:\_csstring#1}}%
288   \_ifx #1\_tt
289     \_protected\_def\_tt{{\_def\_xfamv{tt}#2\_ea}\_the\_font \_def\_currvar{\_tt}\_matt}%
290     \_let\_tt=\_tt
291   \_else \_setnewmeaning #1=\_tmpa \_famvardef
292   \_fi
293 }
294 \_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.

```

312 \_protected\_def \_fontfam [#1]{%
313   \_lowercase{\_edef\_famname{\_ea\_removespaces \_expanded{#1} {} }}%
314   \_isfile {f-\_famname.opm}\_iftrue \_edef\_famfile{f-\_famname}\_opinput {f-\_famname.opm}%
315   \_else
316     \_tryloadfamslocal
317     \_edef\_famfile{\_trycs{\_famf:\_famname}{}}%
318     \_ifx\_famfile\_empty
319       \_ifcsname \_fams:f-\_famname \_endcsname \_edef\_famfile{f-\_famname}\_famsubstitute
320     \_else \_listfamnames
321     \_fi
322   \_else \_opinput {\_famfile.opm}%
323   \_fi\_empty\_fi
324 }
325 \_def\_tryloadfamslocal{%
326   \_isfile {fams-local.opm}\_iftrue
327   \_opinput {fams-local.opm}\_famfrom={}\_famsrc={}%
328   \_fi
329   \_let \_tryloadfamslocal=\_relax % need not to load fams-local.opm twice
330 }
331 \_def\_listfamnames {%
332   \_wterm{==== List of font families =====}
333   \_begingroup
334     \_let\_famtext=\_wterm
335     \_def\_faminfo [#1]##2##3##4{%
336       \_wterm{ \_space\_noexpand\fontfam [#1] -- ##2}%
337       \_let\_famalias=\_famaliasA}%
338     \_opinput {fams-ini.opm}%
339     \_isfile {fams-local.opm}\_iftrue \_opinput {fams-local.opm}\_fi
340     \_message{^^J}%
341   \_endgroup
342 }
343 \_def\_famaliasA{\_message{ \_space\_space\_space\_space -- alias:}}
344 \_def\_famalias[##1]{\_message{[##1]}\_famalias
345 }
346 \_public \_fontfam ;

```

`\fontfamsub` [*Family*] [*byFamily*] declares automatic substitution of *Family* by *byFamily* which is done when *Family* is not installed. I.e. if there is no `f-family.opm` file or there is no regular font of the family installed. `\_famsubstitute` is internal macro used in `\fontfam` and `\_famdecl` macros. It consumes the rest of the macro, runs `\nospacefuturelet` in order to do `\endinginput` to the current `f-file` and runs `\fontfam` again. The table of such substitutions are saved in the macros `\_fams:family-file`.

```

359 \_def \_fontfamsub [#1]#2[#3]{\_tryloadfamslocal
360 \_lowercase{\_edef\_tmp{\_removespaces #1 {}}}%
361 \_sxdef{\_fams:\_trycs{\_famf:\_tmp}{f-\_tmp}}{#3}%
362 }
363 \_def\_famsubstitute #1\_empty\_fi{\_fi\_fi\_fi
364 \_wterm {FONT-SUB: \_famfile\_space -> [\_cs{\_fams:\_famfile}]]}%
365 \_nospacefuturelet\_tmp\_famsubstituteA % we want to \_endinput current f-file
366 }
367 \_def\_famsubstituteA{\_fontfam[\_cs{\_fams:\_famfile}]}
368
369 \_public \_fontfamsub ;

```

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>}` (only if *<file-name>* differs from `f-<familyname>`) or prints information on the terminal. The *<mod-plus-vars>* data are used when printing the font catalog.

`\_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.

`\_famsrc` declares the source, where is the font family from (used in `fams-ini.opm` and if the font isn't found when the fonts catalog is printed).

```

396 \_def\_famtext #1{}
397 \_def\_faminfo [#1]#2#3#4{%
398 \_lowercase{\_edef\_tmp{\_ea\_removespaces #1 {}}}%
399 \_edef\_tmpa{f-\_tmp}\_def\_famfile{#3}%
400 \_unless\_ifx\_tmpa\_famfile \_sdef{\_famf:\_tmp}{#3}\_fi
401 }
402 \_def\_famalias [#1]{%
403 \_lowercase{\_edef\_tmpa{\_ea\_removespaces #1 {}}}%
404 \_sdef{\_famf:\_tmpa\_ea}\_ea{\_famfile}%
405 }
406 \_newtoks\_famfrom \_newtoks\_famsrc
407 \_input fams-ini.opm
408 \_let\_famfile=\_undefined
409 \_famfrom={} \_famsrc={}

```

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`, `\_catalogexclude` and `\_catalognextfam` of the catalog are declared here.

```

422 \_newtoks \_catalogsample
423 \_newtoks \_catalogmathsample
424 \_newtoks \_catalogonly
425 \_newtoks \_catalogexclude
426 \_newtoks \_catalognextfam
427 \_catalogsample={ABCDabcd Qsty fi fl áéíóúü ã žč ĀĖĬŲ ŔŽČ 0123456789}
428 \_catalognextfam={\_bigskip}
429
430 \_public \_catalogonly \_catalogexclude \_catalogsample \_catalogmathsample \_catalognextfam ;

```

The font features are managed in the `\_fontfeatures` macro. It expands to

- `\_defaultfontfeatures` – used for each font,
- `\_ffadded` – features added by `\_setff`,
- `\_ffcolor` – features added by `\_setfontcolor` (this is obsolete)
- `\_ffletterspace` – features added by `\_setletterspace`,
- `\_ffwordspace` – features added by `\_setwordspace`.

The macros `\_ffadded`, `\_ffcolor`, `\_ffletterspace`, `\_ffwordspace` are empty by default.

fonts-select.opm

```
446 \_def \_fontfeatures{\_defaultfontfeatures\_ffadded\_ffcolor\_ffletterspace\_ffwordspace}
447 \_def \_defaultfontfeatures {+tlig;}
448 \_def \_ffadded{}
449 \_def \_ffcolor{}
450 \_def \_ffletterspace{}
451 \_def \_ffwordspace{}
```

The `\setff {⟨features⟩}` adds next font features to `\_ffadded`. Usage `\setff{}` resets empty set of all `\_ffadded` features.

fonts-select.opm

```
458 \_def \_setff #1{%
459   \_ifx^#1^\_def\_ffadded{}\\_else \_edef\_ffadded{\_ffadded #1;}\_fi
460 }
461 \_public \_setff ;
```

`\setletterspace` is based on the special font features provided by luaotfload package.

The `\setwordspace` recalculates the `\fontdimen2,3,4` of the font using the `\setwsp` macro which is used by the `\_fontselA` macro. It activates a dummy font feature `+Ws` too in order the font is reloaded by the `\font` primitive (with independent `\fontdimen` registers). If the `\setwordspace` is used again to the same font then we need to reset `\fontdimen` registers first. It is done by `\_sws:⟨fontname⟩` macro which keeps the original values of the `\fontdimen`s.

`\setfontcolor` is kept here only for backward compatibility but not recommended. Use real color switches and the `\transparency` instead.

fonts-select.opm

```
478 \_def \_setfontcolor #1{%
479   \_edef\_tmp{\_calculatefontcolor{#1}}%
480   \_ifx\_tmp\_empty \_def\_ffcolor{}\\_else \_edef\_ffcolor{color=\_tmp;}\_fi
481 }
482 \_def \_setletterspace #1{%
483   \_if^#1^\_def\_ffletterspace{}\\_else \_edef\_ffletterspace{letterspace=#1;}\_fi
484 }
485 \_def \_setwordspace #1{%
486   \_if^#1^\_def\_setwsp##1{}\\_def\_ffwordspace{}%
487   \_else \_def\_setwsp{\_setwspA#1/}\_def\_ffwordspace{+Ws;}\_fi
488 }
489 \_def\_setwsp #1{%
490   \_def\_setwspA #1{\_ifx/#1\_ea\_setwspB \_else\_afterfi{\_setwspC#1}\_fi}
491   \_def\_setwspB #1/#2/#3/#4{%
492     \_csname\_sws:\_fontname#4\_endcsname \_relax
493     \_ea\_xdef \_csname\_sws:\_fontname#4\_endcsname
494       {\_foreach 234\_do{\_fontdimen##1#4=\_the\_fontdimen##1#4}}%
495       \_fontdimen2#4=#1\_fontdimen2#4%
496       \_fontdimen3#4=#2\_fontdimen3#4\_fontdimen4#4=#3\_fontdimen4#4}
497   \_def\_setwspC #1/{\_setwspB #1/#1/#1/}
498 }
499 \_def\_calculatefontcolor#1{\_trycs{\_fc:#1}{#1}} % you can define more smart macro ...
500 \_sdef{\_fc:red}{FF0000FF} \_sdef{\_fc:green}{00FF00FF} \_sdef{\_fc:blue}{0000FFFF}
501 \_sdef{\_fc:yellow}{FFFF00FF} \_sdef{\_fc:cyan}{00FFFFFF} \_sdef{\_fc:magenta}{FF00FFFF}
502 \_sdef{\_fc:white}{FFFFFFFF} \_sdef{\_fc:grey}{00000080} \_sdef{\_fc:lgrey}{00000025}
503 \_sdef{\_fc:black}{} % ... you can declare more colors...
504
505 \_public \_setfontcolor \_setletterspace \_setwordspace ;
```

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

`\_optfontalias {⟨template A⟩}{⟨template B⟩}` is defined as `\let\_oz:⟨templateA⟩=\_oz:⟨templateB⟩`.

fonts-select.opm

```
518 \_def\_regoptsizes #1 #2?#3 #4*{\_sdef{\_oz:#1}{#2?#3 #4* }}
519 \_def\_optname #1{\_ifcsname\_oz:#1\_endcsname
520   \_ea\_ea\_ea \_optnameA \_csname\_oz:#1\_ea\_endcsname
521   \_else \_failedoptname{#1}\_fi
522 }
523 \_def\_failedoptname #1{optname-fails:(#1)}
```

```

524 \def\optnameA #1?#2 #3 <#4 {\_ifx*#4#1#3#2\_else
525 \_ifdim\_optsize<#4pt #1#3#2\_optnameC
526 \_else \_afterfifi \_optnameA #1?#2 \_fi\_fi
527 }
528 \def\optnameC #1* {\_fi\_fi}
529 \def\_afterfifi #1\_fi\_fi{\_fi\_fi #1}
530 \def\_optfontalias #1#2{\_slet{\_oz:#1}{\_oz:#2}}
531
532 \setfontsize {at10pt} % default font size

```

## 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 <2022-12-01>} % 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 cmmi % CM Math Italic
15 \_loadmathfamily 2 cmsy % CM Standard symbols
16 \_loadmathfamily 3 cmex % CM extra symbols
17 \_loadmathfamily 4 msam % AMS symbols A
18 \_loadmathfamily 5 msbm % AMS symbols B
19 \_loadmathfamily 6 rsfs % script
20 \_loadmathfamily 7 eufm % fractur
21 \_loadmathfamily 8 bfsans % sans serif bold
22 \_loadmathfamily 9 bisans % sans serif bold slanted (for vectors)
23 % \_setmathfamily 10 \_tentt
24 % \_setmathfamily 11 \_tenit
25 \_setmathdimens
26 }
27 \def\_boldmath{%
28 \_loadmathfamily 0 cmbx % CM Roman Bold Extended
29 \_loadmathfamily 1 cmmib % CM Math Italic Bold
30 \_loadmathfamily 2 cmbxy % CM Standard symbols Bold
31 \_loadmathfamily 3 cmexb % CM extra symbols Bold
32 \_loadmathfamily 4 msam % AMS symbols A (bold not available?)
33 \_loadmathfamily 5 msbm % AMS symbols B (bold not available?)
34 \_loadmathfamily 6 rsfs % script (bold not available?)
35 \_loadmathfamily 7 eufb % fractur bold
36 \_loadmathfamily 8 bbfsans % sans serif extra bold
37 \_loadmathfamily 9 bbisans % sans serif extra bold slanted (for vectors)
38 % \_setmathfamily 10 \_tentt
39 % \_setmathfamily 11 \_tenbi
40 \_setmathdimens
41 }
42 \def \normalmath {\_normalmath} \def \boldmath {\_boldmath}
math-preload.opm

```

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

These macros are redefined when `unimat-codes.opm` is loaded, see the section 2.16.2.

```

55 \chardef\bffam = 8
56 \chardef\bifam = 9
57 %\chardef\ttfam = 10
58 %\chardef\itfam = 11
59
60 \protected\def \marm {\fam0 }
61 \protected\def \mabf {\fam\bffam}
62 \protected\def \mait {\fam1 }
63 \protected\def \mabi {\fam\bifam}
64 \protected\def \matt {}
65
66 \protected\def \mit {\fam1 }
67 \protected\def \cal {\fam2 }
68 \protected\def \bbchar {\fam5 } % double stroked letters
69 \protected\def \frak {\fam7 } % fraktur
70 \protected\def \script {\fam6 } % more extensive script than \cal
71
72 \public \mit \cal \bbchar \frak \script ;

```

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

```

79 %% CM math fonts, optical sizes:
80
81 \regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
82             cmmi10 11.1 cmmi12 *
83 \regtfm cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
84 \regtfm cmte 0 cstex8 8.5 cstex9 9.5 cstex10 *
85 \regtfm cmsy 0 cmsy5 5.5 cmsy6 6.5 cmsy7 7.5 cmsy8 8.5 cmsy9 9.5 cmsy10 *
86 \regtfm cmb 0 cmb5 5.5 cmb6 6.5 cmb7 7.5 cmb8 8.5 cmb9 9.5 cmb10 *
87 \regtfm cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
88 \regtfm cmexb 0 cmexb10 *
89
90 \regtfm cmr 0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
91             cmr10 11.1 cmr12 15 cmr17 *
92 \regtfm cmbx 0 cmbx5 5.5 cmbx6 6.5 cmbx7 7.5 cmbx8 8.5 cmbx9 9.5
93             cmbx10 11.1 cmbx12 *
94 \regtfm cmti 0 cmti7 7.5 cmti8 8.5 cmti9 9.5 cmti10 11.1 cmti12 *
95 \regtfm cmtt 0 cmtt8 8.5 cmtt9 9.5 cmtt10 11.1 cmtt12 *
96
97 %% AMS math fonts, optical sizes:
98
99 \regtfm msam 0 msam5 5.5 msam6 6.5 msam7 7.5 msam8 8.5 msam9 9.5 msam10 *
100 \regtfm msbm 0 msbm5 5.5 msbm6 6.5 msbm7 7.5 msbm8 8.5 msbm9 9.5 msbm10 *
101
102 %% fraktur, rsfs, optical sizes:
103
104 \regtfm eufm 0 eufm5 6 eufm7 8.5 eufm10 *
105 \regtfm eufb 0 eufb5 6 eufb7 8.5 eufb10 *
106 \regtfm rsfs 0 rsfs5 6 rsfs7 8.5 rsfs10 *
107
108 %% bf and bi sansserif math alternatives:
109
110 \regtfm bfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
111             8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
112 \regtfm bisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
113             8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *
114 \regtfm bbfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
115             8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
116 \regtfm bbisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
117             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  $\langle number \rangle$  is saved by `\loadmathfamily`, `\setmathfamily` to the `\mfam`.

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\ptmunit` and it is set to 1pt by default.

`\_mfactor` sets scaling factor for given math fonts family related to text font size. It does the setting `\_ptmunit=(factor)\_ptmunit` where the  $\langle factor \rangle$  is defined by `\sdef{\_mfactor:\langle family \rangle}{\langle factor \rangle}`. For example, you can set `\sdef{\_mfactor:1}{0.95}` if you found that this scaling of math family 1 gives better visual compatibility with used text fonts. If not declared then scaling factor is 1.

math-preload.opm

```

146 \def\loadmathfamily {\afterassignment\loadmathfamilyA \chardef\_mfam}
147 \def\loadmathfamilyA #1 {\_mfactor
148 \edef\optsize{\_the\_optsize}%
149 \optsize=\_sizemtext \font\_mF \optfn{#1} at\_optsize \textfont\_mfam=\_mF
150 \optsize=\_sizemscript \font\_mF \optfn{#1} at\_optsize \scriptfont\_mfam=\_mF
151 \optsize=\_sizemsscript \font\_mF \optfn{#1} at\_optsize \scriptscriptfont\_mfam=\_mF
152 \optsize=\optsize\save
153 }
154 \def\setmathfamily {\afterassignment\setmathfamilyA \chardef\_mfam}
155 \def\setmathfamilyA #1 {\_mfactor \let\_mF=#1%
156 \edef\optsize{\_the\_optsize}%
157 \optsize=\_sizemtext \fontlet#1#1at\_optsize \textfont\_mfam=#1%
158 \optsize=\_sizemscript \fontlet#1#1at\_optsize \scriptfont\_mfam=#1%
159 \optsize=\_sizemsscript \fontlet#1#1at\_optsize \scriptscriptfont\_mfam=#1%
160 \optsize=\optsize\save \let#1=\_mF
161 }
162 \def\setmathsizes[#1/#2/#3]{\_ptmunit=\_ptmunit
163 \def\_sizemtext{#1\_ptmunit}\def\_sizemscript{#2\_ptmunit}%
164 \def\_sizemsscript{#3\_ptmunit}%
165 }
166 \def\_mfactor{\_ptmunit=\_trycs{\_mfactor:\_the\_mfam}{}}\_ptmunit}
167
168 \newdimen\_ptunit \_ptunit=1pt
169 \newdimen\_ptmunit \_ptmunit=1\_ptunit
170
171 \public \setmathsizes \ptunit ;

```

`\_setmathparam` $\langle latex-param \rangle$   $\{\langle factor \rangle\}$  sets  $\langle latex-param \rangle$  (like `\Umathspaceafterscript`) to values dependent on 1em of textfont, scriptfont, scriptscriptfont. The  $\langle factor \rangle$  is scaling factor of mentioned 1em.

math-preload.opm

```

180 \def\setmathparam#1#2{%
181 #1\_displaystyle =#2\_fontdimen6\_textfont1
182 #1\_crampeddisplaystyle =#2\_fontdimen6\_textfont1
183 #1\_textstyle =#2\_fontdimen6\_textfont1
184 #1\_crampedtextstyle =#2\_fontdimen6\_textfont1
185 #1\_scriptstyle =#2\_fontdimen6\_scriptfont1
186 #1\_crampedscriptstyle =#2\_fontdimen6\_scriptfont1
187 #1\_scriptscriptstyle =#2\_fontdimen6\_scriptscriptfont1
188 #1\_crampedscriptscriptstyle =#2\_fontdimen6\_scriptscriptfont1
189 }

```

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.

`\_setmathparam``\Umathspaceafterscript` is used instead `\scriptspace` setting because LuaT<sub>E</sub>X ignores `\scriptspace` in most cases. There is small difference from classical T<sub>E</sub>X: we set “scaled” `\Umathspaceafterscript` dependent on textstyle, scriptstyle, etc. sizes. The `\_scriptspacefactor` is set to 0.05 which gives the same result as Plain T<sub>E</sub>X `\scriptspace=0.5pt` at 10pt font size.

math-preload.opm

```

204 \def\setmathdimens{% PlainTeX sets these dimens for 10pt size only:
205 \delimitershortfall=0.5\_fontdimen6\_textfont3
206 \nulldelimiterspace=0.12\_fontdimen6\_textfont3
207 \setmathparam\Umathspaceafterscript \scriptspacefactor
208 \skewchar\_textfont1=127 \skewchar\_scriptfont1=127
209 \skewchar\_scriptscriptfont1=127
210 \skewchar\_textfont2=48 \skewchar\_scriptfont2=48
211 \skewchar\_scriptscriptfont2=48
212 \skewchar\_textfont6=127 \skewchar\_scriptfont6=127
213 \skewchar\_scriptscriptfont6=127
214 }
215 \def\_scriptspacefactor{.05}

```

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  $\TeX$  distribution installed.

math-preload.opm

```
225 \_suppressfontnotfounderror=1
226 \_setmathsizes[10/7/5]
227 \_ifx\fontspreload\_relax \_else \_normalmath \_fi
228 \_suppressfontnotfounderror=0
```

## 2.15 Math macros

math-macros.opm

```
3 \_codedecl \sin {Math macros plus mathchardefs <2026-03-18>} % 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 \_ifmathsb \_else \_directlua{
37 callback.add_to_callback("process_input_buffer",
38 function (str)
39 local num
40 str, num = string.gsub(str.." ", \_gsubrule)
41 if num>0 then str = string.gsub(str, \_gsubrule) end % \phi _i\rho _j -> \phi _i\rho _j
42 return str
43 end, "_mathsb")} \_fi
44 \_global\_mathsbtrue
45 }
46 \_def \_mathsboff {%
47 \_ifmathsb \_directlua{ callback.remove_from_callback("process_input_buffer", "_mathsb")} \_fi
48 \_global \_mathsbfalse
49 }
50 \_edef\_gsubrule{"(\_nbb[a-zA-Z]+)([a-zA-Z]?[^\_a-zA-Z])", "\_pcent 1 \_pcent 2"}
51
52 \_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

```
60 \_mathcode\^{}="2201 % \cdot
61 \_mathcode\^{}A="3223 % \downarrow
62 \_mathcode\^{}B="010B % \alpha
63 \_mathcode\^{}C="010C % \beta
64 \_mathcode\^{}D="225E % \land
65 \_mathcode\^{}E="023A % \lnot
66 \_mathcode\^{}F="3232 % \in
67 \_mathcode\^{}G="0119 % \pi
68 \_mathcode\^{}H="0115 % \lambda
69 \_mathcode\^{}I="010D % \gamma
70 \_mathcode\^{}J="010E % \delta
```

```

71 \_mathcode\^^K="3222 % \uparrow
72 \_mathcode\^^L="2206 % \pm
73 \_mathcode\^^M="2208 % \oplus
74 \_mathcode\^^N="0231 % \infty
75 \_mathcode\^^O="0140 % \partial
76 \_mathcode\^^P="321A % \subset
77 \_mathcode\^^Q="321B % \supset
78 \_mathcode\^^R="225C % \cap
79 \_mathcode\^^S="225B % \cup
80 \_mathcode\^^T="0238 % \forall
81 \_mathcode\^^U="0239 % \exists
82 \_mathcode\^^V="220A % \otimes
83 \_mathcode\^^W="3224 % \leftrightharpoonup
84 \_mathcode\^^X="3220 % \leftarrow
85 \_mathcode\^^Y="3221 % \rightarrow
86 \_mathcode\^^Z="8000 % \neq
87 \_mathcode\^^[="2205 % \diamond
88 \_mathcode\^^\="3214 % \leq
89 \_mathcode\^^]= "3215 % \geq
90 \_mathcode\^^^="3211 % \equiv
91 \_mathcode\^^_="225F % \lor
92 \_mathcode\ \_="8000 % \space
93 \_mathcode\ != "5021
94 \_mathcode\ '="8000 % ^\prime
95 \_mathcode\ (="4028
96 \_mathcode\ )="5029
97 \_mathcode\ *="2203 % \ast
98 \_mathcode\ += "202B
99 \_mathcode\ \,="613B
100 \_mathcode\ -= "2200
101 \_mathcode\ \,="013A
102 \_mathcode\ \,="013D
103 \_mathcode\ := "303A
104 \_mathcode\ ;="603B
105 \_mathcode\ \leq="313C
106 \_mathcode\ \leq="303D
107 \_mathcode\ \geq="313E
108 \_mathcode\ ?="503F
109 \_mathcode\ [= "405B
110 \_mathcode\ \backslash="026E % \backslash
111 \_mathcode\ \,="505D
112 \_mathcode\ \_="8000 % math-active subscript
113 \_mathcode\ \{="4266
114 \_mathcode\ \,="026A
115 \_mathcode\ \}="5267
116 \_mathcode\ \^?="1273 % \smallint
117
118 \_delcode\ (= "028300
119 \_delcode\ )="029301
120 \_delcode\ [= "05B302
121 \_delcode\ \,="05D303
122 \_delcode\ \leq="26830A
123 \_delcode\ \geq="26930B
124 \_delcode\ \,="02F30E
125 \_delcode\ \,="26A30C
126 \_delcode\ \backslash="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

```

139 \_mathchardef\alpha="010B
140 \_mathchardef\beta="010C
141 \_mathchardef\gamma="010D
142 \_mathchardef\delta="010E
143 \_mathchardef\epsilon="010F
144 \_mathchardef\zeta="0110

```

```

145 \_mathchardef\eta="0111
146 \_mathchardef\theta="0112
147 \_mathchardef\iota="0113
148 \_mathchardef\kappa="0114
149 \_mathchardef\lambda="0115
150 \_mathchardef\mu="0116
151 \_mathchardef\nu="0117
152 \_mathchardef\xi="0118
153 \_mathchardef\pi="0119

```

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

The math functions like `log`, `sin`, `cos` are declared in the same way as in plain $\TeX$ , but they are `\protected` in Op $\TeX$ .

`math-macros.opm`

```

311 \_protected\_def\log {\_mathop{\_rm log}\_nolimits}
312 \_protected\_def\lg {\_mathop{\_rm lg}\_nolimits}
313 \_protected\_def\ln {\_mathop{\_rm ln}\_nolimits}
314 \_protected\_def\lim {\_mathop{\_rm lim}}
315 \_protected\_def\limsup {\_mathop{\_rm lim\_think sup}}
316 \_protected\_def\liminf {\_mathop{\_rm lim\_think inf}}
317 \_protected\_def\sin {\_mathop{\_rm sin}\_nolimits}
318 \_protected\_def\arcsin {\_mathop{\_rm arcsin}\_nolimits}
319 \_protected\_def\sinh {\_mathop{\_rm sinh}\_nolimits}
320 \_protected\_def\cos {\_mathop{\_rm cos}\_nolimits}
321 \_protected\_def\arccos {\_mathop{\_rm arccos}\_nolimits}
322 \_protected\_def\cosh {\_mathop{\_rm cosh}\_nolimits}
323 \_protected\_def\tan {\_mathop{\_rm tan}\_nolimits}
324 \_protected\_def\arctan {\_mathop{\_rm arctan}\_nolimits}
325 \_protected\_def\tanh {\_mathop{\_rm tanh}\_nolimits}
326 \_protected\_def\cot {\_mathop{\_rm cot}\_nolimits}
327 \_protected\_def\coth {\_mathop{\_rm coth}\_nolimits}
328 %\_protected\_def\sec {\_mathop{\_rm sec}\_nolimits} % \sec is section
329 \_protected\_def\secant {\_mathop{\_rm sec}\_nolimits}
330 \_protected\_def\csc {\_mathop{\_rm csc}\_nolimits}
331 \_protected\_def\max {\_mathop{\_rm max}}
332 \_protected\_def\min {\_mathop{\_rm min}}
333 \_protected\_def\sup {\_mathop{\_rm sup}}
334 \_protected\_def\inf {\_mathop{\_rm inf}}
335 \_protected\_def\arg {\_mathop{\_rm arg}\_nolimits}
336 \_protected\_def\ker {\_mathop{\_rm ker}\_nolimits}
337 \_protected\_def\dim {\_mathop{\_rm dim}\_nolimits}
338 \_protected\_def\hom {\_mathop{\_rm hom}\_nolimits}
339 \_protected\_def\det {\_mathop{\_rm det}}
340 \_protected\_def\exp {\_mathop{\_rm exp}\_nolimits}
341 \_protected\_def\Pr {\_mathop{\_rm Pr}}
342 \_protected\_def\gcd {\_mathop{\_rm gcd}}
343 \_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.

`\sp` is an alternative for `^`. The `\sb` alternative for `_` was defined at line 27 of the file `math-macros.opm`. `\_think`, `\_medsk`, `\_thicksk` and `\_thinneg` should be used instead `\,`, `\>`, `\;` and `\!` in macros because a user can re-define these single-letter sequences.

`math-macros.opm`

```

356 \_let\_sp=^ \public \sp ;
357 % \sb=_, defined at beginning of this file
358
359 \_def\_think {\_mskip\_thinmskip}
360 \_protected\_def\,{\_relax\_ifmmode \_think \_else \_thinspace \_fi}
361 \_protected\_def\>{\_mskip\_medmskip} \_let\_medsk = \>
362 \_protected\_def\;{\_mskip\_thickmskip} \_let\_thicksk = \;
363 \_protected\_def\!{\_mskip\_thinmskip} \_let\_thinneg = \!
364 %\_def\*{\discretionary{\_thinspace\the\_textfont2\char2\char2}{}} % obsolete

```

Active `\prime` character is defined here.

`math-macros.opm`

```

370 {\_catcode\'= \active \_gdef{\^\_bgroup\_primes}} % primes dance
371 \_def\_primes{\_prime\_isnextchar'\\_primesA}%
372 \_isnextchar^{\_primesB}{\_egroup}}

```

```

373 \def\primesA #1{\primes}
374 \def\primesB #1#2{#2\egroup}
375 \private \prime ;

```

`\big`, `\bbig`, `\Big`, `\bBig`, `\bigg`, `\Bigg`, `\bigl`, `\bigm`, `\bigr`, `\bbigl`, `\bbigm`, `\bbigr`, `\bBigl`, `\Bigl`, `\Bigm`, `\bBigm`, `\Bigr`, `\bBigr`, `\biggl`, `\biggm`, `\biggr`, `\Biggl`, `\Biggm`, `\Biggr` are based on the `\scalebig⟨fence⟩⟨num⟩`; macro because we need the dependency on the various sizes of the fonts. The `\scalebigcoef⟨num⟩` returns relevant coefficient for these macros. Multiply this coefficient by two and you get the strut height+depth in em units. If the `⟨num⟩` is bigger than 6 then the next parameter is used directly, for example `\def\morebig#1{\scalebig{#1}7{1.8}}`

The `\big`, `\Big`, `\bigg`, `\Bigg` macros keep the strut height+depth from plain T<sub>E</sub>X and `\bbig` is a new macro in OpT<sub>E</sub>X. It generates the size 1.44em between `\big` and `\Big` which is accessible in most of Unicode math fonts (but not in classical cmex10).

math-macros.opm

```

394 %{\catcode\^^Z=\active \gdef^^Z{\not=}} % ^^Z is like \ne in math %obsolete
395
396 \def\scalebig#1#2#3{ {\left#1%
397   \raise\Umathaxis\textstyle\ vbox to\scalebigcoef{#2}{#3}\fontdimen6\textfont1}%
398   \kern-\nulldelimiterspace\right.} }
399 \def\scalebigcoef#1#2{\ifcase #1 0\or
400 % \big (1.22) \bbig (1.44) \Big (1.8) \Bbig (2.1) \bigg (2.4) \Bigg (3.0) % \scalebig7{scale}
401   .62\or .72\or .9\or 1.05\or 1.2\or 1.5\else #2\fi}
402 \protected\def\big #1{\scalebig{#1}1;}
403 \protected\def\bbig#1{\scalebig{#1}2;}
404 \protected\def\Big #1{\scalebig{#1}3;}
405 \protected\def\bBig#1{\scalebig{#1}4;}
406 \protected\def\bigg#1{\scalebig{#1}5;}
407 \protected\def\Bigg#1{\scalebig{#1}6;}
408 \public \big \bbig \Big \bBig \bigg \Bigg ; % only \big \Big \bigg \Bigg in original plain TeX
409
410 \protected\def\bigl{\mathopen\big}
411 \protected\def\bigm{\mathrel\big}
412 \protected\def\bigr{\mathclose\big}
413 \protected\def\bbigl{\mathopen\bbig}
414 \protected\def\bbigm{\mathrel\bbig}
415 \protected\def\bbigr{\mathclose\bbig}
416 \protected\def\Bigl{\mathopen\Big}
417 \protected\def\bBigl{\mathopen\bBig}
418 \protected\def\Bigm{\mathrel\Big}
419 \protected\def\bBigm{\mathrel\bBig}
420 \protected\def\Bigr{\mathclose\Big}
421 \protected\def\bBigr{\mathclose\bBig}
422 \protected\def\biggl{\mathopen\bigg}
423 \protected\def\biggm{\mathrel\bigg}
424 \protected\def\biggr{\mathclose\bigg}
425 \protected\def\Biggl{\mathopen\Bigg}
426 \protected\def\Biggm{\mathrel\Bigg}
427 \protected\def\Biggr{\mathclose\Bigg}
428 \public \bigl \bigm \bigr \bbigl \bbigm \bbigr
429   \Bigl \Bigm \Bigr \bBigl \bBigm \bBigr \biggl \biggm \biggr \Biggl \Biggm \Biggr ;

```

Math relations defined by the `\jointrel` plain T<sub>E</sub>X macro:

math-macros.opm

```

435 \protected\def\joinrel{\mathrel{\mkern-2.5mu}} % -3mu in plainTeX
436 \protected\def\relbar{\mathrel{\smash{-}}} % \smash, because - has the same height as +
437 \protected\def\Relbar{\mathrel=}
438 \mathchardef\lhook="312C
439 \protected\def\hookrightarrow{\lhook\joinrel\rightarrow}
440 \mathchardef\rhook="312D
441 \protected\def\hookleftarrow{\leftarrow\joinrel\rhook}
442 \protected\def\bowtie{\mathrel\triangleright\joinrel\mathrel\triangleleft}
443 \protected\def\models{\mathrel|\joinrel=}
444 \protected\def\Longrightarrow{\Relbar\joinrel\rightarrow}
445 \protected\def\longrightarrow{\relbar\joinrel\rightarrow}
446 \protected\def\longleftarrow{\leftarrow\joinrel\relbar}
447 \protected\def\Longleftarrow{\Leftarrow\joinrel\Relbar}
448 \protected\def\longmapsto{\mapstochar\longrightarrow}
449 \protected\def\longlefttrightarrow{\leftarrow\joinrel\rightarrow}

```

```

450 \protected\def\Longleftarrow{\Leftarrow\joinrel\Rightarrow}
451 \protected\def\iff{\thicksk\Longleftarrow\thicksk}
452 \private \hook \rightarrow \leftarrow \rhook \triangleright \triangleleft
453 \Relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
454 \longrightarrow \Longleftarrow ;
455 \public \joinrel ;

```

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

math-macros.opm

```

461 \mathchardef\ldotp="613A % ldot as a punctuation mark
462 \mathchardef\cdotp="6201 % cdot as a punctuation mark
463 \mathchardef\colon="603A % colon as a punctuation mark
464 \public \ldotp \cdotp \colon ;
465
466 \protected\def\ldots{\mathinner{\ldotp\ldotp\ldotp}}
467 \protected\def\cdots{\mathinner{\cdotp\cdotp\cdotp}}
468 \protected\def\vdots{\vbox{\baselineskip=.4em \lineskiplimit=\_zo
469 \kern.6em \hbox{.}\hbox{.}\hbox{.}}}
470 \protected\def\ddots{\mathinner{%
471 \mkern1mu\raise.7em\vbox{\kern.7em\hbox{.}}\mkern2mu
472 \raise.4em\hbox{.}\mkern2mu\raise.1em\hbox{.}\mkern1mu}}
473
474 \public \ldots \cdots \vdots \ddots ;

```

`\adots` inspired by plain T<sub>E</sub>X

math-macros.opm

```

480 \protected\def\adots{\mathinner{%
481 \mkern1mu\raise.1em\hbox{.}\mkern2mu
482 \raise.4em\hbox{.}\mkern2mu\raise.7em\vbox{\kern.7em\hbox{.}}\mkern1mu}}
483
484 \public \adots ;

```

Math accents (encoding dependent declarations).

math-macros.opm

```

490 \protected\def\acute{\mathaccent"7013 }
491 \protected\def\grave{\mathaccent"7012 }
492 \protected\def\ddot{\mathaccent"707F }
493 \protected\def\tilde{\mathaccent"707E }
494 \protected\def\bar{\mathaccent"7016 }
495 \protected\def\breve{\mathaccent"7015 }
496 \protected\def\check{\mathaccent"7014 }
497 \protected\def\hat{\mathaccent"705E }
498 \protected\def\vec{\mathaccent"017E }
499 \protected\def\dot{\mathaccent"705F }
500 \protected\def\widetilde{\mathaccent"0365 }
501 \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

```

509 \def\_math{\mathsurround\_zo}
510 \protected\def\_skew #1#2#3{(\_muskip0=#1mu\_divide\_muskip0by2 \_mkern\_muskip0
511 #2{\_mkern-\_muskip0{#3}\_mkern\_muskip0}\_mkern-\_muskip0){}}
512 \protected\def\_overrightarrow #1{\vbox{\_math\_ialign{##\_crr
513 \rightarrowfill\_crr\_noalign{\_kern-.1em \_nointerlineskip}
514 $\_hfil\_displaystyle{#1}\_hfil$\_crr}}}
515 \protected\def\_overleftarrow #1{\vbox{\_math\_ialign{##\_crr
516 \leftarrowfill\_crr\_noalign{\_kern-.1em \_nointerlineskip}
517 $\_hfil\_displaystyle{#1}\_hfil$\_crr}}}
518 \protected\def\_overbrace #1{\_mathop{%
519 \vbox{\_math\_ialign{##\_crr\_noalign{\_kern.3em}
520 \downbracefill\_crr\_noalign{\_kern.3em \_nointerlineskip}
521 $\_hfil\_displaystyle{#1}\_hfil$\_crr}}}\_limits}
522 \protected\def\_underbrace #1{\_mathop{\_vtop{\_math\_ialign{##\_crr
523 $\_hfil\_displaystyle{#1}\_hfil$\_crr\_noalign{\_kern.3em \_nointerlineskip}
524 \upbracefill\_crr\_noalign{\_kern.3em}}}}}\_limits}
525
526 \public \overrightarrow \overleftarrow \overbrace \underbrace \skew ;

```

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



```

532 \protected\def\lmoustache{\delimiter"437A340 } % top from (, bottom from )
533 \protected\def\rmoustache{\delimiter"537B341 } % top from ), bottom from (
534 \protected\def\lgroup{\delimiter"462833A } % extensible ( with sharper tips
535 \protected\def\rgroup{\delimiter"562933B } % extensible ) with sharper tips
536 \protected\def\arrowvert{\delimiter"26A33C } % arrow without arrowheads
537 \protected\def\Arrowvert{\delimiter"26B33D } % double arrow without arrowheads
538 \protected\def\bracevert{\delimiter"77C33E } % the vertical bar that extends braces
539 \protected\def\Vert{\delimiter"26B30D } \let\|= \Vert
540 \protected\def\vert{\delimiter"26A30C }
541 \protected\def\uparrow{\delimiter"3222378 }
542 \protected\def\downarrow{\delimiter"3223379 }
543 \protected\def\updownarrow{\delimiter"326C33F }
544 \protected\def\Uparrow{\delimiter"322A37E }
545 \protected\def\Downarrow{\delimiter"322B37F }
546 \protected\def\Updownarrow{\delimiter"326D377 }
547 \protected\def\backslash{\delimiter"26E30F } % for double coset G\backslash H
548 \protected\def\langle{\delimiter"426830A }
549 \protected\def\rangle{\delimiter"526930B }
550 \protected\def\lbrace{\delimiter"4266308 } \let\lbrace=\lbrace
551 \protected\def\rbrace{\delimiter"5267309 } \let\rbrace=\rbrace
552 \protected\def\{{\ifmmode \lbrace\else\char`\{ \fi}
553 \protected\def\}{\ifmmode \rbrace\else\char`\} \fi}
554
555 \protected\def\rceil{\delimiter"5265307 }
556 \protected\def\lceil{\delimiter"4264306 }
557 \protected\def\rfloor{\delimiter"5263305 }
558 \protected\def\lfloor{\delimiter"4262304 }
559
560 \protected\def\choose{\atopwithdelims()}
561 \protected\def\brack{\atopwithdelims[]}
562 \protected\def\brace{\atopwithdelims\lbrace\rbrace}
563
564 \protected\def\sqrt{\radical"270370 } \public \sqrt ;

```

`\mathpalette`, `\vphantom`, `\hphantom`, `\phantom`, `\mathstrut`, and `\smash` macros from plain  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ .

```

571 \def\mathpalette#1#2{\mathchoice{#1\displaystyle{#2}}%
572 {#1\textstyle{#2}}{#1\scriptstyle{#2}}{#1\scriptscriptstyle{#2}}}
573 \newbox\rootbox
574 \protected\def\root#1of{\setbox\rootbox
575 \hbox{$\math\scriptscriptstyle{#1}$}\mathpalette\rootA}
576 \def\rootA#1#2{\setbox0=\hbox{$\math#1\sqrt{#2}$}\dimen0=\ht0
577 \advance\dimen0by-\dp0
578 \mkern5mu\raise.6\dimen0\copy\rootbox \mkern-10mu\box0 }
579 \newif\ifvp \newif\ifhp
580 \protected\def\vphantom{\vptrue\hpfalse\phant}
581 \protected\def\hphantom{\vpfalse\hptrue\phant}
582 \protected\def\phantom{\vptrue\hptrue\phant}
583 \def\phant{\ifmmode\def\next{\mathpalette\mathphant}%
584 \else\let\next=\makephant\fi\next}
585 \def\makephant#1{\setbox0\hbox{#1}\finphant}
586 \def\mathphant#1#2{\setbox0=\hbox{$\math#1{#2}$}\finphant}
587 \def\finphant{\setbox2=\null
588 \ifvp \ht2=\ht0 \dp2=\dp0 \fi
589 \ifhp \wd2=\wd0 \fi \hbox{\box2}}
590 \def\mathstrut{\vphantom{}}
591 \protected\def\smash{\relax % \relax, in case this comes first in \halign
592 \ifmmode\def\next{\mathpalette\mathsmash}\else\let\next\makesmash
593 \fi\next}
594 \def\makesmash#1{\setbox0=\hbox{#1}\finsmash}
595 \def\mathsmash#1#2{\setbox0=\hbox{$\math#1{#2}$}\finsmash}
596 \def\finsmash{\ht0=\zo \dp0=\zo \hbox{\box0}}
597 \public \mathpalette \vphantom \hphantom \phantom \mathstrut \smash ;

```

`\cong`, `\notin`, `\rightleftharpoons`, `\buildrel`, `\doteq`, `\bmod` and `\pmod` macros from plain  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ .

```

604 \protected\def\cong{\mathrel{\mathpalette\overeq\sim}} % congruence sign
605 \def\overeq#1#2{\lower.05em\box{\lineskiplimit\maxdimen\lineskip=-.05em
606 \ialign{$\math#1\hfil##\hfil$\crrc#2\crrc=\crrc}}}
607 \protected\def\notin{\mathrel{\mathpalette\icancel\in}}

```

```

608 \def\icancel#1#2{\math\oalign{$\_hfil#1\_mkern1mu/\_hfil$\_crrc$#1#2$}}
609 \protected\_def\_rightleftharpoons{\mathrel{\_mathpalette\_rlhp{}}}
610 \def\_rlhp#1{\_vcenter{\_math\_hbox{\_oalign{\_raise.2em
611 \_hbox{$#1\_rightharpoonup$}\_crrc
612 $#1\_leftharpoondown$}}}}
613 \protected\_def\_buildrel#1\over#2{\mathrel{\_mathop{\_kern\_zo #2}\_limits^{#1}}}
614 \protected\_def\_doteq{\_buildrel\_textstyle.\over=}
615 \private \in \sim ;
616 \public \cong \notin \rightleftharpoons \buildrel \doteq ;
617
618 \protected\_def\_bmod{\_nonscript\_mskip-\_medmuskip\_mkern5mu
619 \_mathbin{\_rm mod}\_penalty900\_mkern5mu\_nonscript\_mskip-\_medmuskip}
620 \protected\_def\_pmod#1{\_allowbreak\_mkern18mu({\_rm mod}\\_thinsk\_thinsk#1)}
621 \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

```

631 \protected\_def\_matrix#1{\_null\_thinsk
632 \_edef\_tmpa{\_the\_numexpr \_mathstyle/4\_relax}% 0 0 1 1 1 1 2 2
633 \_vcenter{\_matrixbaselines\_math
634 \_ialign{\_the\_lmfil$\_matrixstyle##$\_hfil&\_quad\_the\_lmfil$\_matrixstyle##$\_hfil\_crrc
635 \_mathstrut\_crrc\_noalign{\_kern-\_baselineskip}
636 #1\_crrc\_mathstrut\_crrc\_noalign{\_kern-\_baselineskip}}}\_thinsk}
637
638 \def\_matrixbaselines{\_normalbaselines \_def\_matrixstyle{}%
639 \_let\_matrixbaselines=\_relax % \matrix inside matrix does not change size again
640 \_ifcase\_tmpa \_or
641 \_baselineskip=.7\_baselineskip \_def\_quad {\_hskip.7em\_relax}%
642 \_let\_matrixstyle=\_scriptstyle
643 \_or
644 \_baselineskip=.5\_baselineskip \_def\_quad {\_hskip.5em\_relax}%
645 \_let\_matrixstyle=\_scriptscriptstyle
646 \_fi
647 }
648 \protected\_def\_pmatrix#1{\_left(\_matrix{#1}\_right)}
649
650 \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

```

658 \protected\_long\_def\_cases#1{\_left{\_thinsk\_vcenter{\_normalbaselines\_math
659 \_ialign{##$\_hfil&\_quad{##$\_unsskip}\_hfil\_crrc#1\_crrc}}\_right.}
660
661 \_newdimen\_ptrenwd
662 \_ptrenwd=8.75pt % width of the big left (
663 \protected\_def\_bordermatrix{\_bordermatrixwithdelims()}
664 \_def\_bordermatrixwithdelims#1#2#3{\_begingroup \_math
665 \_setbox0=\_vbox{\_bordermatrixA #3\_stopbmatrix}%
666 \_setbox2=\_vbox{\_unvcopy0 \_global\_setbox1=\_lastbox}%
667 \_setbox2=\_hbox{\_unhbox1 \_unskip\_global\_setbox1=\_lastbox}%
668 \_setbox2=\_hbox{$\_kern\_wd1 \_kern-\_ptrenwd\_left#1\_kern-\_wd1
669 \_global\_setbox1=\_vbox{\_box1 \_kern.2em}%
670 \_vcenter{\_kern-\_ht1 \_unvbox0 \_kern-\_baselineskip}\_thinsk\_right#2$}%
671 \_null\_thinsk\_vbox{\_kern\_ht1 \_box2}\_endgroup}
672 \_def\_bordermatrixA #1\cr#2\_stopbmatrix{%
673 \_ialign{##$\_hfil\_kern.2em\_kern\_ptrenwd&\_thinspace\_hfil$##$\_hfil
674 &\_quad\_hfil$##$\_hfil\_crrc
675 \_omit\_strut\_hfil\_crrc\_noalign{\_kern-\_baselineskip}%
676 #1\_crrc\_noalign{\_kern.2em}\_2\_crrc\_omit\_strut\_cr}}
677
678 \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 OpTeX, this macro is more flexible. See section 4.4 in the [Typesetting Math with OpTeX](#). 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 TeX): `rlcr` 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

```
699 \long\def\eqalign#1{\_null\_thinsk\_vcenter{\_the\_eqlines\_math
700 \_ialign{&\_hfil$_\_the\_eqstyle{##}$&$_\_the\_eqstyle{{}##}$\_hfil
701 &\_hskip.5\_eqspace\_hfil$_\_the\_eqstyle{##}$\_hskip.5\_eqspace\_hfil
702 \_crrcr#1\_crrcr}}\_thinsk}
703
704 \_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`.

OpTeX offers `\displaylines to⟨dimen⟩{⟨formula⟩\cr⟨formula⟩\cr...⟨formula⟩}` as an alternative case of usage `\displaylines`. See section 4.3 in the [Typesetting Math with OpTeX](#). 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

```
724 \def\displaylines #1#{\_ifx&#1&\_ea\_displaylinesD
725 \_else \_def\_tmp to#1\_end{\_def\_tmp{\_dimexpr #1}\\_tmp #1\_end
726 \_ea\_displaylinesto \_fi}
727 \long\def\displaylinesD #1{\_display \_tabskip=\_zoskip
728 \_halign{\_hbox to\_displaywidth{\\_elign\_hfil\_displaystyle##\_hfil}\\_crrcr
729 #1\_crrcr}}
730 \long\def\displaylinesto #1{\_vcenter{\_openup\_jot \_math \_tabskip=\_zoskip
731 \_halign{\_strut\_hbox to\_span\_tmp{\\_hss\_displaystyle##\_hss}\\_crrcr
732 #1\_crrcr}}}
733
734 \_public\displaylines ;
```

`\openup`, `\eqalignno` and `\leqalignno` macros are inspired from Plain TeX, but internal macros `\begeqanno`, `\eqnoright`, `\eqnoleft` are used to make the macros more economical.

math-macros.opm

```
742 \def\openup{\_afterassignment\_openupA\_dimen0=}
743 \def\openupA{\_advance\_lineskip by\_dimen0
744 \_advance\_baselineskip by\_dimen0
745 \_advance\_lineskiplimit by\_dimen0 }
746 \_newifi\_ifdtop
747 \def\display{\_global\_dtoptrue\_openup\_jot\_math
748 \_everycr{\_noalign{\_ifdtop \_global\_dtopfalse \_ifdim\_prevdepth>-1000pt
749 \_vskip-\_lineskiplimit \_vskip\_normallineskiplimit \_fi
750 \_else \_penalty\_interdisplaylinepenalty \_fi}}}
751 \def\_elign{\_tabskip=\_zoskip\_everycr{}} % restore inside \_display
752 \long\def\eqalignno#1{\_begeqanno &\_eqnoright{##}\_tabskip=\_zoskip\_crrcr #1\_crrcr\_egroup}
753 \long\def\leqalignno#1{\_begeqanno &\_eqnoleft{##}\_tabskip=\_zoskip\_crrcr #1\_crrcr\_egroup}
754 \def\_begeqanno{\_unless\_ifnum\_predisplaydirection=0 \_swapeqno\_fi
755 \_display \_tabskip=\_centering
756 \_halign to\_displaywidth{\_bgroup\_hfil$_\_elign\_displaystyle{##}$\_tabskip=\_zoskip
757 &$_\_elign\_displaystyle{{}##}$\_hfil\_tabskip\_centering}
758 \_def\_eqnoright#1{\_hbox to\_zo{\_hss$_\_elign#1$}}
759 \_def\_eqnoleft #1{\_hbox to\_zo{\_kern-\_displaywidth$_\_elign#1$_\_hss}}
760
761 \_public \openup \eqalignno \leqalignno ;
```

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

math-macros.opm

```
768 \def\_amsafam{4} \def\_amsbfam{5}
769
770 \_mathchardef \boxdot "2\_amsafam 00
771 \_mathchardef \boxplus "2\_amsafam 01
772 \_mathchardef \boxtimes "2\_amsafam 02
773 \_mathchardef \square "0\_amsafam 03
774 \_mathchardef \blacksquare "0\_amsafam 04
775 \_mathchardef \centerdot "2\_amsafam 05
776 \_mathchardef \lozenge "0\_amsafam 06
```

```

777 \mathchardef \blacklozenge "0\amsafam 07
778 \mathchardef \circlearrowright "3\amsafam 08
779 \mathchardef \circlearrowleft "3\amsafam 09
780 \mathchardef \rightleftharpoons "3\amsafam 0A
781 \mathchardef \leftrightharpoons "3\amsafam 0B
782 \mathchardef \boxminus "2\amsafam 0C

```

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

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

```

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

```

`math-macros.opm`

```

1017 \mathchardef \_notchar "3236
1018
1019 \protected\_def \_not#1{%
1020 \_ifx #1<\_nless \_else
1021 \_ifx #1>\_ngtr \_else
1022 \_edef\_tmpn{\_csstring#1}%
1023 \_ifcsname \_not\_tmpn\_endcsname \_csname \_not\_tmpn\_endcsname
1024 \_else \_ifcsname \_n\_tmpn\_endcsname \_csname \_n\_tmpn\_endcsname
1025 \_else \_mathrel{\_mathord{\_notchar}\_mathord{#1}}%
1026 \_fi \_fi \_fi \_fi}
1027 \_private
1028 \nleq \ngeq \_nless \_ngtr \_nprec \_nsucc \_nleqslant \_ngeqslant \_npreceq
1029 \nsucceq \_nleqq \_ngeqq \_nsim \_ncong \_nsubseteqq \_nsupseteqq \_nsubseteq
1030 \_nsupseteq \_nparallel \_nmid \_nshortmid \_nshortparallel \_nvDash \_nVdash
1031 \_nvDash \_nVDash \_ntrianglerighteq \_ntrianglelefteq \_ntriangleleft
1032 \_ntriangleright \_nleftarrow \_nrightarrow \_nLeftarrow \_nRightarrow
1033 \_nLeftrightarrow \_nleftrightharrow \_nexists ;
1034 \_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`

```

1054 \_newcount\_stylenum
1055 \_def\_mathstyles#1{\_mathchoice{\_stylenum0 #1}{\_stylenum1 #1}%
1056 \_stylenum2 #1}{\_stylenum3 #1}}
1057 \_def\_dobystyle#1#2#3#4{\_ifcase\_stylenum#1\_or#2\_or#3\_or#4\_fi}
1058 \_def\_currstyle{\_dobystyle\_displaystyle\_textstyle\_scriptstyle\_scriptscriptstyle}
1059 \_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` Lua $\TeX$  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`

```

1073 \_def\_cramped{\_ifcase\_numexpr(\_mathstyle+1)/2\_relax\_or
1074 \_crampeddisplaystyle \_or \_crampedtextstyle \_or
1075 \_crampedscriptstyle \_or \_crampedscriptscriptstyle \_fi
1076 }
1077 \_public \_cramped ;

```

`\setmathstyle` saves current math style (including its cramped/normal subversion) and `\usemathstyle` restores the saved math style. These macros are based on the LuaTeX’s `\mathstyle` primitive, i.e. they don’t work in generalized fractions.

Usage: `\def\mathclap #1{\setmathstyle \hbox to0pt{\hss$\usemathstyle#1$\hss}}}`.

math-macros.opm

```
1087 \_newcount\_mstylenum
1088 \_def\_setmathstyle{\_mstylenum=\_mathstyle\_relax}
1089 \_def\_usemathstyle{\_ifcase\_mstylenum
1090   \_displaystyle\_or \_crampeddisplaystyle\_or \_textstyle\_or \_crampedtextstyle\_or
1091   \_scriptstyle\_or \_crampedscriptstyle\_or \_scriptscriptstyle\_or \_crampedscriptscriptstyle
1092   \_fi
1093 }
1094 \_public \_setmathstyle \_usemathstyle ;
```

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

```
1107 \_def\_mathbox#1{\_mathstyles{\_dirhbox{%
1108   \_ifnum\_stylenum<2 \_everymath{\_currstyle}%
1109   \_else \_ifx \_normalmath\_normalunimath
1110     \_ifnum\_stylenum=2 \_def\_textmff{ssty=1;}\_fi
1111     \_ifnum\_stylenum=3 \_def\_textmff{ssty=2;}\_def\_scriptmff{ssty=2;}\_fi
1112     \_typosize[\_ea\_ignorept\_the\_fontdimen6\_dobystyle]{\_scriptfont\_scriptscriptfont1}/%
1113     \_else \_typoscale[\_dobystyle]{\_}{700}{500}/%
1114     \_fi\_fi #1}}}%
1115 }
1116 \_public \_mathbox ;
```

`\dirhbox` and `\dirvbox` are equal to `\hbox` and `\vbox`, but they set the printing of their parameter according to the `\inboxdirection` register.

math-macros.opm

```
1123 \_def\_dirhbox{\_hbox bdir\_inboxdirection} \_def\_dirvbox{\_vbox bdir\_inboxdirection}
1124 \_public \_dirhbox \_dirvbox ;
```

`\swapeqno` swaps the meaning of `\eqno` with `\leqno` and `\eqnorignt` with `\eqnoleft`. You can declare it if you want to move all equation marks from right to left or from left to right in an already prepared document. Other use case is with r2l languages: If `\pardirection=1` then the `\eqno` and `\leqno` are swapped and if you don’t want it then use `\swapeqno`.

math-macros.opm

```
1135 \_def\_swapeqno{\_let\_tmp=\_eqno \_let\_eqno=\_leqno \_let\_leqno=\_tmp
1136   \_let\_tmp=\_eqnorignt \_let\_eqnorignt=\_eqnoleft \_let\_eqnoleft=\_tmp
1137   \_public \_eqno \_leqno ;
1138 }
1139 \_let\_orieqno=\_eqno % you can do \_ifx\_eqno\_orieqno to test if they are swapped
1140 \_public \_swapeqno ;
```

## 2.16 Unicode-math fonts

The `\loadmath <optional-factor> {<Unicode-math font>}` macro loads the given math font 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.

The `<optional-factor>` is scaling factor of loaded font with respect to the size of the text font. It can be used if the used text font and loaded math font have incompatible ex height. If missing then the scaling factor is 1.

`\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 *main math font* is loaded by `\loadmath` (typically indirectly using `\fontfam`) and you can load more *additional math fonts* by `\addUmathfont`:

```
\addUmathfont \famname {[<normal-font>]}{<ffeatures>} {[<bold-font>]}{<ffeatures>} {<factor>}
```

The `\famname` is a control sequence declared by `\addUmathfont` for later use. It gets math family number. The `<factor>` is decimal number for size corrections in view of the main math font. If it is empty then `<factor>=1`. If `<bold-font>` is empty, the “faked bold” derived from `<normal-font>` is used. Example:

```
\fontfam[1m] % does \lodmath{[latinmodern-math]}
\addUmathfont \xits {[XITSMath-Regular]}{} {[XITSMath-Bold]}{} {}
```

declares `latinmodern-math` as main math font (its bold variant is “faked bold”). The additional math font family `\xits` is declared in the example. It uses `XITSMath-Regular` for normal printing and `XITSMath-Bold` for bold printing.

All characters used in math formula are printed from main math font by default. But you can re-declare characters for printing from additional font by `\mathchars \famname {<list of sequences>}`. For example:

```
\mathchars \xits {\stareq \triangleq \veeeq \wedgeq}
```

sets the characters `\stareq`, `\triangleq`, `\veeeq`, `\wedgeq` from the `\xits` additional font. The `<list of sequences>` can include control sequences from the `unicode-table.tex`, but no math accents. These control sequences can be printed by `\input print-unimath.opm`.

The `\mathchars` macro keeps the class and slot of declared math objects and re-declares only family of them. It is applied to all control sequences given in the parameter. The relevant math codes are re-declared.

Use `\addto\selector{\fam\famname}` if you want to print whole math alphabet from an additional math font. For example `\addto\cal{\fam\xits}` declares all `\cal` characters from the `\xits` font loaded by `\addUmathfont`.

The `\mathcodes` macro provides comfortable settings of math codes of math objects. Its syntax is `\mathcodes <family> {<list-of-pairs>}`. Each pair in the `<list-of-pairs>` is `<class-number><character>` (separated by optional space) or `<class-number>{<list-of-characters>}`. The `<list-of-characters>` includes declared characters or `\Urange <from>-<to>` which is equal to the list of characters beginning `<from>` and ending `<to>`, for example `\Urange a-d` is equal to `abcd`. The characters can be given directly or by the math sequences like `\times`, `\doteq` too.

The `\mathcodes` macro declares mathcode of given characters internally by

```
\Umathcode `<character> = <class-number> <family> `<character>
```

The `\mathcodes` macro sets math codes of given Unicode characters. The relevant control sequence from `unicode-table.tex` changes its behavior too. For example, If you change math code of `×` then the `\times` control sequence will behave like new declared `×`.

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

unimath-macros.opm

```
3 \_codedecl \loadmath {Unicode Math fonts <2025-04-03>} % preloaded in format
```

`\loadmath <optional-factor> {<Unicode-math font>}` loads the given font. It does:

- define `\_unimathfont` as `<Unicode-math font>`,
- redefine `\normalmath` and `\boldmath` macros to their Unicode counterparts,
- save the `<optional-factor>` as scaling factor, see also `\_mfactor`,
- 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`.



```

20 \newif \ifmathloading \mathloadingtrue
21
22 \def\loadmath{\ifmathloading\mathloadingtrue}
23 \def\doloadmath{\ifmathloading\mathloadingtrue}
24
25 \def\loadmath#1{\loadmathA{#1}}
26 \def\loadmathA#1#2{%
27   \ifmathloading
28   \initunifonts
29   \isfont{#2}\iffalse
30   \opwarning{Math font "#2" not found, skipped...}%
31   \else
32   \sdef{mfactor:1}{#1}\def\unimathfont{#2}%
33   \let\normalmath = \normalunimath \let\boldmath = \boldunimath
34   \normalmath
35   \wterm {MATH-FONT: "#2" -- unicode math prepared.}%
36   \ifx\ncharraA\undefined \opinput {unimath-codes.opm}\fi
37   \mathloadingfalse
38   \fi\fi}
39
40 \public \loadmath \loadmath \loadmath ;

```

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

```

50 \def\loadboldmath#1#2\to #3{%
51   \isfont{#1}\iffalse
52   \opwarning{Bold-Math font "#1" not found, skipped...}%
53   \else \sdef{bmath:#1}{#3}%
54   \wterm {BOLD-MATH-FONT: "#1" with "#3".}%
55   \fi}
56
57 \public \loadboldmath ;

```

The Unicode version of the `\normalmath` and `\boldmath` macros are defined here as `\normalunimath` and `\boldunimath` macros. The base font `\unimathfont` is loaded to family 1 with `script=math` feature. It is generally used for all variables, digits and math symbols. The same font is loaded to family 2 with `script=latn`. It can be used for sequences of letters where we don't want to insert italic correction between them. The default `\rm` and `\it` macros run `\rmletters` and `\itletters` and they use family 2, so these selectors prohibit italic corrections between letters.

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 at family 1 with 8bit Math font at family 4. See definition of `\script` macro where `\fam4` is used. The final macro `\setunimathdimens` does similar work as `\setmathdimens`.

```

79 \def\normalunimath{%
80   \setmathfamily 0 \tenrm           % font for non-math objects in math mode
81   \loadumathfamily 1 {\unimathfont}{script=math;} % Base math font
82   \loadumathfamily 2 {\unimathfont}{script=latn;} % Base font, no italic corr. between letters
83   \loadmathfamily 4 rsfs           % script
84   \setunimathdimens
85 }%
86 \def\boldunimath{%
87   \setmathfamily 0 \tenbf           % font for non-math objects in math mode
88   \ifcsname _bmath:\unimathfont \endcsname
89   \loadumathfamily 1 {\cs{bmath:\unimathfont}}{script=math;} % Base real bold font
90   \loadumathfamily 2 {\cs{bmath:\unimathfont}}{script=latn;} % Base bold, no italcorr.
91   \else
92   \loadumathfamily 1 {\unimathfont}{script=math;embolden=1.7;} % Base faked bold
93   \loadumathfamily 2 {\unimathfont}{script=latn;embolden=1.7;} % Base faked, no italcor.
94   \fi
95   \loadmathfamily 4 rsfs           % script
96   \setunimathdimens
97 }%
98 \def\setunimathdimens{% PlainTeX sets these dimens for 10pt size only:

```

```

99 \delimitershortfall=0.5\fontdimen6\textfont1
100 \nulldelimiterspace=0.12\fontdimen6\textfont1
101 \setmathparam\Umathspaceafterscript \scriptspacefactor
102 \setbox0=\hbox{\everymath}\fam1\displaystyle{0\atop0}}%
103 \Umathfractiondelsize\displaystyle = \dimexpr(\ht0-\Umathaxis\displaystyle)*2\relax
104 \setbox0=\box\voidbox
105 }

```

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`  $\langle number \rangle$   $\{\langle font \rangle\}$   $\{\langle font features \rangle\}$  loads the given Unicode-math fonts in three sizes using single  $\langle font \rangle$  with different `mathsize=1,2,3` font features. The math font family is set with given  $\langle number \rangle$ . The  $\langle font features \rangle$  are added to the default `\_mfontfeatures` and to the size-dependent features `ssty=1` if script size is asked or `ssty=2` if scriptscriptsize is asked.

`\_mparams` can insert additional font features depending on the current `\_mfam`.

The `\_mfactor`  $\langle family \rangle \langle space \rangle$  sets scaling factor, see section 2.14 for more information.

The `\_textmff`, `\_scriptmff` and `\_sscriptmff` are font features for text, script and sscript sizes respectively. They are locally re-defined in `\mathbox` macro.

unimath-macros.opm

```

140 \def\_umathname#1#2{"#1:\_mfontfeatures#2"}
141 \def\_mfontfeatures{mode=base;}
142
143 \def\_loadumathfamily{\_afterassignment\_loadumathfamilyA \chardef\_mfam}
144 \def\_loadumathfamilyA #1#2 {\_mfactor
145   \font\_mF \umathname{#1}\_textmff \_mparams #2} at\_sizemtext \textfont \_mfam=\_mF
146   \font\_mF \umathname{#1}\_scriptmff \_mparams #2} at\_sizemtext \scriptfont \_mfam=\_mF
147   \font\_mF \umathname{#1}\_sscriptmff \_mparams #2} at\_sizemtext \scriptscriptfont \_mfam=\_mF
148 }
149 \def\_textmff {ssty=0;mathsize=1;}
150 \def\_scriptmff {ssty=1;mathsize=2;}
151 \def\_sscriptmff {ssty=2;mathsize=3;}
152 \def\_mparams{}

```

`\addUmathfont`  $\langle fam \rangle$   $\{[\langle normal-font \rangle]\}$   $\{\langle ffeatures \rangle\}$   $\{[\langle bold-font \rangle]\}$   $\{\langle ffeatures \rangle\}$   $\{\langle factor \rangle\}$  allocates new  $\langle fam \rangle$  using `\newfam` and adds loading this font to the `\normalmath` and `\boldmath` macros. Note that allocationos using `\newfam` starts from 43 because numbers 1–42 are reserved for direct usage without `\newfam`. We use `\aheadto` here because we want to read the main family 1 as last one (for definitive setting of math parameters).

unimath-macros.opm

```

164 \def\_addUmathfont #1#2#3#4#5#6{% #1: fam (will be set), #2#3: normal font, #4#5: bold font
165   \ifx\_ncharraA\_undefined \errmessage{basic Unicode math font must be loaded first}%
166   \else \isfont{#2}\iffalse \opwarning{font #2 is unavailable}%
167   \else
168     \newfam#1\relax
169     \sdef\_mfactor:\_the\_numexpr#1\relax}{#6}%
170     \global\_aheadto\_normalmath{\loadumathfamily #1{#2}\_addUmathdefault #3} }%
171     \ifx\_relax#4\relax
172       \global\_aheadto\_boldmath{\loadumathfamily #1{#2}\_addUmathdefault embolden=1.7;#3} }%
173     \else
174       \global\_aheadto\_boldmath{\loadumathfamily #1{#4}\_addUmathdefault #5} }%
175     \fi
176     \normalmath
177     \wterm{add-MATH-FONT: #1=\_the#1, "#2", \ifx"#4"\_else bold: "#4"\_fi}%
178     \fi \fi

```

```

179 }
180 \def\addUmathdefault {script=math;} % default font features when \addUmathfont is used

```

The math characters can be given directly (by their Unicode) or by a macro like `\doteq`, `\times`, etc. These macros simply expand to the math character with its Unicode. And this math character has its `\Umathcode` given by `\langle class \rangle`, `\langle family \rangle`, `\langle slot-number \rangle`. Sometimes, we may want to get these quantities from the given Unicode math character by our macros. It is possible by `\themathcodeclass\langle math-char \rangle`, `\themathcodefam\langle math-char \rangle` and `\themathcodechar\langle math-char \rangle` macros. The parameter `\langle math-char \rangle` is a math character or it is a macro like `\doteq`, `\times`. Moreover, `\thedelcodefam\langle math-char \rangle` and `\thedelcodechar\langle math-char \rangle` return delcode quantities of given math character. All these commands use the common Lua code defined in the `\getmathcode\langle code \rangle\{\langle math or del \rangle\}\langle character number \rangle` macro.

unimath-macros.opm

```

197 \def\getmathcode#1#2{\directlua{tex.print(tex.get#2code(token.scan_int())[#1])}}
198 \def\themathcodeclass #1{\getmathcode 1{math}\_ea`#1 }
199 \def\themathcodefam #1{\getmathcode 2{math}\_ea`#1 }
200 \def\themathcodechar #1{\getmathcode 3{math}\_ea`#1 }
201 \def\thedelcodefam #1{\getmathcode 1{del}\_ea`#1 }
202 \def\thedelcodechar #1{\getmathcode 2{del}\_ea`#1 }
203
204 \public \themathcodeclass \themathcodefam \themathcodechar \thedelcodefam \thedelcodechar ;

```

`\mathchars \langle fam \rangle \{\langle list of sequences \rangle\}` saves `\langle fam \rangle` to `\_mafam` and runs for each sequence from the `\langle list of sequences \rangle` the relevant code settings using `\Umathcode` primitive. In case of `\int`-like operators the `\langle math class \rangle=8` and we only re-declare `\_int:\langle int-character \rangle` as an operator with the new `\_mafam`. Note that the used primitives have the syntax:

```

\Umathchardef \langle sequence \rangle = \langle math class \rangle \langle math family \rangle \langle slot number \rangle
\Umathcode \langle code \rangle = \langle math class \rangle \langle math family \rangle \langle slot number \rangle
\Udelcode \langle code \rangle = \langle math family \rangle \langle slot number \rangle

```

unimath-macros.opm

```

220 \def\mathchars {\_afterassignment\mathcharsA \chardef\_mafam=}
221 \def\mathcharsA #1{\_foreach #1\_do{%
222   \chardef\_tmp=\themathcodeclass##1\_relax
223   \ifnum\_tmp=8 % \int, \iint, \oint, etc.
224     \ea\Umathchardef \csname\_int:##1\_endcsname =1 \_mafam \_ea`##1
225   \else
226     \Umathcode \_ea`##1=\_tmp \_mafam \themathcodechar##1
227   \_fi
228 }}

```

`\mathcodes \langle fam \rangle \{\langle list of pairs \rangle\}` sets mathcodes of given characters with explicit `\langle class \rangle`s. Each pair can be `\langle class \rangle\{\langle list of chars \rangle\}` and `\langle list of chars \rangle` can include `\Urange \langle from \rangle-\langle to \rangle`. This is reason why we apply `\expanded` to the `\langle list of chars \rangle` before reading it by `\foreach`: the `\Urange` is expandable and expands to the relevant list of characters.

unimath-macros.opm

```

239 \def\mathcodes{\_afterassignment\mathcodesA\chardef\_mafam=}
240 \def\mathcodesA#1{%
241   \_foreach #1\_do ##1##2{%
242     \ea\_foreach\_expanded{##2}\_do{\_Umathcode `####1=##1\_mafam \_ea`####1}%
243   }%
244 }
245 \def\Urange #1-#2{\_fornum \_ea`#1..\_ea`#2\_do{\_Uchar##1 }}
246
247 \public \addUmathfont \mathchars \mathcodes \Urange ;

```

## 2.16.2 Macros and codes set when `\loadmath` is processed firstly

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 \_ncharmA {Uni math codes <2026-05-08>} % preloaded on demand by \loadmath

```

Unicode math font includes all typical math alphabets together, user needs no load more  $\TeX$  math families. These math alphabets are encoded by different parts of Unicode table. We need auxiliary macros for setting mathcodes by selected math alphabet.

`\_umathrange`  $\langle from \rangle - \langle to \rangle \langle class \rangle \langle family \rangle \langle first \rangle$  sets `\Umathcodes` of the characters in the interval  $\langle from \rangle - \langle to \rangle$  to  $\langle first \rangle$ ,  $\langle first \rangle + 1$ ,  $\langle first \rangle + 2$  etc., but `\_umathcharholes` are skipped (`\_umathcharholes` are parts of the Unicode table not designed for math alphabets, 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 characters like A–Z. Note that the `\_umathrange` sets the `\_classfam` macro as  $\langle class \rangle \langle family \rangle$  for later use.

unimath-codes.opm

```

25 \_newcount\_umathnumA \_newcount\_umathnumB
26
27 \_def\_umathholes[#1]#2{\_ifx~#1~\_else
28   \_ea\_chardef \_csname \_mh:#1\_endcsname=#2 \_ea\_umathholes\_fi}
29 \_umathholes % holes in math alphabets:
30 [119893]{ "210E}[119965]{ "212C}[119968]{ "2130}[119969]{ "2131}%
31 [119971]{ "210B}[119972]{ "2110}[119975]{ "2112}[119976]{ "2133}[119981]{ "211B}%
32 [119994]{ "212F}[119996]{ "210A}[120004]{ "2134}%
33 [120070]{ "212D}[120075]{ "210C}[120076]{ "2111}[120085]{ "211C}[120093]{ "2128}%
34 [120122]{ "2102}[120127]{ "210D}[120133]{ "2115}[120135]{ "2119}%
35 [120136]{ "211A}[120137]{ "211D}[120145]{ "2124"}{}%
36 \_let\_umathholes=\_undefined
37
38 \_def\_umathrange#1#2#3#4{\_umathnumB=#4\_def\_classfam{#2 #3 }\_umathrangeA#1}
39 \_def\_umathrangeA#1-#2{\_umathnumA=#1\_relax
40   \_loop
41     \_Umathcode \_umathnumA = \_classfam \_trycs{\_mh:\_the\_umathnumB}\_umathnumB
42     \_ifnum\_umathnumA<#2\_relax
43       \_advance\_umathnumA by1 \_advance\_umathnumB by1
44     \_repeat
45 }

```

A few math characters have very specific Unicode and must be handled individually. We can run `\_umathrangespec`  $\langle list\ of\ characters \rangle \_relax$  just after `\_umathrange`. The `\_umathnumB` must be set to the first destination code. The `\_umathrangespec` applies to each character from the  $\langle list\ of\ characters \rangle$  this: `\Umathcode`  $\langle char-code \rangle = \_classfam \_umathnumB$  and increments `\_umathnumB`. If `\_umathnumB=0` then it applies `\Umathcode`  $\langle char-code \rangle = \_classfam \langle char-code \rangle$ . The `\_classfam` and `\_umathnumB` were typically set by previous call of the `\_umathrange` macro.

unimath-codes.opm

```

60 \_def\_umathrangespec#1{\_ifx#1\_relax \_else
61   \_Umathcode `#1=\_classfam \_ifnum\_umathnumB=0 `#1 \_else \_umathnumB\_fi
62   \_unless\_ifnum\_umathnumB=0 \_advance\_umathnumB by1 \_fi
63   \_ea\_umathrangespec \_fi
64 }

```

The math alphabets are set by `\_rmvariables`, `\_rmletters`, `\_bfvariables`, `\_bfletters`, `\_itvariables`, `\_itletters`, `\_bivvariables`, `\_bileters`, `\_calvariables`, `\_bcalvariables`, `\_frakvariables`, `\_bfrakvariables`, `\_bbvariables`, `\_sansvariables`, `\_bsansvariables`, `\_isansvariables`, `\_bisansvariables`, `\_ttvariables`, `\_itgreek`, `\_rmgreek`, `\_bfgreek`, `\_bigreek`, `\_bsansgreek`, `\_bisansgreek`, `\_itGreek`, `\_rmGreek`, `\_bfGreek`, `\_biGreek`, `\_bsansGreek`, `\_bisansGreek`, `\_rmdigits`, `\_bfdigits`, `\_bbdigits`, `\_sansdigits`, `\_bsansdigits`, `\_ttdigits`.

They are declared using the `\_umathrange`  $\langle range \rangle \langle class \rangle \langle family \rangle \langle starting-code \rangle$  macro.

unimath-codes.opm

```

82 \_chardef\_ncharmA="A \_chardef\_ncharma="a
83 \_chardef\_ncharbA="1D400 \_chardef\_ncharbfa="1D41A
84 \_chardef\_nchariA="1D434 \_chardef\_ncharita="1D44E
85 \_chardef\_ncharbiA="1D468 \_chardef\_ncharbia="1D482
86 \_chardef\_ncharcA="1D49C \_chardef\_ncharcla="1D4B6
87 \_chardef\_ncharbcA="1D4D0 \_chardef\_ncharbca="1D4EA
88 \_chardef\_ncharfrA="1D504 \_chardef\_ncharfra="1D51E
89 \_chardef\_ncharbrA="1D56C \_chardef\_ncharbra="1D586
90 \_chardef\_ncharbbA="1D538 \_chardef\_ncharbba="1D552
91 \_chardef\_ncharsnA="1D5A0 \_chardef\_ncharsna="1D5BA
92 \_chardef\_ncharbsA="1D5D4 \_chardef\_ncharbsa="1D5EE
93 \_chardef\_ncharsiA="1D608 \_chardef\_ncharsia="1D622
94 \_chardef\_ncharsx A="1D63C \_chardef\_ncharsxa="1D656
95 \_chardef\_ncharttA="1D670 \_chardef\_nchartta="1D68A
96

```

```

97 \protected\def\rmvariables {\umathrange{A-Z}71\_ncharmA \umathrange{a-z}71\_ncharmA}
98 \protected\def\rmletters {\umathrange{A-Z}72\_ncharmA \umathrange{a-z}72\_ncharmA}
99 \protected\def\bfvariables {\umathrange{A-Z}71\_ncharbFA \umathrange{a-z}71\_ncharbFA}
100 \protected\def\bfletters {\umathrange{A-Z}72\_ncharbFA \umathrange{a-z}72\_ncharbFA}
101 \protected\def\itvariables {\umathrange{A-Z}71\_ncharitA \umathrange{a-z}71\_ncharitA}
102 \protected\def\itletters {\umathrange{A-Z}72\_ncharitA \umathrange{a-z}72\_ncharitA}
103 \protected\def\bivvariables {\umathrange{A-Z}71\_ncharbiA \umathrange{a-z}71\_ncharbiA}
104 \protected\def\biletters {\umathrange{A-Z}72\_ncharbiA \umathrange{a-z}72\_ncharbiA}
105 \protected\def\calvariables {\umathrange{A-Z}71\_ncharclA \umathrange{a-z}71\_ncharclA}
106 \protected\def\bcvariables {\umathrange{A-Z}71\_ncharbcA \umathrange{a-z}71\_ncharbcA}
107 \protected\def\frakvariables {\umathrange{A-Z}71\_ncharfrA \umathrange{a-z}71\_ncharfrA}
108 \protected\def\bfrakvariables {\umathrange{A-Z}71\_ncharbrA \umathrange{a-z}71\_ncharbrA}
109 \protected\def\bbvariables {\umathrange{A-Z}71\_ncharbbA \umathrange{a-z}71\_ncharbbA}
110 \protected\def\sansvariables {\umathrange{A-Z}71\_ncharsnA \umathrange{a-z}71\_ncharsnA}
111 \protected\def\bsansvariables {\umathrange{A-Z}71\_ncharsbA \umathrange{a-z}71\_ncharsbA}
112 \protected\def\isansvariables {\umathrange{A-Z}71\_ncharsiA \umathrange{a-z}71\_ncharsiA}
113 \protected\def\bisansvariables {\umathrange{A-Z}71\_ncharsxA \umathrange{a-z}71\_ncharsxA}
114 \protected\def\ttvariables {\umathrange{A-Z}71\_ncharttA \umathrange{a-z}71\_ncharttA}
115
116 \chardef\greekrmA="0391 \chardef\greekrma="03B1
117 \chardef\greekbfa="1D6A8 \chardef\greekbfa="1D6C2
118 \chardef\greekita="1D6E2 \chardef\greekita="1D6FC
119 \chardef\greekbia="1D71C \chardef\greekbia="1D736
120 \chardef\greekna="1D756 \chardef\greekna="1D770
121 \chardef\greekia="1D790 \chardef\greekia="1D7AA
122
123 \protected\def\itgreek {\umathrange{greek}71\_greekita}
124 \protected\def\rmgreek {\umathrange{greek}71\_greekrma}
125 \protected\def\bfgreek {\umathrange{greek}71\_greekbfa}
126 \protected\def\bigreek {\umathrange{greek}71\_greekbia}
127 \protected\def\bsansgreek {\umathrange{greek}71\_greekna}
128 \protected\def\bisansgreek {\umathrange{greek}71\_greekia}
129 \protected\def\itGreek {\umathrange{GREEK}71\_greekita}
130 \protected\def\rmGreek {\umathrange{GREEK}71\_greekrma}
131 \protected\def\bfGreek {\umathrange{GREEK}71\_greekbfa}
132 \protected\def\biGreek {\umathrange{GREEK}71\_greekbia}
133 \protected\def\bsansGreek {\umathrange{GREEK}71\_greekna}
134 \protected\def\bisansGreek {\umathrange{GREEK}71\_greekia}
135
136 \chardef\digitrm0="0
137 \chardef\digitbf0="1D7CE
138 \chardef\digitbb0="1D7D8
139 \chardef\digitsn0="1D7E2
140 \chardef\digitbs0="1D7EC
141 \chardef\digittt0="1D7F6
142
143 \protected\def\rmdigits {\umathrange{0-9}71\_digitrm0}
144 \protected\def\bfdigits {\umathrange{0-9}71\_digitbf0}
145 \protected\def\bbdigits {\umathrange{0-9}71\_digitbb0}
146 \protected\def\sansdigits {\umathrange{0-9}71\_digitsn0}
147 \protected\def\bsansdigits {\umathrange{0-9}71\_digitbs0}
148 \protected\def\tdigits {\umathrange{0-9}71\_digittt0}

```

The control sequences for `\alpha`, `\beta`, etc. are redefined here. The `\alpha` will expand 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. These sequences are declared by `\_greekdef{list of sequences}\relax`.

```

158 \def\_greekdef#1{\_ifx#1\_relax
159   \else
160     \edef#1{\_Uchar\_umathnumB}%
161     \advance\_umathnumB by 1
162     \ea\_greekdef \_fi
163 }
164 \umathnumB="0391
165 \greekdef \Alpha \Beta \Gamma \Delta \Epsilon \Zeta \Eta \Theta \Iota \Kappa
166   \Lambda \Mu \Nu \Xi \Omicron \Pi \Rho \varTheta \Sigma \Tau \Upsilon \Phi
167   \Chi \Psi \Omega \relax
168
169 \umathnumB="03B1

```

unimath-codes.opm

```

170 \greekdef \alpha \beta \gamma \delta \varepsilon \zeta \eta \theta \iota \kappa \lambda
171 \lambda \mu \nu \xi \omicron \pi \rho \varsigma \sigma \tau \upsilon \pi
172 \varphi \chi \psi \omega \relax

```

The `\umathrangeGREEK`(*class*)(*family*)(*first*) and `\umathrangegreek`(*class*)(*family*)(*first*) macros for setting math codes of Greek characters are defined here. They use `\umathrange` for general codes but the exceptions must be handled by the `\umathrangespec` macro. The exceptions are seven Greek characters:  $\epsilon, \vartheta, \kappa, \phi, \varrho, \varpi, \nabla$ . The first six of these characters should behave as lowercase Greek letters and the last one `\nabla` is uppercase Greek letter.

unimath-codes.opm

```

186 \def\epsilon{~~~~~03f5} \def\vartheta{~~~~~03d1} \def\varkappa{~~~~~03f0}
187 \def\phi{~~~~~03d5} \def\varrho{~~~~~03f1} \def\varpi{~~~~~03d6}
188 \def \nabla{~~~~~2207}
189
190 \def\umathrangeGREEK#1#2#3{\umathrange{~~~~~0391-~~~~~03a9}#1#2#3% \Alpha-\Omega
191 \resetnabla % you can do \let\resetnabla=\relax if you don't want to change \nabla shape
192 }
193 \def\resetnabla {%
194 \ifnum\umathnumB<950 \umathnumB=0 \else \advance\umathnumB by1 \fi
195 \umathrangespec ~~~~~2207\relax % \nabla
196 }
197 \def\umathrangegreek#1#2#3{%
198 \umathrange{~~~~~03b1-~~~~~03c9}#1#2#3% \alpha-\omega
199 \ifnum#3=\greekrma \umathnumB=0 \else \advance\umathnumB by2 \fi
200 \umathrangespec ~~~~~03f5~~~~~03d1~~~~~03f0~~~~~03d5~~~~~03f1~~~~~03d6\relax % \epsilon-\varpi
201 }

```

The math alphabets `\cal`, `\bbchar`, `\frac`, `\script`, `\misans`, `\mbisans` are re-defined here. The `\marm`, `\mabf`, `\mait`, `\mabi`, `\matt` used in `\rm`, `\bf`, `\it`, `\bi` are re-defined too.

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\mabf {\inmath{\bfvariables\bfgreek\bfGreek\bfdigits}}
\protected\def\mabi {\inmath{\bfvariables\bigreek\bfGreek\bfdigits}}

```

`\inmath` {*cmds*} applies *cmds* only in math mode.

unimath-codes.opm

```

217 \protected\def\inmath#1{\relax \ifmmode#1\fi} % to keep off \loop processing in text mode
218
219 % You can redefine these macros to follow your wishes.
220 % For example, you need upright lowercase greek letters, you don't need
221 % \bf and \bi behave as sans serif in math, ...
222
223 \protected\def\marm {\inmath{\rmletters \rmdigits}}
224 \protected\def\mait {\inmath{\itletters \itGreek}}
225 \protected\def\mabf {\inmath{\bsansvariables \bsansgreek \bsansGreek \bsansdigits}}
226 \protected\def\mabi {\inmath{\bisansvariables \bisansgreek \bisansGreek \bisansdigits}}
227 \protected\def\matt {\inmath{\ttvariables \ttdigits}}
228 \protected\def\bbchar {\bbvariables \bbdigits}
229 \protected\def\cal {\calvariables}
230 \protected\def\frac {\fracvariables}
231 \protected\def\misans {\isansvariables \sansdigits}
232 \protected\def\mbisans {\bisansvariables \bisansgreek \bisansGreek \bsansdigits}
233 \protected\def\script {\rmvariables \fam4}
234 \protected\def\mit {\itvariables \rmdigits \itgreek \rmGreek}
235
236 \public \bbchar \cal \frac \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:

mathclass.opm

```

70 002E;P
71 002F;B
72 0030..0039;N
73 003A;P
74 003B;P
75 003C;R
76 003D;R
77 003E;R

```



```

78 003F;P
79 0040;N
80 0041..005A;A
81 005B;O
82 005C;B
83 005D;C
84 005E;N
85 005F;N

```

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

unimath-codes.opm

```

246 \begingroup % \input mathclass.opm (which is a copy of MathClass.txt):
247 \long\def\p#1;#2 {\ifx^#2\else
248 \edef\tmp{\csname _c:#2\endcsname}\if\relax\tmp\else \pA#1...\end#2\fi
249 \ea\p \fi }
250 \def\pA#1..#2..#3\end#4{%
251 \ifx\relax#2\relax \pset{#1}{#4}\else \fornum "#1..#2\do{\pset{##1}{#4}}\fi
252 }
253 \sdef{c:L}{1}\sdef{c:B}{2}\sdef{c:V}{2}\sdef{c:R}{3}\sdef{c:N}{0}\sdef{c:U}{0}
254 \sdef{c:F}{0}\sdef{c:O}{4}\sdef{c:C}{5}\sdef{c:P}{6}\sdef{c:A}{7}
255 \def\pset#1#2{\Umathcode#1=\tmp\space 1 #1\relax
256 \if#20\Udelcode#1=1 #1\relax\fi
257 \if#2C\Udelcode#1=1 #1\relax\fi
258 \if#2F\Udelcode#1=1 #1\relax\fi
259 }
260 \catcode\#14 \everyeof={;} } \def\par{}
261 \globaldefs=1 \ea \p \input mathclass.opm
262 \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:

unimath-table.opm

```

36 \UnicodeMathSymbol{"000B1}{\pm }{\mathbin}{plus-or-minus sign}%
37 \UnicodeMathSymbol{"000B6}{\mathparagraph }{\mathord}{paragraph symbol}%
38 \UnicodeMathSymbol{"000B7}{\cdotp }{\mathbin}{/centerdot b: middle dot}%
39 \UnicodeMathSymbol{"000D7}{\times }{\mathbin}{multiply sign}%
40 \UnicodeMathSymbol{"000F0}{\matheth }{\mathalpha}{eth}%
41 \UnicodeMathSymbol{"000F7}{\div }{\mathbin}{divide sign}%

```

We have to read this information and set given control sequences as macros which expand to the given Unicode character. This solution enables to use such control sequences in PDF outlines where they expand to the appropriate Unicode character. We don't use `\mathchardef`, we set the mathcodes (class, family, slot) only at single place: for Unicode math characters. For example for we define `\times`:

```
\def\times{^~d7} \Umathcode "D7 = 2 1 "D7
```

Because math codes of Greek upright letters vary depending on `\_itgreek`, `\_bfgreek`, etc. macros, we need to keep the access directly to these characters. We define `\mupalpha`, `\mupbeta`, ..., `\mupomega` macros as a code from PUA (Private Use Area) of Unicode table and set mathcode of these codes to the real upright alpha, beta, ..., omega.

unimath-codes.opm

```

286 \begingroup % \input unimath-table.opm (it is a copy of unicode-math-table.tex):
287 \umathnumB="F800 % pointer to the Private User Area
288 \def\UnicodeMathSymbol #1#2#3#4{%
289 \edef#2{\_Uchar #1}% control sequence is a macro which expands to the Unicode character
290 \ifnum#1=\_Umathcodenum#1 \_Umathcode#1=0 1 #1 \fi % it isn't set by mathclass.opm
291 \ifx#3\_mathaccent \_protected\def#2{\_Umathaccent fixed 7 1 #1 }\fi
292 \ifnum#1>"390 \ifnum#1<"3F6
293 \edef#2{\_Uchar\_umathnumB}% \mupAlpha, \mupBeta, \mupalpha, \mupbeta, ...
294 \_Umathcode\_umathnumB=0 1 #1
295 \advance\_umathnumB by1
296 \fi\fi % \muGreek, \mugreek symbols
297 }
298 \def\mathfence{F}%
299 \globaldefs=1 \input unimath-table.opm
300 \endgroup

```

The macro `\int` expands to an `\langle int-character \rangle`. We save the `\mathcode` of the `\langle int-character \rangle` to `\_int:\langle int-character \rangle` using `\Umathchardef` and declare `\langle int-character \rangle` as math-active and define it

as `\_int:⟨int-character⟩ \_nolimits`. Moreover, we define `\intop` as `\_int:⟨int-character⟩` (it is the integral with limits like in plain TeX). We do this with other int-like operators listed below too.

unimath-codes.opm

```

311 \def\_intwithnolimits#1{\_ifx#1\_relax \_else
312 \_ea\_Umathcharnumdef\_csname\_int:#1\_endcsname=\_Umathcodenum\_ea`#1 %
313 \_ea\_def \_csname\_csstring#1op\_ea\_endcsname\_ea{\_csname\_int:#1\_endcsname}%
314 \_bgroup \_lccode`~=\_ea`#1 \_lowercase{\_egroup
315 \_ea\_def\_ea-\_ea{\_csname\_int:#1\_endcsname\_nolimits}\_mathcode`~="8000 }%
316 \_ea \_intwithnolimits \_fi
317 }
318 \_intwithnolimits \_int \_iint \_iiint \_oint \_oiint \_oiiint
319 \_intclockwise \_varointclockwise \_ointctrlockwise \_sumint \_iiint \_intbar \_intBar \_fint
320 \_pointint \_sqint \_intlarhk \_intx \_intcap \_intcup \_upint \_lowint \_relax

```

Many special characters must be declared with care...

unimath-codes.opm

```

326 \_global\_Udelcode`<=1 "027E8 % these characters have different meaning
327 \_global\_Udelcode`>=1 "027E9 % as normal and as delimiter
328
329 \_mit % default math alphabets setting
330
331 % hyphen character is transformed to minus:
332 \_Umathcode `~ = 2 1 "2212
333
334 % mathclass defines : as Punct, plain.tex as Rel, we keep mathclass,
335 % i.e. there is difference from plain.tex, you can use $f:A\to B$.
336
337 % mathclas defines ! as Ord, plain.tex as Close
338 \_Umathcode `! = 5 1 `! % keep plain.tex declaration
339 % mathclas defines ? as Punct, plain.tex as Close
340 \_Umathcode `? = 5 1 `? % keep plain.tex declaration
341
342 \_Umathcode `* = 2 1 "02217 % equivalent to \ast, like in plain TeX
343
344 \_Umathcode "03A2 = 7 1 "03F4 % \varTheta
345
346 \_Umathcode `© = 0 1 `© % usage $\copyright$ can be seen in old documents
347
348 \_protected\_def \_sqrt {\_Uradical 1 "0221A }
349 \_protected\_def \_cuberoot {\_Uradical 1 "0221B }
350 \_protected\_def \_fourthroot {\_Uradical 1 "0221C }
351
352 \_public \_sqrt \_cuberoot \_fourthroot ;
353
354 \_protected\_def \_overbrace #1{\_mathop {\_Umathaccent 7 1 "023DE{#1}}\_limits}
355 \_protected\_def \_underbrace #1{\_mathop {\_Umathaccent bottom 7 1 "023DF{#1}}\_limits}
356 \_protected\_def \_overparen #1{\_mathop {\_Umathaccent 7 1 "023DC{#1}}\_limits}
357 \_protected\_def \_underparen #1{\_mathop {\_Umathaccent bottom 7 1 "023DD{#1}}\_limits}
358 \_protected\_def \_overbracket #1{\_mathop {\_Umathaccent 7 1 "023B4{#1}}\_limits}
359 \_protected\_def \_underbracket #1{\_mathop {\_Umathaccent bottom 7 1 "023B5{#1}}\_limits}
360
361 \_public \_overbrace \_underbrace \_overparen \_underparen \_overbracket \_underbracket ;
362
363 \_protected\_def \_widehat {\_Umathaccent 7 1 "00302 }
364 \_protected\_def \_widetilde {\_Umathaccent 7 1 "00303 }
365 \_protected\_def \_overleftharpoon {\_Umathaccent 7 1 "020D0 }
366 \_protected\_def \_overrightharpoon {\_Umathaccent 7 1 "020D1 }
367 \_protected\_def \_overleftarrow {\_Umathaccent 7 1 "020D6 }
368 \_protected\_def \_overrightarrow {\_Umathaccent 7 1 "020D7 }
369 \_protected\_def \_overleftrightharpoon {\_Umathaccent 7 1 "020E1 }
370
371 \_protected\_def \_wideoverbar {\_Umathaccent 7 1 "00305 }
372 \_protected\_def \_widebreve {\_Umathaccent 7 1 "00306 }
373 \_protected\_def \_widecheck {\_Umathaccent 7 1 "0030C }
374 \_protected\_def \_wideutilde #1{\_Umathaccent bottom 7 1 "00330{\_kern\_zo #1}}
375 \_protected\_def \_mathunderbar #1{\_Umathaccent bottom 7 1 "00332{\_kern\_zo #1}}
376 \_protected\_def \_underleftrightharpoon #1{\_Umathaccent bottom 7 1 "0034D{\_kern\_zo #1}}
377 \_protected\_def \_widebridgeabove {\_Umathaccent 7 1 "020E9 }
378 \_protected\_def \_underrightharpoondown #1{\_Umathaccent bottom 7 1 "020EC{\_kern\_zo #1}}

```

```

379 \protected\def \underleftharpoondown #1{\_Umathaccent bottom 7 1 "020ED{\\_kern\_zo #1}}
380 \protected\def \underleftarrow #1{\_Umathaccent bottom 7 1 "020EE{\\_kern\_zo #1}}
381 \protected\def \underrightarrow #1{\_Umathaccent bottom 7 1 "020EF{\\_kern\_zo #1}}
382
383 \mathchardef\ldotp="612E
384 \let\|=\Vert
385 \mathcode\_"="8000
386
387 \global\_Umathcode "22EF = 0 1 "22EF % mathclass says that it is Rel
388 \global\_Umathcode "002E = 0 1 "002E % mathclass says that dot is Punct
389
390 \global\_Umathcode `/ = 0 1 `/ % mathclass says that / is Bin, Plain TeX says that it is Ord.
391
392 % compressed dots in S and SS styles (usable in \matrix when it is in T, S and SS style)
393 \protected\def \vdots {\_relax \_ifnum \_mathstyle>3 \_unicodevdots \_else \_vdots \_fi}
394 \protected\def \ddots {\_relax \_ifnum \_mathstyle>3 \_unicodeddots \_else \_ddots \_fi}
395 \protected\def \adots {\_relax \_ifnum \_mathstyle>3 \_unicodeadots \_else \_adots \_fi}
396
397 % Unicode superscripts (^) and subscripts as simple macros with \mathcode"8000
398 \bgroup
399 \def\_tmp#1#2{\_global\_mathcode#1="8000 \_lccode\_"#1 \_lowercase{\_gdef~}{#2}}
400 \_fornum 0..1 \_do {\_tmp{"207#1}{#1}}
401 \_tmp{"B2}{#2}\_tmp{"B3}{#3}
402 \_fornum 4..9 \_do {\_tmp{"207#1}{#1}}
403 \_fornum 0..9 \_do {\_tmp{"208#1}{#1}}
404 \egroup

```

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

unimath-codes.opm

```

411 \let \setminus=\smallsetminus
412 \let \diamond=\smwhtdiamond
413 \let \colon=\mathcolon
414 \let \bullet=\smbkcircle
415 \let \circ=\vysmwhtcircle
416 \let \bigcirc=\mdlgwhtcircle
417 \let \to=\rightarrow
418 \let \le=\leq
419 \let \ge=\geq
420 \let \neq=\ne
421 \protected\def \triangle {\mathord{\bigtriangleup}}
422 \let \emptyset=\varnothing
423 \let \hbar=\hslash
424 \let \land=\wedge
425 \let \lor=\vee
426 \let \owns=\ni
427 \let \gets=\leftarrow
428 \let \mathring=\ocirc
429 \let \lnot=\neg
430 \let \longdivisionsign=\longdivision
431 \let \backepsilon=\upbackepsilon
432 \let \eth=\matheth
433 \let \dbkarow=\dbkarrow
434 \let \drbkarow=\drbkarrow
435 \let \hksearrow=\hksearrow
436 \let \hkswarrow=\hkswarrow
437 \let \square=\mdlgwhtsquare
438 \let \blacksquare=\mdlgblksquare
439
440 \let \upalpha=\mupalpha
441 \let \upbeta=\mupbeta
442 \let \upgamma=\mupgamma
443 \let \updelta=\mupdelta
444 \let \upepsilon=\mupvarepsilon
445 \let \upvarepsilon=\mupvarepsilon
446 \let \upzeta=\mupzeta
447 \let \upeta=\mupeta
448 \let \uptheta=\muptheta
449 \let \upiota=\mupiota

```

```

450 \_let \upkappa=\mupkappa
451 \_let \uplambda=\muplambda
452 \_let \upmu=\mupmu
453 \_let \upnu=\mupnu
454 \_let \upxi=\mupxi
455 \_let \upomicron=\mupomicron
456 \_let \uppi=\muppi
457 \_let \uprho=\muprho
458 \_let \upvarrho=\mupvarrho
459 \_let \upvarsigma=\mupvarsigma
460 \_let \upsigma=\mupsigma
461 \_let \uptau=\mu\tau
462 \_let \upupsilon=\mupupsilon
463 \_let \upvarphi=\mupvarphi
464 \_let \upchi=\mupchi
465 \_let \uppsi=\muppsi
466 \_let \upomega=\mupomega
467 \_let \upvartheta=\mupvartheta
468 \_let \upphi=\mupphi
469 \_let \upvarpi=\mupvarpi
470 \_let \varTheta=\mupvarTheta
471 \_let \vardelta=\delta

```

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

```

479 \_protected\_def\_not#1{%
480 \_trycs{\not!\_csstring#1}{\_mathrel{\_Umathaccent overlay fixed 7 1 "00338 {#1}}}}
481
482 \_def\_negationof #1#2f\_ea\_let \_csname \_not!\_csstring#1\_endcsname =#2}
483
484 \_negationof = \neq
485 \_negationof < \nless
486 \_negationof > \ngtr
487 \_negationof \gets \nleftarrow
488 \_negationof \simeq \nsime
489 \_negationof \equal \ne
490 \_negationof \le \nleq
491 \_negationof \ge \ngeq
492 \_negationof \greater \ngtr
493 \_negationof \forksnot \forks
494 \_negationof \in \notin
495 \_negationof \mid \nmid
496 \_negationof \cong \ncong
497 \_negationof \leftarrow \nleftarrow
498 \_negationof \rightarrow \nrightarrow
499 \_negationof \leftrightarrow \nleftrightarrow
500 \_negationof \Leftrightarrow \nLeftrightarrow
501 \_negationof \Leftrightarrow \nLeftrightarrow
502 \_negationof \Rrightarrow \nRrightarrow
503 \_negationof \exists \nexists
504 \_negationof \ni \nni
505 \_negationof \parallel \nparallel
506 \_negationof \sim \nsim
507 \_negationof \approx \napprox
508 \_negationof \equiv \nequiv
509 \_negationof \asymp \nasympt
510 \_negationof \lesssim \nlesssim
511 \_negationof \ngtrsim \ngtrsim
512 \_negationof \lessgtr \nlessgtr
513 \_negationof \gtrless \ngtrless
514 \_negationof \prec \nprec
515 \_negationof \succ \nsucc
516 \_negationof \subset \nsubset
517 \_negationof \supset \nsupset
518 \_negationof \subseteq \nsubseteq
519 \_negationof \supseteq \nsupseteq
520 \_negationof \vdash \nvdash
521 \_negationof \Vdash \nVdash

```

```

522 \negationof \Vdash \nVdash
523 \negationof \VDash \nVDash
524 \negationof \preccurlyeq \npreccurlyeq
525 \negationof \succcurlyeq \nsucccurlyeq
526 \negationof \sqsubseteq \nsqsubseteq
527 \negationof \sqsupseteq \nsqsupseteq
528 \negationof \vartriangleleft \nvartriangleleft
529 \negationof \vartriangleright \nvartriangleright
530 \negationof \trianglelefteq \ntrianglelefteq
531 \negationof \trianglerighteq \ntrianglerighteq
532 \negationof \vinfty \nvinfty
533
534 \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

```

542 \private
543 \ldotp \cdotp \bullet \triangleleft \triangleright \mapstochar \rightarrow
544 \prime \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
545 \rbrace \lbrace \Relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
546 \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. More examples are in the [OpTeX tricks](#) and in the [math.opm](#) package.

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 TeX 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 \codedec1 \undefined {Printing Unicode-math table \string<2020-06-08>}
4
5 \ifx\nccharmA\undefined \opwarning{No Unicode math font loaded, printing ignored}
6 \endinput \fi
7
8 \begingroup
9 \def\UnicodeMathSymbol#1#2#3#4{%
10 \ifnum#1>"10000 \endinput \else \printmathsymbol{#1}{#2}{#3}{#4}\fi
11 }
12 \def\UnicodeMathSymbolA#1#2#3#4{%
13 \ifnum#1>"10000 \printmathsymbol{#1}{#2}{#3}{#4}\fi
14 }
15 \def\printmathsymbol#1#2#3#4{%
16 \hbox{\hbox to2em{${#2}$}\hss}\hbox to3em
17 {\small\printop#3\hss}{\tt\string#2\trycs{eq:\string#2}{}}}
18 }
19 \def\eq#1#2{\sdef{eq:\string#2}{=\string#1}}
20 \eq\diamond\smwhtdiamond \eq\bullet\smblkcircle \eq\circ\vysmwhtcircle
21 \eq\bigcirc\mdlgwhtcircle \eq\to\rightarrow \eq\le\leq
22 \eq\ge\geq \eq\neq\ne \eq\emptyset\varnothing \eq\hbar\hslash
23 \eq\land\wedge \eq\lor\vee \eq\owns\ni \eq\gets\leftarrow
24 \eq\mathring\ocirc \eq\not\neg \eq\backepsilon\upbackepsilon
25 \eq\eth\matheth \eq\dbkarow\dbkarow \eq\drbkarow\drbkarow
26 \eq\hksearrow\hksearrow \eq\hkswarrow\hkswarrow
27
28 \tracinglostchars=0
29 \fontdef\small{\setfontsize{at5pt}\rm}
30 \def\printop{\def\mathop{Op}}
31 \def\mathalpha{Alph}\def\mathord{Ord}\def\mathbin{Bin}\def\mathrel{Rel}
32 \def\mathopen{Open}\def\mathclose{Close}\def\mathpunct{Punct}\def\mathfence{Fence}
33 \def\mathaccent{Acc}\def\mathaccentwide{Accw}\def\mathbotaccentwide{AccBw}

```

```

34 \def\mathbotaccent{AccB}\def\mathaccentoverlay{Acc0}
35 \def\mathover{Over}\def\mathunder{Under}
36 \typsize[7.5/9]\normalmath \everymath={}
37
38 Codes U+00000 \dots\ U+10000
39 \begmulti 3
40 \input unimath-table.opm
41 \endmulti
42
43 \medskip\goodbreak
44 Codes U+10001 \dots\ U+1EEF1 \let\UnicodeMathSymbol=\UnicodeMathSymbolA
45 \begmulti 4
46 \input unimath-table.opm
47 \endmulti
48 \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.

```

3 \codedec1 \typsize {Font managing macros from OPmac <2026-05-12>} % preloaded in format

```

**\typsize** [ $\langle font-size \rangle / \langle baselinekip \rangle$ ] sets given parameters. It sets text font size by the **\setfontsize** macro and math font sizes by setting internal macros **\sizemtext**, **\sizemscript** and **\sizemsscript**. It uses common concept font sizes: 100 %, 70 % and 50 %. The **\setmainvalues** sets the parameters as main values when the **\typsize** is called first.

```

15 \protected\def \typsize [#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=#1pt
23 \edef\sizemtext{\ea\ignorept \the\tmpdim \ptmunit}%
24 \tmpdim=0.7\tmpdim
25 \edef\sizemscript{\ea\ignorept \the\tmpdim \ptmunit}%
26 \tmpdim=#1pt \tmpdim=0.5\tmpdim
27 \edef\sizemsscript{\ea\ignorept \the\tmpdim \ptmunit}%
28 \fi
29 }
30 \public \typsize ;

```

**\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 **\typsize** parameters and runs the **\typsize**.

```

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\typsize\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**.

```

59 \def \setbaselineskip #1{\if$#1$\else
60 \tmpdim=#1\ptunit

```



```

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

```

`\setmainvalues` sets the current font size and `\baselineskip` values to the `\mainfosize` and `\mainbaselineskip` registers 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.

```

99 \_newskip \_mainbaselineskip \_mainbaselineskip=Opt \_relax
100 \_newdimen \_mainfosize \_mainfosize=Opt
101 \_newcount \_mfontsrule \_mfontsrule=1
102
103 \_def\_setmainvalues {%
104 \_mainbaselineskip=\_baselineskip
105 \_mainfosize=\_optsize
106 \_topskip=\_mainfosize \_splittopskip=\_topskip
107 \_ifmmode \_else \_rm \_fi % load and initialize \rm variant
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\_currentgrouplevel=0 \_normalmath
114 \_else \_everymath={\_setmathfonts}\_everydisplay={\_normalmath}%
115 \_let\_runboldmath=\_relax \_fi
116 \_or \_normalmath \_fi}
117 \_def\_scalemain {%
118 \_ifdim \_mainfosize=\_zo
119 \_mainfosize=10pt \_mainbaselineskip=12pt
120 \_let \_setmainvalues=\_setmainvaluesL
121 \_fi
122 \_optsize=\_mainfosize \_baselineskip=\_mainbaselineskip
123 }
124 \_public \scalemain \mainfosize \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.

```

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.

```

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]{\if$#1$\else
153   \tmpdim=#1pt \divide\_\tmpdim by1000
154   \tmpdim=\_ea\_ea\_ea\_ignorept \pdffontsize\_\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.

```

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{\_ifhmode \_\ifdim \_\lastskip>\_\zo
175   \_\skip0=\_\lastskip \_\unskip \_\additcorrA \_\hskip\_\skip0 \_\else \_\additcorrA \_\fi\_fi}
176 \def\_\additcorrA{\_ifnum \_\lastpenalty=\_\zo \_\italcorr \_\else
177   \_ea\_unpenalty \_ea\_italcorr \_ea\_penalty \_\the\_lastpenalty \_\relax \_\fi}
178 \def\_\afteritcorr{\_futurelet \_\next \_\afteritcorrA}
179 \def\_\afteritcorrA{\_ifhmode \_\ifx \_\next.\_\else\_ifx \_\next,\_\else \_\italcorr \_\fi\_fi\_fi}
180 \let\_\italcorr=\_

```

The `\boldify` macro does `\let\rm\bf`, `\let\it\bi` and `\let\normalmath=\boldmath`. All following text will be in bold. If 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.

```

191 \protected\def \boldify {%
192   \_let \_setmainvalues=\_setmainvaluesL
193   \_let\it =\_bi \_let\rm =\_bf \_let\normalmath=\_boldmath \_\bf
194   \_runboldmath
195   \_ifx \_\ncharmA \_\undefined \_\protected\_\addto\rm{\_\fam0 } \_\protected\_\addto\it{\_\fam1 }%
196   \_\else \_\protected\_\def\rm {\_\fmodbf \_\fontsel \_\marm}%
197   \_\protected\_\def\it {\_\fmodbi \_\fontsel \_\mait}%
198   \_\fi
199 }
200 \def\_\runboldmath{\_boldmath}
201
202 \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 `\typosize` command and redefines itself to be only the font switch for the next pages.

```

212 \def \rmfixed {% used in default \footline
213   {\_ifdim \_\mainfosize=0pt \_\mainfosize=10pt \_\fi
214   \_\fontdef\_\tenrm{\_\setfontsize{at\_\mainfosize}\_\resetmod\_\rm}%
215   \_\glet\_\rmfixed=\_\tenrm}% next use will be font switch only
216   \_\rmfixed
217 }
218 \_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 `\gpageno`,
  - prints `\_Xpage{\gpageno}{\pageno}` to the `.ref` file (if `\openref` is active),
  - calculates `\hoffset`,
  - sets local meaning of macros used in headlines/footlines (see `\regmacro`).
- `\shipout\_completepage`, which is `\vbox` of –
  - background box, if `\pgbackground` is non-empty,
  - headline box by `\_makeheadline`, if the `\headline` is nonempty,
  - `\vbox` to `\vsize` of `\_pagecontents` which consists of –
    - `\_pagedest`, the page destination `pg:\gpageno` for hyperlinks is created here,
    - `\topins` box if non-empty (from `\topinserts`),
    - `\box255` with completed vertical material from main vertical mode,
    - `\_footnoterule` and `\footins` box if nonempty (from `\fnote`, `\footnote`),
    - `\pgbottomskip` (default is 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.

```
3 \_codedecl \nopagenumbers {Output routine <2026-05-19>} % preloaded in format
```

output.opm

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

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

output.opm

Default `\_begoutput` and `\_endoutput` is defined. If you need another functionality implemented in the output routine, you can `\addto\_begoutput{...}` or `\addto\_endoutput{...}`. The settings here are local in the `\output` group.

The `\_preproffsets` 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 \_preproffsets \_the\_regmark}
29 \_def \_endoutput{\_advancepageno
30 {\_globaldefs=1 \_the\_nextpages \_nextpages={}}%
31 \_ifnum\_outputpenalty>-20000 \_else\_dosupereject\_fi
32 }
33 \_def \_preproffsets {}
```

output.opm

The `\_optexshipout` does similar work like the `\shipout` primitive. The color literals are added to the `\box0` using the `\_preshipout<destination box number><box specification>` pseudo-primitive. It is defined using lua code, see section 2.39. Finally the `\shipout` primitive is used.

```
43 \_def \_optexshipout #1{\_setbox0=#1\_preshipout0\_box0 \_shipout\_box0 }
```

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.

```
57 \_newdimen \_xhsize \_xhsize=\_hsize
58 \_def\_setxhsize {\_global\_xhsize=\_hsize \_glet\_setxhsize=\_relax}
```

output.opm

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

```
64 \_newcount\_gpageno
65 \_public \gpageno ;
```

output.opm

The `\_completepage` is similar to what plain  $\TeX$  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. Please, correct the words “left” and/or “down” in previous two sequences if `\pagedirection` is nonzero.

The `\_resetattrs` used here means that all newly created texts in output routine (texts used in headline, footline) have default color and no transparency.

output.opm

```

79 \def\_completepage{\_vbox bdir\_pagedirection{%
80   \_resetattrs \_pdfrunninglinkoff
81   \_istokseempty \_pgbackground
82   \_iffalse \_backgroundbox{\_the\_pgbackground}\_nointerlineskip \_fi
83   \_makeheadline
84   \_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
85   \_makefootline}%
86 }
87 \def \_backgroundbox #1{\_moveleft
88   \_ifodd\_pagedirection \_dimexpr\_pagerightoffset+1in\_else \_hoffset\_fi
89   \_vbox to\_zo{\_kern-\_voffset #1\_vss}}

```

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

output.opm

```

96 \def\_makeheadline {\_istokseempty \_headline \_iffalse
97   \_vbox to\_zo{\_vss \_baselineskip=\_headlinedist \_lineskiplimit=-\_maxdimen
98     \_hbox to\_xsize{\_normalbaselines\_the\_headline}\_hbox{}}%
99     }\_nointerlineskip
100   \_fi
101 }

```

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

output.opm

```

107 \def\_makefootline{\_istokseempty \_footline \_iffalse
108   \_baselineskip=\_footlinedist
109   \_lineskiplimit=-\_maxdimen \_hbox to\_xsize{\_normalbaselines\_the\_footline}
110   \_fi
111 }

```

The `\_pagecontents` is similar as in plain  $\TeX$ . The only difference is that the `\_pagedest` is inserted at the top of `\_pagecontents`.

The `\_footnoterule` is defined here.

output.opm

```

119 \def\_pagecontents{\_pagedest % destination of the page
120   \_ifvoid\_topins \_else \_unvbox\_topins\_fi
121   \_dimen0=\_dp255 \_pdfrunninglinkon \_unvbox255 \_pdfrunninglinkoff % open up \box255
122   \_ifvoid\_footins \_else % footnote info is present
123     \_vskip\_skip\_footins
124     \_footnoterule \_unvbox\_footins \_fi
125   \_kern-\_dimen0 \_vskip \_pgbottomskip
126 }
127 \def \_pagedest {{\_def\_destheight{25pt}\_dest[pg:\_the\_pageno]}}
128 \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  $\TeX$  is defined here. Only the `\raggedbottom` macro is defined differently. We use the `\pgbottomskip` register here which is set to 0pt by default.

output.opm

```

139 \countdef\_pageno=0 \_pageno=1 % first page is number 1
140 \def \_folio {\_ifnum\_pageno<0 \_romannumeral-\_pageno \_else \_number\_pageno \_fi}
141 \def \_nopagenumbers {\_footline={}}
142 \def \_advancepageno {%
143   \_ifnum\_pageno<0 \_decr\_pageno \_else \_incr\_pageno \_fi
144 } % increase |pageno|
145 \def \_raggedbottom {\_topskip=\_dimexpr\_topskip plus60pt \_pgbottomskip=0pt plus1fil\_relax}
146 \def \_normalbottom {\_topskip=\_dimexpr\_topskip \_pgbottomskip=0pt\_relax}
147
148 \_public \pageno \folio \nopagenumbers \advancepageno \raggedbottom \normalbottom ;

```

Macros for footnotes are the same as in plain  $\TeX$ . 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 TeX. The `\footnote` and `\footstrut` is defined as in plain TeX.

output.opm

```

161 \_newinsert\_footins
162 \_def \_footnote #1{\_let\_osf=\_empty % parameter #2 (the text) is read later
163   \_ifhmode \_edef\_osf{\_spacefactor\_the\_spacefactor}\\_fi
164   #1\_osf\_vfootnote{#1}}
165 \_def \_vfootnote{\_opfootnote{}}
166 \_def \_opfootnote #1#2{\_insert\_footins\_bgroup
167   \_interlinepenalty=\_interfootnotelinepenalty
168   \_leftskip=\_zo \_rightskip=\_zo \_spaceskip=\_zo \_xspaceskip=\_zo \_relax
169   \_resetatrs
170   #1\_relax % local settings used by \fnote macro
171   \_splittopskip=\_ht\_strutbox % top baseline for broken footnotes
172   \_splitmaxdepth=\_dp\_strutbox \_floatingpenalty=20000
173   \_textindent{#2}\_footstrut
174   \_isnextchar \_bgroup
175   {\_bgroup \_aftergroup\_vfootA \_afterassignment\_ignorespaces \_let\_next=}{\_vfootB}%
176 }
177 \_def \_vfootA{\_unskip\_strut\_egroup}
178 \_def \_vfootB #1{#1\_unskip\_strut\_egroup}
179 \_def \_footstrut {\_vbox to\_splittopskip{}}
180 \_skip\_footins=\_bigskipamount % space added when footnote is present
181 \_count\_footins=1000 % footnote magnification factor (1 to 1)
182 \_dimen\_footins=8in % maximum footnotes per page
183 \_public
184 \footins \footnote \vfootnote \footstrut ;

```

The `\topins` macros `\topinsert`, `\midinsert`, `\pageinsert`, `\endinsert` are the same as in plain TeX.

output.opm

```

192 \_newinsert\_topins
193 \_newif\_ifupage \_newif\_ifumid
194 \_def \_topinsert {\_umidfalse \_upagefalse \_oins}
195 \_def \_midinsert {\_umidtrue \_oins}
196 \_def \_pageinsert {\_umidfalse \_upagetrue \_oins}
197 \_skip\_topins=\_zoskip % no space added when a topinsert is present
198 \_count\_topins=1000 % magnification factor (1 to 1)
199 \_dimen\_topins=\_maxdimen % no limit per page
200 \_def \_oins {\_par \_begingroup\_setbox0=\_vbox\_bgroup\_resetatrs} % start a \_vbox
201 \_def \_endinsert {\_par\_egroup % finish the \_vbox
202   \_ifumid \_dimen0=\_ht0 \_advance\_dimen0 by\_dp0 \_advance\_dimen0 by\_baselineskip
203   \_advance\_dimen0 by\_pagetotal \_advance\_dimen0 by-\_pageshrink
204   \_ifdim\_dimen0>\_pagegoal \_umidfalse \_upagefalse \_fi \_fi
205   \_ifumid \_bigskip \_box0 \_bigbreak
206   \_else \_insert\_topins {\_penalty100 % floating insertion
207     \_splittopskip=0pt
208     \_splitmaxdepth=\_maxdimen \_floatingpenalty=0
209     \_ifupage \_dimen0=\_dp0
210     \_vbox to\_vsize {\_unvbox0 \_kern-\_dimen0}% depth is zero
211     \_else \_box0 \_nobreak \_bigskip \_fi}\_fi\_endgroup}
212
213 \_public \topins \topinsert \midinsert \pageinsert \endinsert ;

```

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

output.opm

```

220 \_def \_draft {\_pgbackground={\_draftbox{\_draftfont DRAFT}}%
221   \_fontdef\_draftfont{\_setfontsize{at10pt}\_bf}%
222   \_glet\_draftfont=\_draftfont
223 }
224 \_def \_draftbox #1{\_setbox0=\_hbox{\_setgreycolor{.8}#1}%
225   \_kern.5\_vsize \_kern\_voffset \_kern4.5\_wd0
226   \_hbox to0pt{\_kern.5\_xhsize \_kern\_hoffset \_kern-2\_wd0
227   \_pdfsave \_pdfrotate{55}\_pdfscale{10}{10}%
228   \_hbox to0pt{\_box0\_hss}%
229   \_pdfrestore
230   \_hss}%
231 }
232 \_public \draft ;

```

## 2.19 Margins

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

margins.opm

`\margins`/*<pg>* *<fmt>* (*<left>*,*<right>*,*<top>*,*<bot>*)*<unit>* takes its parameters, does calculation and sets `\hoffset`, `\voffset`, `\hsize` and `\vsize` registers. Note that OpTeX sets the page origin at the top left corner of the paper, no at the obscure position 1in, 1in. 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     \_xsize=\_hsize \_setxsize % \_xsize used by \output routine
29   \_fi\_fi
30   \_if$#5$\_if$#6$\_voffset=\_dimexpr (\_pgheight -\_vsize)/2 \_relax
31     \_else \_voffset=\_dimexpr \_pgheight -\_vsize - #6\_tmp \_relax % only bottom margin
32   \_fi
33   \_else \_if$#6$\_voffset=#5\_tmp \_relax % only top margin
34   \_else \_vsize=\_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       \_setpagerightoffset}%
41   \_else \_opwarning{use \_string\_margins/1 or \_string\_margins/2}%
42   \_fi\_fi\_fi
43   \_setpagerightoffset
44 }
45 \_def\_setpagedimens{\_isnextchar{\\_setpagedimensB}{\_setpagedimensA}}
46 \_def\_setpagedimensA#1 {\_ifcsname\_pgs:#1\_endcsname
47   \_ea\_ea\_ea\_setpagedimensB \_csname\_pgs:#1\_ea\_endcsname\_space
48   \_else \_opwarning{page specification "#1" is undefined}\_fi}
49 \_def\_setpagedimensB (#1,#2)#3 {\_setpagedimensC\_pgwidth=#1:#3
50   \_setpagedimensC\_pgheight=#2:#3
51   \_pdfpagewidth=\_pgwidth \_pdfpageheight=\_pgheight
52 }
53 \_def\_setpagedimensC #1=#2:#3 {#1=#2\_ifx^#3\_tmp\_else#3\_fi\_relax\_truedimen#1}
54
55 \_public \margins ;

```

The common page dimensions are defined here.

margins.opm

```

61 \_sdef\_pgs:a3}{(297,420)mm} \_sdef\_pgs:a4}{(210,297)mm} \_sdef\_pgs:a5}{(148,210)mm}
62 \_sdef\_pgs:a3l}{(420,297)mm} \_sdef\_pgs:a4l}{(297,210)mm} \_sdef\_pgs:a5l}{(210,148)mm}
63 \_sdef\_pgs:b5}{(176,250)mm} \_sdef\_pgs:letter}{(8.5,11)in}

```

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

`\_truedimen`*<dimen-register>* returns true value of *<dimen-register>* regardless of `\mag`.

margins.opm

```

72 \_def\_trueunit{}
73 \_def\_magscale[#1]{\_mag=#1\_def\_trueunit{true}%
74   \_ifdim\pgwidth=0pt \_else \_truedimen\_pgwidth \_truedimen\_pgheight \_fi
75   \_truedimen\_pdfpagewidth \_truedimen\_pdfpageheight
76 }
77 \_def\_truedimen#1{\_ifx\_trueunit\_empty \_else#1=\_ea\_ignorept\_the#1truept \_fi}
78
79 \_public \magscale ;

```



When left-to-right direction of typesetting is selected (default) then “main vertical line” of the page has `\hoffset` distance from the left paper border and all lines at the page start here and run to the right side (exceptions can be done by `\moveleft` or `\moveright`, of course). When we have set right-to-left direction (using `\textdir TRT`, for example), then the “main vertical line” cannot be at the same position because lines run to the left, i.e. they would be off paper. This is reason why the setting `\pagedir TRT` shifts the “main vertical line” to an alternative position: it has `\pagerightoffset+1in` distance from the *right* paper border and thus right-to-left lines are visible on the paper. We have to set `\pagerightoffset` properly for such cases. This is done in the macro `\_setpagerightoffset`. It must be called whenever `\hoffset` is changed.

margins.opm

```
96 \def\_setpagerightoffset{%
97   \_pagerightoffset=\_dimexpr\_pdfpagewidth-\_xsize-\_hoffset-1in\_relax
98 }
99 \_setpagerightoffset % setting default value from default values
```

Page numbers and numbers of (sub)sections have to be printed in left-to-right mode even though the document mode is right-to-left. We print these numbers via `\_numprint{<number>}` in OpTeX macros. The `\_numprint` is `\_useit` by default (i.e. do nothing special) because we have left-to-right mode as default. But a user can define

```
\_def\_numprint#1{{\_textdir TLT #1}}
```

if the document is set to right-to-left mode.

margins.opm

```
113 \_let\_numprint=\_useit
```

## 2.20 Colors

### 2.20.1 Basic concept

Setting of color in PDF is handled by graphics operators which change the graphics context. Colors for fills/strokes are distinguished, but apart from that, only one color is active at time and is used for all material drawn by following graphics operators, until next color is set. Each PDF content (e.g. page or form XObject) has its own graphics context, that is initialized from zero. Hence we have different concept of selecting fonts in T<sub>E</sub>X (it depends on T<sub>E</sub>X groups but does not depends on pages) and color handling in PDF.

T<sub>E</sub>X itself has no concept of colors. Colors have always been handled by inserting whatsits (either using `\special` for DVI or using `\pdfliteral/\pdfcolorstack` for PDF). It is very efficient and T<sub>E</sub>X doesn’t even have to know anything about colors, but it is also problematic in many ways.

That is the reason why we decided to change color handling from `\pdfcolorstack` to LuaT<sub>E</sub>X attributes in version 1.04 of OpT<sub>E</sub>X. Using attributes, the color setting behaves exactly like font selection from T<sub>E</sub>X point of view: it respects T<sub>E</sub>X groups, colors can span more pages, independent colors can be set for `\inserts`, etc. Moreover, once a material is created (using `\setbox` for example) then it has its fonts and its colors frozen and you can rely on it when you are using e.g. `\unhbox`. There are no internal whatsits for colors which can interfere with other typesetting material. In the end something like setting text to red (`{\Red text}`) should have the same nice behavior like setting text to bold (`{\bf text}`).

LuaT<sub>E</sub>X attributes can be set like count register – one attribute holds one number at a time. But the value of attribute is propagated to each created typesetting element until the attribute is unset or set to another value. Very much like the font property. We use one attribute `\_colorattr` for storing the currently selected color (in number form).

Macros `\setcmykcolor{<C> <M> <Y> <K>}` or `\setrgbcolor{<R> <G> <B>}` or `\setgreycolor{<Grey>}` are used in color selectors. These macros expand to internal `\_setcolor` macro which sets the `\_colorattr` attribute to an integer value and prepares mapping between this value and the real color data. This mapping is used just before each `\shipout` in output routine. The `\_preshipout` pseudo-primitive is used here, it converts attribute values to internal PDF commands for selecting colors.

The concept with color attributes has one limitation: the colors cannot be changed inside a ligature unless the ligature is broken manually. It means that `{\Red f}i` doesn’t lead to the expected result but `{\Red f\null}i` does.

## 2.20.2 Color mixing

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+.6\Yellow}
```

The `\rgbcolordef` can be used to mix colors in additive color model RGB. If `\onlyrgb` is declared, then `\colordef` works as `\rgbcolordef`.

If a CMYK to RGB or RGB to CMYK conversion is needed then direct conversion of given color is used (if declared using `\rgbcmykmap{\langle rgb \rangle}{\langle cmyk \rangle}`) or 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.

## 2.20.3 Implementation

```
3 \_codedecl \colordef {Colors <2022-03-07>} % preloaded in format
```

colors.opm

The basic colors in CMYK `\Blue` `\Red` `\Brown` `\Green` `\Yellow` `\Cyan` `\Magenta` `\Grey` `\LightGrey` `\White` and `\Black` are declared here.

colors.opm

```
12 \_def\Blue      {\_setcmykcolor{1 1 0 0}}
13 \_def\Red      {\_setcmykcolor{0 1 1 0}}
14 \_def\Brown    {\_setcmykcolor{0 .67 .67 .5}}
15 \_def\Green    {\_setcmykcolor{1 0 1 0}}
16 \_def\Yellow   {\_setcmykcolor{0 0 1 0}}
17 \_def\Cyan     {\_setcmykcolor{1 0 0 0}}
18 \_def\Magenta  {\_setcmykcolor{0 1 0 0}}
19 \_def\Grey     {\_setcmykcolor{0 0 0 .5}}
20 \_def\LightGrey{\_setcmykcolor{0 0 0 .2}}
21 \_def\White    {\_setgreycolor{1}}
22 \_def\Black    {\_setgreycolor{0}}
```

By default, the `\setcmykcolor` `\setrgbcolor` and `\setgreycolor` macros with  $\{\langle componentns \rangle\}$  parameter expand to `\_setcolor{\langle color-data \rangle}{\langle fill-op \rangle}{\langle stroke-op \rangle}` where  $\langle color-data \rangle$  is  $\langle R \rangle$   $\langle G \rangle$   $\langle B \rangle$  or  $\langle C \rangle$   $\langle M \rangle$   $\langle Y \rangle$   $\langle K \rangle$  or  $\langle G \rangle$  and  $\langle fill-op \rangle$  is color operator for filling,  $\langle stroke-op \rangle$  is color operator for stroking.

colors.opm

```
33 \_def\_setcmykcolor#1{\_setcolor{#1}kK}
34 \_def\_setrgbcolor#1{\_setcolor{#1}{rg}{RG}}
35 \_def\_setgreycolor#1{\_setcolor{#1}gG}
36 \_public \setcmykcolor \setrgbcolor \setgreycolor ;
```

The `\onlyrgb` declaration redefines `\setcmykcolor` to do conversion to RGB just before `\_setcolor` is used. The `\onlycmyk` declaration redefines `\setrgbcolor` to do conversion to CMYK just before `\_setcolor` is used. Moreover, `\onlyrgb` re-defines three basic RGB colors for RGB color space and re-declares `\colordef` as `\rgbcolordef`.

colors.opm

```
47 \_def\_onlyrgb{\_def\Red{\_setrgbcolor{1 0 0}}%
48 \_def\Green{\_setrgbcolor{0 1 0}}\_def\Blue{\_setrgbcolor{0 0 1}}%
49 \_let\_colordef=\rgbcolordef
50 \_def\_setrgbcolor##1{\_setcolor{##1}{rg}{RG}}%
51 \_def\_setcmykcolor##1{\_ea\_setcolor\_ea{\_expanded{\_cmyktorgb ##1 ;}}{rg}{RG}}%
52 \_public \colordef \setrgbcolor \setcmykcolor ;}
53 \_def\_onlycmyk{%
54 \_let\_colordef=\cmykcolordef
55 \_def\_setrgbcolor##1{\_ea\_setcolor\_ea{\_expanded{\_rgbtocmyk ##1 ;}}kK}%
56 \_def\_setcmykcolor##1{\_setcolor{##1}kK}%
57 \_public \colordef \setrgbcolor \setcmykcolor ;}
58 \_public \onlyrgb \onlycmyk ;
```

The `\_colorattr` for coloring is allocated and `\_setcolor{\langle color-data \rangle}{\langle fill-op \rangle}{\langle stroke-op \rangle}` is defined here. This macro does `\_colorattr=\_colorcnt` if the  $\langle color data \rangle$  was not used before and prepare mapping from this integer value to the  $\langle color data \rangle$  and increments `\_colorcnt`. If the  $\langle color data \rangle$  were used already, then `\_setcolor` does `\_colorattr=\langle stored-value \rangle`. This work is done by the `\_translatecolor` macro. The following mapping macros are created:

```
\_color::\langle data \rangle \langle fill-op \rangle ... expands to used \langle attribute-value \rangle
\_color:\langle attribute-value \rangle ... expands to \langle data \rangle \langle fill-op \rangle
\_color-s:\langle attribute-value \rangle ... expands to \langle data \rangle \langle stroke-op \rangle
```

colors.opm

```
77 \_newattribute \_colorattr
78 \_newcount \_colorcnt \_colorcnt=1 % allocations start at 1
79 \_protected\_def\_setcolor{\_colorprefix\_colorattr=\_translatecolor}
80 \_def\_translatecolor#1#2#3{\_ifcsize\_color::#1 #2\_endcsize\_lastnamedcs\_relax
81 \_else
82 \_colorcnt
83 \_sxdef{\_color::#1 #2}{\_the\_colorcnt}%
84 \_sxdef{\_color:\_the\_colorcnt}{#1 #2}%
85 \_sxdef{\_color-s:\_the\_colorcnt}{#1 #3}%
86 \_incr \_colorcnt
87 \_fi
88 }
89 % Black is the default color.
90 \_sdef{\_color::0 g}{0}
91 \_sdef{\_color:0}{0 g}
92 \_sdef{\_color-s:0}{0 G}
```

We support concept of non-local color, i.e. all changes of the color attribute are global by setting `\_colorprefix` to `\global`. `\localcolor` is the default, i.e. `\_colorprefix` is `\relax`. You can write `\global\Red` if you want to have global setting of the color.

colors.opm

```
102 \_protected\_def \_localcolor {\_let\_colorprefix=\_relax}
103 \_protected\_def \_nolocalcolor {\_let\_colorprefix=\_global}
104 \_public \localcolor \nolocalcolor ;
105 \_localcolor
```

The attribute `\_transpattr` is allocated and set by the `\transparency\langle number \rangle` macro. If such level of the transparency was never used in the document then `\addextgstate{tr\langle number \rangle}{<</ca X /CA X>>}` is applied (where X is  $(255 - \langle number \rangle) / 255$ ). This information is used when shipout is processed (similarly

as colors). It means `/tr⟨number⟩ gs` is inserted when the attribute is changed.  
`\_resetattrs` resets the `\_colorattr` and `\_transpattr` to their initial value `~"7FFFFFFF`.

colors.opm

```

119 \_newattribute\_transpattr
120 \_def\_transparency {\_afterassignment\_transparencyA \_transpattr}
121 \_def\_transparencyA{%
122   \_ifnum\_transpattr<1 \_transpattr=\_noattr \_else
123     \_ifnum\_transpattr>255 \_opwarning{\_noexpand\_transparency > 255 not allowed}%
124     \_transpattr=\_noattr
125   \_else
126     \_ifcsname\_transp:\_the\_transpattr\_endcsname \_else
127       \_edef\_transpv{\_expr{(255-\_the\_transpattr)/255}}%
128       \_addextgstate{tr\_the\_transpattr}{<</ca \_transpv\_space /CA \_transpv>>}%
129       \_sxdef{\_transp:\_the\_transpattr}{}%
130       \_ifcsname\_transp:0\_endcsname \_else
131         \_addextgstate{tr0}{<</ca 1 /CA 1>>}%
132         \_sxdef{\_transp:0}{}%
133     \_fi
134   \_fi
135 \_fi
136 }
137 }
138 \_def\_thetransparency{\_ifnum \_transpattr=-"7FFFFFFF 0\_else \_the\_transpattr \_fi}
139 \_def\_resetattrs{\_colorattr=\_noattr \_transpattr=\_noattr}
140
141 \_public \_transparency \_thetransparency ;

```

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. The `\_tocmyk:⟨rgb⟩` or `\_torgb:⟨cmyk⟩` control sequences (given by `\_rgbcmykmap`) have precedence.

colors.opm

```

158 \_def\_rgbtocmyk #1 #2 #3 ;{\_trycs{\_tocmyk:#1 #2 #3}{%
159   \_ea \_stripzeros \_detokenize \_ea{\_directlua{
160     local kr = math.max(#1,#2,#3)
161     if (kr==0) then
162       tex.print('0. 0. 0. 1 ;')
163     else
164       tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
165         (kr-#1)/kr, (kr-#2)/kr, (kr-#3)/kr, 1-kr))
166     end
167   }}}
168 \_def\_cmyktorgb #1 #2 #3 #4 ;{\_trycs{\_torgb:#1 #2 #3 #4}{%
169   \_ea \_stripzeros \_detokenize \_ea{\_directlua{
170     local kr = 1-#4
171     tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f ;',
172       (1-#1)*kr, (1-#2)*kr, (1-#3)*kr))
173   }}}
174 \_def\_colorcrop{\_directlua{
175   local m=math.max(color_C, color_M, color_Y, color_K)
176   if (m>1) then
177     color_C=color_C/m color_M=color_M/m color_Y=color_Y/m color_K=color_K/m
178   end
179 }}
180 \_def\_colordefFin{\_colorcrop \_ea \_stripzeros \_detokenize \_ea{\_directlua{
181   tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
182     color_C, color_M, color_Y, color_K))
183 }}}
184 \_def\_douseK{\_colorcrop \_directlua{
185   kr=math.min(color_C, color_M, color_Y)
186   if (kr>=1) then
187     color_C=0 color_M=0 color_Y=0 color_K=1
188   else
189     color_C=(color_C-kr)/(1-kr) color_M=(color_M-kr)/(1-kr)
190     color_Y=(color_Y-kr)/(1-kr) color_K=math.min(color_K+kr,1)

```

```

191   end
192 }}

```

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`.

colors.opm

```

201 \_def\_stripzeros #1.#2 #3{\_ifx0#1\_else#1\_fi.\_stripzeroA #2 0 :%
202   \_ifx;#3\_else \_space \_ea\_stripzeros\_ea#3\_fi}
203 \_def\_stripzeroA #10 #2:{\_ifx^#2\_stripzeroC#1:\_else \_stripzeroB#1 0 :\_fi}
204 \_def\_stripzeroB #10 #2:{\_ifx^#2\_stripzeroC#1:\_else #1\_fi}
205 \_def\_stripzeroC #1 #2:{#1}

```

`\rgbcolordef`  $\langle R \rangle \langle G \rangle \langle B \rangle \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle$  declares mapping from RGB to CMYK and from CMYK to RGB for given color. It has precedence before general formulae used in the `\_rgbtocmyk` and `\_cmyktorgb` macros. Note, that the values  $\langle R \rangle \langle G \rangle \langle B \rangle \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle$  must be given exactly in the same format as in `\setcmykcolor` and `\setrgbcolor` parameters. For example, 0.5 or .5 or .50 are different values from point of view of this mapping.

colors.opm

```

217 \_def\_rgbcolordef#1#2{\_sxdef\_torgb:#2}{#1}\_sxdef\_tocmyk:#1}{#2}}
218 \_public \rgbcolordef ;

```

The `\rgbcolordef` and `\cmykcolordef` use common macro `\_commoncolordef` with different first four parameters. The `\_commoncolordef`  $\langle selector \rangle \langle K \rangle \langle R \rangle \langle G \rangle \langle what-define \rangle \{ \langle data \rangle \}$  does the real work. It initializes the Lua variables for summation. It expands  $\langle data \rangle$  in the group where color selectors have special meaning, then it adjusts the resulting string by `\replstring` and runs it. Example shows how the  $\langle data \rangle$  are processed:

```

input <data>: ".3\Blue + .6~\KhakiC \useK -\Black"
expanded to: ".3 !=K 1 1 0 0 +.6~!=R .804 .776 .45 \_useK -!=G 0"
adjusted to: "\_addcolor .3!=K 1 1 0 0 \_addcolor .6!~R .804 .776 .45
              \_useK \_addcolor -1!=G 0"
and this is processed.

```

`\_addcolor`  $\langle coef. \rangle ! \langle mod \rangle \langle type \rangle$  expands to `\_addcolor: \langle mod \rangle \langle type \rangle \langle coef \rangle` for example it expands to `\_addcolor:=K \langle coef \rangle` followed by one or three or four numbers (depending on  $\langle type \rangle$ ).  $\langle mod \rangle$  is = (use as is) or ~ (use complementary color).  $\langle type \rangle$  is K for CMYK, R for RGB and G for GREY color space. Uppercase  $\langle type \rangle$  informs that `\cmykcolordef` is processed and lowercase  $\langle type \rangle$  informs that `\rgbcolordef` is processed. All variants of commands `\_addcolor: \langle mod \rangle \langle type \rangle` are defined. All of them expand to `\_addcolorA \langle v1 \rangle \langle v2 \rangle \langle v3 \rangle \langle v4 \rangle` which adds the values of Lua variables. The `\rgbcolordef` uses `\_addcolorA \langle R \rangle \langle G \rangle \langle B \rangle 0` and `\cmykcolordef` uses `\_addcolorA \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle`. So the Lua variable names are a little confusing when `\rgbcolordef` is processed.

Next, `\_commoncolordef` saves resulting values from Lua to `\_tmpb` using `\_colordefFin`. If `\rgbcolordef` is processed, then we must to remove the last  $\langle K \rangle$  component which is in the format .0 in such case. The `\_stripK` macro does it. Finally, the  $\langle what-define \rangle$  is defined as  $\langle selector \rangle \{ \langle expanded\_tmpb \rangle \}$ , for example `\_setcmykcolor{1 0 .5 .3}`.

colors.opm

```

255 \_def\_rgbcolordef {\_commoncolordef \setrgbcolor krg}
256 \_def\_cmykcolordef {\_commoncolordef \setcmykcolor KRG}
257 \_def\_commoncolordef#1#2#3#4#5#6{%
258   \_begingroup
259     \_directlua{color_C=0 color_M=0 color_Y=0 color_K=0}%
260     \_def\_setcmykcolor##1{!=#2 ##1 }%
261     \_def\_setrgbcolor ##1{!=#3 ##1 }%
262     \_def\_setgreycolor##1{!=#4 ##1 }%
263     \_let\_useK=\_relax
264     \_edef\_tmpb{+ #6}%
265     \_replstring\_tmpb{+ }{+}\_replstring\_tmpb{- }{-}%
266     \_replstring\_tmpb{+}{\_addcolor}\_replstring\_tmpb{-}{\_addcolor-}%
267     \_replstring\_tmpb{^!=}{!}\_replstring\_tmpb{-!}{-!}%
268     \_ifx K#2\_let\_useK=\_douseK \_fi
269     \_tmpb
270     \_edef\_tmpb{\_colordefFin}%
271     \_ifx k#2\_edef\_tmpb{\_ea\_stripK \_tmpb}\_fi
272   \_ea\_endgroup

```

```

273 \_ea\_def\_ea#5\_ea{\_ea#1\_ea{\_tmpb}}}%
274 }
275 \_def\_addcolor#1!#2#3{\_cs{addcolor:#2#3}#1}
276 \_def\_addcolorA #1 #2 #3 #4 #5 {%
277   \_def\_tmpa{#1}\_ifx\_tmpa\_empty \_else \_edef\_tmpa{\_tmpa*}\_fi
278   \_directlua{color_C=math.max(color_C+\_tmpa#2,0)
279               color_M=math.max(color_M+\_tmpa#3,0)
280               color_Y=math.max(color_Y+\_tmpa#4,0)
281               color_K=math.max(color_K+\_tmpa#5,0)
282 }}
283 \_sdef{addcolor:=K}#1 #2 #3 #4 #5 {\_addcolorA #1 #2 #3 #4 #5 }
284 \_sdef{addcolor:`K}#1 #2 #3 #4 #5 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) #5 }
285 \_sdef{addcolor:`G}#1 #2 {\_addcolorA #1 0 0 0 #2 }
286 \_sdef{addcolor:=G}#1 #2 {\_addcolorA #1 0 0 0 (1-#2) }
287 \_sdef{addcolor:=R}#1 #2 #3 #4 {%
288   \_edef\_tmpa{\_noexpand\_addcolorA #1 \_rgbtocmyk #2 #3 #4 ; }\_tmpa
289 }
290 \_sdef{addcolor:`R}#1 #2 #3 #4 {\_cs{addcolor:=R}#1 (1-#2) (1-#3) (1-#4) }
291
292 \_sdef{addcolor:=k}#1 #2 #3 #4 #5 {%
293   \_edef\_tmpa{\_noexpand\_addcolorA #1 \_cmyktorgb #2 #3 #4 #5 ; 0 }\_tmpa
294 }
295 \_sdef{addcolor:`k}#1 #2 #3 #4 #5 {\_cs{addcolor:=k}#1 (1-#2) (1-#3) (1-#4) #5 }
296 \_sdef{addcolor:`g}#1 #2 {\_addcolorA #1 (1-#2) (1-#2) (1-#2) 0 }
297 \_sdef{addcolor:=g}#1 #2 {\_addcolorA #1 #2 #2 #2 0 }
298 \_sdef{addcolor:=r}#1 #2 #3 #4 {\_addcolorA #1 #2 #3 #4 0 }
299 \_sdef{addcolor:`r}#1 #2 #3 #4 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) 0 }
300 \_def\_stripK#1 .0;{#1}
301 \_let\_colordef=\_cmykcolordef % default \_colordef is \_cmykcolordef

```

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

colors.opm

```

309 \_def \useK{\_useK}
310 \_def \colordef {\_colordef}
311 \_public \cmykcolordef \rgbcolordef ;

```

The  $\text{\LaTeX}$  file `x11nam.def` is read by `\morecolors`. The numbers 0,1,2,3,4 are transformed to letters O, *(none)*, B, C, D in the name of the color. Colors defined already are not re-defined. The empty `\_showcolor` macro should be re-defined for color catalog printing. For example:

```

\def\vr{\vrule height10pt depth2pt width20pt}
\def\_showcolor{\hbox{\tt\_bslash\_tmpb: \csname\_tmpb\endcsname \vr}\space\space}
\beginmulti 4 \typsize[10/14]
\morecolors
\endmulti

```

colors.opm

```

327 \_def\_morecolors{%
328   \_long\_def\_tmp##1\preparecolorset##2##3##4##5{\_tmpa ##5;;;}
329   \_def\_tmpa##1,##2,##3,##4;{\_ifx,##1,\_else
330     \_def\_tmpb{##1}\_replstring\_tmpb{1}{\_replstring\_tmpb{2}{B}%
331     \_replstring\_tmpb{3}{C}\_replstring\_tmpb{4}{D}\_replstring\_tmpb{0}{0}%
332     \_ifcsname \_tmpb\endcsname \_else
333     \_sdef{\_tmpb}{\_setrgbcolor{##2 ##3 ##4}}\_showcolor\_fi
334     \_ea\_tmpa\_fi
335   }
336   \_ea\_tmp\_input x11nam.def
337 }
338 \_let\_showcolor=\_relax % re-define it if you want to print a color catalog
339 \_public \morecolors ;

```

## 2.21 The .ref file

A so called `.ref` (`\jobname.ref`) file is used to store data that will be needed in the next  $\text{\TeX}$  run (information about references, TOC lines, etc.). If it exists it is read by `\everyjob`, when processing of the document starts, but it is not created at all if the document doesn't need any forward references. Here are the typical contents of a `.ref` file:



```

\Xrefversion{\ref-version}}
\_Xpage{<gpageno>}{<pageno>}
\_Xtoc{<level>}{<type>}{<text>}{<title>}
\_Xlabel{<label>}{<text>}
\_Xlabel{<label>}{<text>}
...
\_Xpage{<gpageno>}{<pageno>}
\_Xlabel{<label>}{<text>}
...

```

- `\_Xpage` corresponds to the beginning of a page. `<gpageno>` is an internal page number, globally numbered from one. `<pageno>` is the page number (`\the\pageno`) used in pagination (they may differ).
- `\_Xtoc` corresponds to a chapter, section or subsection title on a page. `<title>` is the title of the chapter (`<level>=1`, `<type>=chap`), section (`<level>=2`, `<type>=sec`) or subsection (`<level>=3`, `<type>=secc`).
- `\_Xlabel` corresponds to a labelled object on a page. `<label>` is the label provided by the user in `\label[<label>]`, while `<text>` is the text which should be used for the reference (section or table number, for example 2.3.14).

ref-file.opm

```
3 \_codedecl \openref {File for references <2021-07-19>} % preloaded in format
```

The `\_inputref` macro is executed in `\everyjob`. It reads the `\jobname.ref` file, if it exists. After the file is read then it is removed and opened for writing.

ref-file.opm

```

11 \_newwrite\_reffile
12
13 \_def\_inputref {%
14   \_isfile{\_jobname.ref}\_iftrue
15     \_input {\_jobname.ref}%
16     \_edef\_prevrefhash{\_mdfive{\_jobname.ref}}%
17     \_gnotenum=0 \_lfnenum=0 \_mnotenum=0
18     \_openref
19   \_fi
20 }

```

`\_mdfive{<file>}` expands to the MD5 hash of a given file. We use it to do consistency checking of the `.ref` file. First, we read the MD5 hash of `.ref` file from previous `TeX` run before it is removed and opened for writing again in the `\_inputref` macro. The hash is saved to `\_prevrefhash`. Second, we read the MD5 hash in the `\_byehook` macro again and if these hashes differ, warning that “ref file has changed” is printed. Try running `optex op-demo` twice to see the effect.

ref-file.opm

```

32 \_def\_mdfive#1{\_directlua{optex.mdfive("#1")}}
33 \_def\_prevrefhash{}

```

If the `.ref` file does not exist, then it is not created by default. This means that if you process a document without any forward references then no `\jobname.ref` file is created (it would be unusable). The `\_wref` macro is a dummy in that case.

ref-file.opm

```

42 \_def\_wrefrelax#1#2{}
43 \_let\_wref=\_wrefrelax

```

If a macro needs to create and use the `.ref` file, then such macro must first use `\openref`. It creates the file and redefines `\_wref` `\<macro>{<data>}` so that it saves the line `\<macro>{<data>}` to the `.ref` file using the asynchronous `\write` primitive. Finally, `\_openref` destroys itself, because we don't need to open the file again.

`\_wref{<cname>}{<params>}` in fact does `\write\_reffile{\string{<cname>}{<params>}}` and similarly `\_ewref{<cname>}{<params>}` does `\write\_reffile{\string{<cname>}{<expanded-params>}}`.

ref-file.opm

```

57 \_def\_openref {%
58   \_immediate\_openout\_reffile="\_jobname.ref"\_relax
59   \_gdef\_wref ##1##2{\_write\_reffile{\_bslash\_csstring##1##2}}%
60   \_immediate\_write\_reffile {\_pcent\_pcent\_space OpTeX <\_optexversion> - REF file}%
61   \_immediate\_wref \Xrefversion{\\_REFversion}}%
62   \_ifx\_refdecldata\_empty \_else \_refdeclwrite \_fi

```

```

63 \_gdef\_openref{}%
64 }
65 \_def\_ewref #1#2{\_edef\_ewrefA{#2}\_ea\_wref\_ea#1\_eaf\_ewrefA}}
66 \_def\_openref{\_openref}

```

We are using the convention that the macros used in `.ref` file are named `\_X{foo}`. We don't want to read `.ref` files from old, incompatible versions of OpTeX (and OPmac). This is ensured by using a version number and the `\Xrefversion` macro at the beginning of the `.ref` file:

```
\Xrefversion{<version>}
```

The macro checks the version compatibility. Because OPmac does not understand `\_Xrefversion` we use `\Xrefversion` (with a different number of `<version>` than OPmac) here. The result: OPmac skips `.ref` files produced by OpTeX and vice versa.

```

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

```

ref-file.opm

You cannot define your own `.ref` macros before `.ref` file is read because it is read in `\everyjob`. But you can define such macros by using `\refdecl{<definitions of your ref macros>}`. This command writes `<definitions of your ref macros>` to the `.ref` file. Then the next lines written to the `.ref` file can include your macros. An 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 `<definitions of your ref macros>` while `#` has the catcode 12, because we don't want to duplicate each `#` in the `.ref` file.

`\refdecl` appends its data to the `\_refdecldata` macro. It is pushed to the `.ref` file immediately only if the file is opened already. Otherwise we are waiting to `\openref` because we don't want to open the `.ref` file if it is unnecessary.

```

111 \_def\_refdecldata{}
112 \_def\_refdecl{\_bgroup \_catcode\#=12 \_catcode\=12 \_catcode\ =12 \_refdeclA}
113 \_def\_refdeclA#1{\_egroup
114   \_ifx\_refdecldata\_empty\_else \_global\_addto\_refdecldata{^^J}\_fi
115   \_global\_addto\_refdecldata{#1}%
116   \_ifx\_openref\_empty \_refdeclwrite \_fi
117 }
118 \_def\_refdeclwrite{%
119   \_immediate\_write\_reffile{\_pcent\_space \_string\refdecl:^^J\_detokenize\_ea{\_refdecldata}}%
120   \_gdef\_refdecldata{}%
121 }
122 \_public \refdecl ;

```

ref-file.opm

## 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.

```

3 \_codedecl \ref {References <2023-07-03>} % preloaded in format

```

references.opm

`\_Xpage {<gpageno>}{<pageno>}` 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}}\_lfnotenum=0 }

```

references.opm

Counter for the number of unresolved references `\_unresolvedrefs`. It is set but unused in OpTeX versions 1.04+. You can add the report, for example:

```

\_addto\_byhook{\_ifnum\_unresolvedrefs>0 \_opwarning
  {There are \_the\_unresolvedrefs\_space unresolved references}\_fi}

```

```

22 \_newcount\_unresolvedrefs
23 \_unresolvedrefs=0

```

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

```

30 \_def\_Xlabel#1#2{\_sdef{\_lab:#1}{#2}\_sxdef{\_pgref:#1}{\_currpage}}

```

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

```

38 \_def\_label[#1]{\_isempty{#1}\_iftrue \_glet \_lastlabel=\_undefined
39 \_else \_isdefined{10:#1}%
40 \_iftrue \_slideshow\_opwarning{Duplicated label [#1], ignored}\_else \_xdef\_lastlabel{#1}\_fi
41 \_fi \_ignorespaces
42 }
43 \_let \_slideshow=\_relax % redefined if \slides + \slideshow.
44 \_def\_wlabel#1{%
45 \_ifx\_lastlabel\_undefined \_else
46 \_dest[ref:\_lastlabel]%
47 \_printlabel\_lastlabel
48 \_ewref \_Xlabel {(\_lastlabel){#1}}%
49 \_sxdef{\_lab:\_lastlabel}{#1}\_sxdef{10:\_lastlabel}{}%
50 \_glet\_lastlabel=\_undefined
51 \_fi
52 }
53 \_public \label \wlabel ;

```

`\ref[<label>]{<given-text>}` prints (linked) `<given-text>`. The missing optional `{<given-text>}` is replaced by `@`. The `@` is replaced by `<implicit-text>` from saved `\lab:<label>` using `\_reftext` macro. 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>]{<given-text>}` prints `<given-text>` where `@` is replaced by `<pageno>`. Data in the format `{<gpageno>}{<pageno>}` are read from `\_pgref:<label>` by `\_pgrefB{<gpageno>}{<pageno>}{<given-text>}`. `\_lastreflabel` keeps the value of the last label read by `\ref` or `\pgref`. You can use it for example by definig a macro `\pgr` by `\def\pgr{\pgref[\_lastreflabel]}` and then you need not repeat the same label in typical situations and you can write for instance: see section `\ref[lab]` at page `\pgr`.

```

74 \_def\_ref[#1]{\_xdef\_lastreflabel{#1}\_isnextchar\_bgroup{\_refA}{\_refA{@}}}
75 \_def\_refA #1{\_isdefined{\_lab:\_lastreflabel}%
76 \_iftrue \_ilink[ref:\_lastreflabel]{\_reftext{\_csname\_lab:\_lastreflabel\_endcsname}{#1}}%
77 \_else \_reftext{??}{#1}\_opwarning[label [\_lastreflabel] unknown. Try to TeX me again}%
78 \_incr\_unresolvedrefs \_openref
79 \_fi
80 }
81 \_def\_pgref[#1]{\_xdef\_lastreflabel{#1}\_isnextchar\_bgroup{\_pgrefA}{\_pgrefA{@}}}
82 \_def\_pgrefA #1{\_isdefined{\_pgref:\_lastreflabel}%
83 \_iftrue \_ea\_ea\_ea\_pgrefB \_csname\_pgref:\_lastreflabel\_endcsname{#1}%
84 \_else \_reftext{??}{#1}\_opwarning[pg-label [\_lastreflabel] unknown. Try to TeX me again}%
85 \_incr\_unresolvedrefs \_openref
86 \_fi
87 }
88 \_def\_pgrefB #1#2#3{\_ilink[pg:#1]{\_reftext{#2}{#3}}}
89
90 \_public \ref \pgref ;

```

`\_reftext{<implicit-text>}{<given-text>}` expands to the `<given-text>` but the optional `@` in the `<given-text>` is replaced by the `<implicit-text>` first.

```

97 \_def\_reftext #1#2{\_isatin #2@\_iffalse \_numprint{#2}\_else\_reftextA{#1}#2\_fin \_fi}
98 \_def\_reftextA #1#2#3{\_fin {#2\_numprint{#1}#3}
99 \_def\_isatin #1#2{\_iffalse {\_ifx\_fin#2\_fin}

```

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. The color of labels is set by `\_labelcolor` (default is RGB blue).

```

108 \def\printlabel#1{}
109 \def\labelcolor{\setrgbcolor{0 0 1}}
110 \def\showlabels {%
111   \def\printlabel##1{\_vbox to\_zo{\_vss\_llap{\labelcolor\_labelfont[##1]}\_kern1.7ex}}%
112   \fontdef\_labelfont{\setfontsize{at10pt}\_tt}
113 }
114 \_public \showlabels ;

```

## 2.23 Hyperlinks

There are six types of internal links and one type of external link used in OpTeX. They are used in the format  $\langle type \rangle : \langle spec \rangle$ .

- **ref**: $\langle label \rangle$  – the destination is created when  $\backslash label[\langle label \rangle]$  is used, see also the section 2.22.
- **toc**: $\langle tocrefnun \rangle$  – the destination is created at chap/sec/secc titles, see also the section 2.24.
- **pg**: $\langle gpageno \rangle$  – the destination is created at beginning of each page, see also the section 2.18.
- **cite**: $\langle bibpart \rangle / \langle bibnum \rangle$  – the destination is created in bibliography reference, see section 2.32.1.
- **fnt**: $\langle gfnotenun \rangle$  – link from text to footnote, see also section 2.34.
- **fnf**: $\langle gfnotenun \rangle$  – link from footnote to text, see also section 2.34.
- **url**: $\langle url \rangle$  – used by  $\backslash url$  or  $\backslash link$ , see also the end of this section.

The  $\langle tocrefnun \rangle$ ,  $\langle gpageno \rangle$ ,  $\langle bibnum \rangle$ , and  $\langle gfnotenun \rangle$  are numbers starting from one and globally incremented by one in the whole document. The registers  $\backslash tocrefnun$ ,  $\backslash gpageno$ ,  $\backslash bibnum$ , and  $\backslash gfnotenun$  are used for these numbers.

When a chap/sec/secc title is prefixed by  $\backslash label[\langle label \rangle]$ , then both types of internal links are created at the same destination place: **toc**: $\langle tocrefnun \rangle$  and **ref**: $\langle label \rangle$ .

The color for active links can be declared by  $\backslash def\_ \langle type \rangle linkcolor$ , the border around link can be declared by  $\backslash def\_ \langle type \rangle border$ . These macros are not declared by default, so color for active links are given only by  $\backslash hyperlinks$  macro and borders are invisible. For example  $\backslash def\_ toclinkcolor\{Red\}$  means that links from table of contents are in red. Another example  $\backslash def\_ tocborder\{1 0 0\}$  causes red frames in TOC (not printed, only visible in PDF viewers).

```

3 \_codedec1 \link {Hyperlinks <2026-05-19>} % preloaded in format

```

$\backslash dest[\langle type \rangle : \langle spec \rangle]$  creates a destination of internal links. The destination is declared in the format  $\langle type \rangle : \langle spec \rangle$ . If the  $\backslash hyperlinks$  command is not used, then  $\backslash dest$  does nothing else it is set to  $\backslash destactive$ . The  $\backslash destactive$  is implemented by  $\backslash pdfdest$  primitive. It creates a box using  $\backslash destbox[\langle type \rangle : \langle spec \rangle]$  in which the destination is shifted by  $\backslash destheight$ . 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 differently dependent on the current vertical or horizontal mode.

```

17 \def\destheight{1.4em}
18 \def\destactive[#1:#2]{\_if$#2$\_else\_ifvmode
19   \_tmpdim=\_prevdepth \_prevdepth=-1000pt
20   \_destbox[#1:#2]\_prevdepth=\_tmpdim
21   \_else \_destbox[#1:#2]%
22   \_fi\_fi
23 }
24 \def\_destbox[#1]{\_vbox to\_zo{\_kern-\_destheight \_pdfdest name{#1} xyz\_vss}}
25 \def\_dest[#1]{ }
26 \_public \dest ;

```

Each hyperlink is created internally by  $\backslash xlink\{\langle type \rangle\}\{\langle spec \rangle\}\{\langle color \rangle\}\{\langle text \rangle\}$ . This macro expands to  $\backslash quitvmode\{\langle text \rangle\}$  by default, i.e. no active hyperlink is created, only  $\langle text \rangle$  is printed in horizontal mode (and in a group). If  $\backslash hyperlinks$  is used, then  $\backslash xlink$  gets the meaning of  $\backslash xlinkactive$  and hyperlinks are created by the  $\backslash pdfstartlink/\backslash pdfendlink$  primitives. The  $\langle text \rangle$  has given  $\langle color \rangle$  only when hyperlink is created. If  $\backslash \langle type \rangle linkcolor$  is defined, it has precedence over  $\langle color \rangle$ .

The  $\backslash linkdimens$  macro declares the dimensions of link area.

A specific action can be defined for each link  $\langle type \rangle$  by the macro  $\backslash \langle type \rangle action\{\langle spec \rangle\}$ . OpTeX defines only  $\backslash urlaction\{\langle url \rangle\}$ . The default link action (when  $\backslash \langle type \rangle action$  is not defined) is  $goto name\{\langle type \rangle : \langle spec \rangle\}$  (an internal link). It is declared in the  $\backslash linkactions\{\langle type \rangle\}\{\langle spec \rangle\}$  macro. The  $\backslash pdfstartlink$  primitive uses  $attr\{\backslash pdfborder\{\langle type \rangle\}\}$ . The  $\backslash pdfborder\{\langle type \rangle\}$  macro expands to  $/C[? ? ?] /Border[0 0 .6]$  if the  $\backslash \langle type \rangle border$  macro (i.e.  $\backslash refborder$ ,  $\backslash citeborder$ ,

`\_tocborder`, `\_pgborder`, `\_urlborder`, `\_fntborder` or `\_fnfborder`) is defined.

The `\_xlink` is surrounded by the pair `\_startxlink` and `\_stopxlink`. Their default values are `\pdfrunninglinkon` and `\pdfrunninglinkoff`, i.e. the link can be broken to lines and pages. These primitives are used in output routine too in order to suppress the active area of the link in headline and footline when the link is broken to more pages. But the case of a broken link to more pages AND broken another links to more lines or pages in footnotes at the same page doesn't work correctly. This very specific case should be solved by LuaTeX attributes. For the sake of simplicity, OpTeX does not handle this now.

hyperlinks.opm

```

62 \_protected\_def\_xlinkactive#1#2#3#4{\_quitvmode
63 \_startxlink \_pdfstartlink \_linkdimens attr{\_pdfborder{#1}}\_linkactions{#1}{#2}\_relax
64 {\_localcolor\_trycs{#1linkcolor}{#3}{#4}\_pdfendlink \_stopxlink
65 }
66 \_protected\_def\_xlink#1#2#3#4{\_quitvmode{#4}}
67
68 \_def\_linkdimens{height.9em depth.3em}
69
70 \_def\_linkactions#1#2{\_ifcsname \_#1action\_endcsname
71 \_lastnamedcs{#2}\_else goto name{#1:#2}\_fi}
72 \_def\_urlaction #1{user{/Subtype/Link/A <</Type/Action/S/URI/URI(#1)>>}}
73
74 \_def\_pdfborder#1{\_ifcsname \_#1border\_endcsname
75 /C [\_csname \_#1border\_endcsname] /Border [0 0 .6]\_else /Border [0 0 0]\_fi
76 }
77
78 \let\_startxlink=\_pdfrunninglinkon \let\_stopxlink=\_pdfrunninglinkoff

```

`\link[⟨type⟩:⟨spec⟩]{⟨color⟩}{⟨text⟩}` creates a link. It is kept here for backward compatibility and it is equivalent to `\_xlink{⟨type⟩}{⟨spec⟩}{⟨color⟩}{⟨text⟩}`. If `\_⟨type⟩action` is not defined then `\link` creates internal link do the `\dest[⟨type⟩:⟨spec⟩]`. You can have more links with the same `⟨type⟩:⟨spec⟩` but only one `\dest` in the document.

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

`\ulink[⟨url⟩]{⟨text⟩}` creates external link. The `⟨url⟩` is detokenized with `\escapechar=-1` before it is used, so `\%`, `\#` etc. can be used in the `⟨url⟩`.

hyperlinks.opm

```

99 \_def\_link[#1:#2]{\_xlink{#1}{#2}}
100 \_def\_ilink[#1:#2]#3{\_xlink{#1}{#2}\_ilinkcolor{#3}}
101 \_def\_ulink[#1]#2{\_escapechar=-1 \_ea}\_expanded
102 {\_noexpand\_xlink{url}{\_detokenize{#1}}}\_elinkcolor{#2}}
103
104 \_public \ilink \ulink \link ;

```

`\hyperlinks{ilink color}{ulink color}` activates `\dest`, `\xlink`, so that they create links. Not setting colors (`\hyperlinks{}{}`) is also supported.

hyperlinks.opm

```

112 \_def\_hyperlinks#1#2{%
113 \_let\_dest=\_deactive \_let\_xlink=\_xlinkactive
114 \_let\_ilinkcolor=#1\_empty
115 \_let\_elinkcolor=#2\_empty
116 \_public \dest \xlink ;%
117 }
118 \_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 processed in four steps: 1. the `\|` are replaced by `[|]` (we suppose that such string does not exist in any URL). 2. it is detokenized with `\escapechar=-1`. 3. multi-strings and spaces are replaced by strings in braces `{...}`. 4. internal penalties and skips are put between characters using `\_urlA`, `\_urlB` and `\_urlC`. The step 4 do following: The `\_urlxskip` is inserted between each pair of “normal characters”, i.e. characters not declared by `\sdef{\_ur:⟨character⟩}`. The special characters declared by `\sdef{\_ur:⟨character⟩}` are replaced by the body of their corresponding macro. The `\_urlskip`, `\_urlbskip`, `\_urlgskip` are typical skips used for special characters, their meaning is documented in the code below. You can change them. Default values: penalty 9990 is inserted between each pair of normal characters, penalty 100 is inserted after special characters, nobreak before special characters. The

URL can be broken at any place using these default values. If you want to disable breaking between normal characters, say `\let\urlxskip=\nobreak`.  
The text version of the `\url` is printed in `\urlfont`.

hyperlinks.opm

```

145 \_def\url#1{%
146   \_def\tmpa{#1}\_replstring\tmpa {\|}{}%
147   \_def\tmpb{#1}\_replstring\tmpb {\|}{[|]}%
148   {\_escapechar=-1 \_ea}\_ea\edef\_ea\tmpb\_ea{\_detokenize\_ea\tmpb}%
149   \_replstring\tmpb{[|]}{\_gb|}%
150   \_replstring\tmpb{ }{\_ }%
151   \_replstring\tmpb{://}{\_://}%
152   \_ea\_ulink \_ea[\_ea\tmpa] {\_urlfont \_textdirection=0 \_ea\_urlA\tmpb\_fin}%
153 }%
154 \_def\urlA#1{\_ifx\_fin#1\_else \_urlC{\_#1}\_fi}
155 \_def\urlB#1{\_ifx\_fin#1\_else \_urlC{\_urlxskip}\_#1}\_fi}
156 \_def\urlC#1#2{%
157   \_ifcsname\_ur:#2\_endcsname \_lastnamedcs \_ea\_ea\_ea \_urlA
158   \_else #1#2\_ea\_ea\_ea \_urlB \_fi
159 }
160 \_sdef{\_ur: //}{\_urlskip:\_urlskip/\_urlskip/\_urlbskip}
161 \_sdef{\_ur: /}{\_urlskip/\_urlbskip}
162 \_sdef{\_ur: .}{\_urlskip?\_urlbskip}
163 \_sdef{\_ur: ?}{\_urlskip?\_urlbskip}
164 \_sdef{\_ur: =}{\_urlskip=\_urlbskip}
165 \_sdef{\_ur: -}{\_urlskip-\_urlbskip}
166 \_sdef{\_ur: &}{\_urlskip\_char`&\_urlbskip}
167 \_sdef{\_ur: gb|}{\_urlgskip}
168
169 \_def\urlfont{\_tt} % url font
170 \_def\urlxskip{\_penalty9990\_hskip0pt plus0.03em\_relax} % skip between normal characters
171 \_def\urlskip{\_null\_nobreak\_hskip0pt plus0.1em\_relax} % skip before :// / . ? = - &
172 \_def\urlbskip{\_penalty100\_hskip0pt plus0.1em\_relax} % skip after :// / . ? = - &
173 \_def\urlgskip{\_penalty-500\_relax} % "goodbreak" penalty generated by \|
174
175 \_public \url ;

```

## 2.24 Making table of contents

maketoc.opm

```

3 \_codedecl \maketoc {Macros for maketoc <2021-07-18>} % preloaded in format

```

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

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

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

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

maketoc.opm

```

26 \_def\_toclist{}
27 \_newifi \_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}

```

`\_tocline` `{\langle level \rangle}{\langle type \rangle}{\langle number \rangle}{\langle o-title \rangle}{\langle title \rangle}{\langle gpageno \rangle}{\langle pageno \rangle}` 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##1{\_ilink[pg:#6]{\_numprint{##1}}}%
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
77   \_bf\_llaptoclink{#1}{#2}\_nobreak\_hfill \_pgn{#3}\_tocpar}
78 \_sdef{\_tocl:2}#1#2#3{\_llaptoclink{#1}{#2}\_tocdotfill \_pgn{#3}\_tocpar}
79 \_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 rllapped `⟨pageno⟩`.
- `\_pgn{⟨pageno⟩}` creates `⟨pageno⟩` as link to real `⟨gpage⟩` saved in #6 of `\_tocline`. This is temporarily defined in the `\_tocline`.

```

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

```

maketoc.opm

If you want a special formatting of TOC with adding more special lines (no 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}{⟨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

```
128 \def\maketoc{\par \ifx\toclist\empty
129   \opwarning{\noexpand\maketoc -- data unavailable, TeX me again}\openref
130   \incr\unresolvedrefs
131 \else \begingroup
132   \tocrefnum=0 \penalty11333
133   \the\regtoc \toclist
134 \endgroup \fi
135 }
```

`\regmacro` appends its parameters to `\regtoc`, `\regmark` and `\regoul`. These token lists are used in `\maketoc`, `\begoutput` and `\pdfunidef`.

maketoc.opm

```
143 \newtoks \regtoc \newtoks \regmark \newtoks \regoul
144
145 \def\regmacro #1#2#3{%
146   \toksapp\regtoc{#1}\toksapp\regmark{#2}\toksapp\regoul{#3}%
147 }
148 \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` `{<text>}` inserts one outline with zero direct descendants. It creates a link destination of the type `oul:<num>` into the document (where `\insertoutline` is used) and the link itself is created too in the outline.

outlines.opm

```
3 \codedec1 \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\count#1 by1 \sxddef{#1}{\the\count#1}%
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{\tmp}\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, M. (Even the whitespace would be OK in a PDF file, because it should be ignored by PDF viewers, but LuaT<sub>E</sub>X doesn’t allow it.) This conversion is done by the expandable “pseudoprimitive” `\pdfstring`.

```
3 \_codedecl \pdfunidef {PDFUnicode strings for outlines <2026-04-21>} % preloaded in format
```

`\pdfunidef\macro{<text>}` defines `\macro` as `<text>` converted to Big Endian UTF-16 and enclosed to `<>`. Example of usage: `\pdfunidef\infoauthor{Petr Olšák} \pdfinfo{/Author \infoauthor}`.

`\pdfunidef` 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 `\unexpanded\ea{\scantextokens{.}}` and second `\edef` expands the parameter according to current values on selected macros from `\_regoul`. Then `\_removeoutmath` converts `..$x^2$..` to `..x^2..`, i.e. removes dollars. Then `\_removeoutbraces` converts `..{x}..` to `..x..`. Finally, the `<text>` is detokenized, and `\pdfstring` is applied.

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.

```
22 \_def\pdfunidef#1#2{%
23   \_begingroup
24     \_catcodetable\optexcatcodes \_edef{"'}\_edef{''}%
25     \_the\_regoul \_relax % \_regmacro alternatives of logos etc.
26     \_ifx\_savedttchar\_undefined \_def#1{\_unexpanded\_ea{\_scantextokens{#2}}}%
27     \_else \_lccode\;=\_savedttchar \_lowercase{\_prepinverb#1;}{#2}\fi
28     \_edef#1{#1}%
29     \_escapechar=-1
30     \_edef#1{#1\_empty}%
31     \_escapechar='\
32     \_ea\_edef \_ea#1\_ea{\_ea\_removeoutmath #1$\_fin$}% $x$ -> x
33     \_ea\_edef \_ea#1\_ea{\_ea\_removeoutbraces #1{\_fin$}% {x} -> x
34     \_edef#1{\_detokenize\_ea{#1}}%
35     \_ea
36   \_endgroup
37   \_ea\_edef\_ea#1\_ea{\_pdfstring\_ea{#1}}
38 }
39 \_def\_removeoutbraces #1#{#1\_removeoutbracesA}
40 \_def\_removeoutbracesA #1{\_ifx\_fin#1\_else #1\_ea\_removeoutbraces\_fi}
41 \_def\_removeoutmath #1$#2$#{#1\_ifx\_fin#2\_else #2\_ea\_removeoutmath\_fi}
```

The `\_prepinverb<macro><separator>{<text>}`, e.g. `\_prepinverb\tmpb|{aaa |bbb| cccc |dd| ee}` does `\def\tmpb{<us>{aaa }bbb<us>{ cccc }dd<us>{ ee}}` where `<us>` is `\unexpanded\scantextokens`. 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 in-line verbatim.

```
52 \_def\_prepinverb#1#2#3{\_def#1{%
53   \_def\_dotmpb ##1#2##2{\_addto#1{\_unexpanded\_ea{\_scantextokens{##1}}}%
54   \_ifx\_fin##2\_else\_ea\_dotmpbA\_ea##2\_fi}%
55   \_def\_dotmpbA ##1#2{\_addto#1{##1}\_dotmpb}%
56   \_dotmpb#3#2\_fin
57 }
```

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

```
65 \_regmacro {}{\_let\em=\_empty \_let\rm=\_empty \_let\bf=\_empty
66   \_let\it=\_empty \_let\bi=\_empty \_let\tt=\_empty \_let\/=\_empty
67   \_let~=\_space
68 }
69 \public \pdfunidef ;
```

## 2.26 Chapters, sections, subsections

```
3 \_codedecl \chap {Titles, chapters, sections, subsections <2026-03-15>} % preloaded in format
```

We are using scaled fonts for titles `\_titfont`, `\_chapfont`, `\_secfont` and `\_seccfont`. They are scaled from main fonts size of the document, which is declared by first `\typosize[<fo-size>/<b-size>]` command.

```
13 \_def \_titfont {\_scalemain\_typoscale[\_magstep4/\_magstep5]\_boldify}
14 \_def \_chapfont {\_scalemain\_typoscale[\_magstep3/\_magstep3]\_boldify}
15 \_def \_secfont {\_scalemain\_typoscale[\_magstep2/\_magstep2]\_boldify}
16 \_def \_seccfont {\_scalemain\_typoscale[\_magstep1/\_magstep1]\_boldify}
```

The `\tit` macro is defined 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 plusifill \_rightskip=\_leftskip % centering
27     \_titfont \_noindent \_scantextokens{#1}\_par}%
28     \_nobreak\_bigskip
29 }
30 \def\_tit{\_scantoel\_prnttit}
31 \sdef{eol:tit}{\_prnttit} % enables \bracedparam\tit{title}
32
33 \_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 `<pre><ref-num><post>`. The `<ref-num>` is `\_thechapnum` or `\_thesecnum` 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 `\_nbpar` 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.

sections.opm

```

73 \def\_printchap #1{\_vfill\_supereject \_prevdepth=0pt
74   \_vglue\_medskipamount % shifted by topkip+medskipamount
75   {\_chapfont \_noindent \_mtext{chap} \_printrefnum[@]\_par
76     \_nobreak\_smallskip
77     \_noindent \_raggedright #1\_nbpar}\_mark{ }%
78     \_nobreak \_belowtitle{\_bigskip}%
79     \_firstnoindent
80 }
81 \def\_printsec#1{\_par
82   \_abovetitle{\_penalty-151}\_bigskip
83   {\_seccfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbpar}\_insertmark{#1}%
84   \_nobreak \_belowtitle{\_medskip}%
85   \_firstnoindent
86 }
87 \def\_printsecc#1{\_par
88   \_abovetitle{\_penalty-101}{\_medskip\_smallskip}
89   {\_seccfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbpar}%
90   \_nobreak \_belowtitle{\_medskip}%
91   \_firstnoindent
92 }

```

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](#))

```

106 \_newcount\_sectionlevel
107 \_def \_secinfo {\_ifcase \_sectionlevel
108   part\_or chap\_or sec\_or secc\_or seccc\_fi
109 }

```

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  $\langle num \rangle.\langle num \rangle.\langle num \rangle$ .

```

137 \_newcount \_chapnum % chapters
138 \_newcount \_secnum % sections
139 \_newcount \_seccnum % subsections
140 \_newcount \_tnum % table numbers
141 \_newcount \_fnum % figure numbers
142 \_newcount \_dnum % numbered display maths
143
144 \_def \_chapx {\_secx \_secnum=0 \_lfnotenum=0 }
145 \_def \_secx {\_seccx \_seccnum=0 \_tnum=0 \_fnum=0 \_dnum=0 \_resetABCDE }
146 \_def \_seccx {}
147
148 \_def \_thechapnum {\_the\_chapnum}
149 \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum}
150 \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum.\_the\_seccnum}
151 \_def \_thetnum {\_othe\_chapnum.\_othe\_secnum.\_the\_tnum}
152 \_def \_thefnum {\_othe\_chapnum.\_othe\_secnum.\_the\_fnum}
153 \_def \_thednum {\_the\_dnum}
154
155 \_def\_othe #1.{\_ifnum#1>0 \_the#1.\_fi}

```

sections.opm

The `\notoc` and `\nonum` prefixes are implemented by internal `\_ifnotoc` and `\_ifnonum`. They are reset after each chapter/section/subsection by the `\_resetnonumnotoc` macro.

```

163 \_newifi \_ifnotoc \_notocfalse \_def\_notoc {\_global\_notoctrue}
164 \_newifi \_ifnonum \_nonumfalse \_def\_nonum {\_global\_nonumtrue}
165 \_def \_resetnonumnotoc{\_global\_notocfalse \_global\_nonumfalse}
166 \_public \notoc \nonum ;

```

sections.opm

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  $[\langle label \rangle]$ , 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. Because the first token of the title parameter is tokenized already (due to `\optdef`), we have to use `\_scantoeolX` instead `\scantoeol`. This solves the problem `\sec {text}`.

```

180 \_optdef\_chap[]{\_trylabel \_scantoeolX\_inchap}
181 \_optdef\_sec[]{\_trylabel \_scantoeolX\_insec}
182 \_optdef\_secc[]{\_trylabel \_scantoeolX\_insecc}
183 \_def\_trylabel{\_istokempty\_opt\_iffalse \_label[\_the\_opt]\_fi}
184 \_def\_scantoeolX#1{\_isnextchar\_bgroup{\_ea\_scantoeol\_ea#1\_string}\_scantoeol#1}
185
186 \_sdef{eol:chap}{\_inchap} % enables \bracedparam\chap{title}

```

sections.opm



```

187 \_sdef{\_eol:sec}{\_insec} % enables \bracedparam\sec{title}
188 \_sdef{\_eol:secc}{\_insecc} % enables \bracedparam\secc{title}
189
190 \_def\_inchap #1{\_par \_sectionlevel=1
191 \_def \_savedtitle {#1}% saved to .ref file
192 \_ifnonum \_else {\_globaldefs=1 \_incr\_chapnum \_chapx}\_fi
193 \_edef \_therefnun {\_ifnonum \_space \_else \_thechapnum \_fi}%
194 \_printchap{\_scantextokens{#1}}%
195 \_resetnonumnotoc
196 }
197 \_def\_insec #1{\_par \_sectionlevel=2
198 \_def \_savedtitle {#1}% saved to .ref file
199 \_ifnonum \_else {\_globaldefs=1 \_incr\_secnum \_seccx}\_fi
200 \_edef \_therefnun {\_ifnonum \_space \_else \_theseccnum \_fi}%
201 \_printsec{\_scantextokens{#1}}%
202 \_resetnonumnotoc
203 }
204 \_def\_insecc #1{\_par \_sectionlevel=3
205 \_def \_savedtitle {#1}% saved to .ref file
206 \_ifnonum \_else {\_globaldefs=1 \_incr\_seccnum \_seccx}\_fi
207 \_edef \_therefnun {\_ifnonum \_space \_else \_theseccnum \_fi}%
208 \_printsecc{\_scantextokens{#1}}%
209 \_resetnonumnotoc
210 }
211 \_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  $\text{\LaTeX}$  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

```

222 \_def \_printrefnum [#1@#2]{\_leavevmode % we must be in horizontal mode
223 \_ifnonum \_else #1\_numprint\_therefnun #2\_fi
224 \_wlabel \_therefnun % references, if \label[<label>]` is declared
225 \_ifnotoc \_else \_incr \_tocrefnum
226 \_dest[toc:\_the\_tocrefnum]%
227 \_ewref\_Xtoc{{\_the\_sectionlevel}{\_secinfo}%
228 \_therefnun}{\_theoutline}\_detokenize\_ea{\_savedtitle}}%
229 \_fi
230 \_gdef\_theoutline{%
231 }

```

`\thisoutline{⟨text⟩}` saves text to the `\_theoutline` macro. `\_printrefnum` uses it and removes it.

sections.opm

```

238 \_def\_theoutline{}
239 \_def\_thisoutline#1{\_gdef\_theoutline{#1}}
240 \_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

```

256 \_newskip \_savedtitleskip
257 \_newskip \_savedlastskip
258 \_def\_abovetitle #1#2{\_savedlastskip=\_lastskip % <whatever-skip>
259 \_ifdim\_lastskip>\_zo \_vskip-\_lastskip \_fi
260 \_ifnum\_lastpenalty=11333 \_vskip-\_savedtitleskip \_else #1\_fi
261 \_ifdim\_savedlastskip>\_zo \_setbox0=\vbox{#2\_global\_tmpdim=\_lastskip}%
262 \_else \_tmpdim=\_maxdimen \_fi
263 \_ifdim\_savedlastskip>\_tmpdim \_vskip\_savedlastskip \_else #2\_fi
264 }
265 \_def\_belowtitle #1{#1\_global\_savedtitleskip=\_lastskip \_penalty11333 }

```

`\nbpar` sets `\interlinepenaty` value. `\nl` is “new line” in the text (or titles), but space in toc or headlines or outlines.

```

272 \def\nbpar{{\interlinepenalty=10000\endgraf}}
273
274 \protected\def\nl{\unskip\hfil\break}
275 \regmacro {\def\nl{\unskip\space}} {\def\nl{\unskip\space}} {\def\nl{ }}
276 \regmacro {\def\nl{\unskip\space}} {\def\nl{\unskip\space}} {\def\nl{ }}
277
278 \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`.

```

287 \def \firstnoindent {\global\everypar={\wipeepar \setbox7=\lastbox}}
288 \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.

```

303 \def\insertmark#1{\mark{{\ifnonum\else\therefnum\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. See also [issue 100](#).

```

\addto\chaptx {\globaldefs=0 \vfil\break % headline of previous chapter is printed
\edef\runningchap {\thechapnum: \unexpanded\ea{\savedtitle}}}
\def \formathead #1#2{\isempty{#1}\iffalse #1: #2\fi}
\headline = {%
\ifodd \pageno
\hfil \ea\formathead\firstmark{}{}}%
\else
\ifx\runningchap\undefined \else Chapter \runningchap \fi \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. The variant `\eol:secl` is defined to enable `\bracedparam\secl<number> {<title-text>}`.

```

332 \def\secl{\let\secle=\ea \afterassignment\secla \sectionlevel=}
333 \sdef{eol:secl}{\def\secle{\ea\bracedparam\ea}\afterassignment\secla \sectionlevel=}
334 \def\secla{\ifcase\sectionlevel
335 \or \secle\chap \or \secle\sec \or \secle\secc \else \ea \seclp\fi}
336 \eoldef\seclp#1{\par \ifnum\lastpenalty=0 \removelastskip\medskip\fi
337 \noindent{\bf #1}\vadjust{\nobreak}\nl\ignorepars}
338 \def\ignorepars{\isnextchar\par{\ignoresecond\ignorepars}{}}
339
340 \public \secl ;

```

The `\caption/<letter>` increases `\<letter>num` counter, edefines `\thecapnum` as `\the<letter>num` and defines `\thecapttitle` as language-dependent word using `\mtext`, declares default format by `\captionformat{<letter>}` and runs the `\everycaption<letter>` tokens register. The two groups opened by `\caption` are finalized by first `\par` from an empty line or from `\vskip`, `\cskip` or from `\endinsert`. If a `}` occurs first then `\par` from `\aftergroup` is processed. 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.

```

357 \def\caption/#1{\def\tmpa{#1}\nospaceafter \capA}
358 \optdef\capA []{\trylabel \incaption}
359 \def\incaption {\bgroup
360   \ifcsname _\tmpa num\endcsname \ea\incr \csname _\tmpa num\endcsname
361   \else \opwarning{Unknown caption /\tmpa}\fi
362   \edef\thecapnum {\csname _the\tmpa num\endcsname}%
363   \edef\thecapttitle{\mtext{\tmpa}}%
364   \ea\captionformat\ea{\tmpa}%
365   \ea\the \csname _everycaption\tmpa\endcsname
366   \def\par{\ifhmode\nbpar\egroup\egroup\fi}%
367   \ifx\par\endgraf \let\par=\par \fi
368   \bgroup \aftergroup\par
369   \cs{printcaption\tmpa}%
370 }
371 \def \cskip {\par\nobreak\medskip} % space between caption and the object
372
373 \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). They can use:

- `\thecapttitle` ... expands to the word Table or Figure (depending on the current language).
- `\thecapnum` ... expands to `\the<letter>num` (caption number).

The macro `\printcaptiont` (or `f`) is processed inside group and the `\par` can be run after this group. If you want to re-define formatting parameters for `\par`, do this in the macro `\captionformat`. 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.

```

395 \def \printcaptiont {%
396   \noindent \wlabel\thecapnum {\bf\thecapttitle~\thecapnum}%
397   \futurelet\next\captionsep
398 }
399 \def\captionsep{\ifx\next.\ea\bfnext \else\ifx\next:\ea\ea\ea\bfnext
400   \else \enspace \fi\fi}
401 \def\bfnext#1{\bf#1}
402 \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 format of the `\caption` text is given by the `\captionformat{<caption-letter>}` macro. The default format for `t` and `f` is a paragraph in block narrower by `\iindent` and with the last line is centered. This setting is done by the `\narrowlastlinecentered` macro.

```

421 \def\captionformat#1{\narrowlastlinecentered\iindent}
422 \def\narrowlastlinecentered#1{%
423   \leftskip=#1plus1fil
424   \rightskip=#1plus-1fil
425   \parfillskip=0pt plus2fil\relax
426 }

```

`\eqmark` is processed in display mode (we add `\eqno` primitive) or in internal mode when `\equaligno` is used (we don't add `\eqno`).

```

433 \optdef\eqmark []{\trylabel \ineqmark}
434 \def\ineqmark{\incr\dnum
435   \ifinner\else\eqno \fi
436   \wlabel\thednum \hbox{\numprint\thednum}%
437 }
438 \public \eqmark ;

```

The `\numberedpar`  $\langle letter \rangle \{ \langle name \rangle \}$  is implemented here.

sections.opm

```

444 \_newcount\_counterA \_newcount\_counterB \_newcount\_counterC
445 \_newcount\_counterD \_newcount\_counterE
446
447 \_def\_resetABCDE {\_counterA=0 \_counterB=0 \_counterC=0 \_counterD=0 \_counterE=0 }
448
449 \_def \_theAnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterA}
450 \_def \_theBnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterB}
451 \_def \_theCnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterC}
452 \_def \_theDnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterD}
453 \_def \_theEnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterE}
454
455 \_def\_numberedpar#1#2{\_ea \_incr \_csname\_counter#1\_endcsname
456 \_def\_tmpa{#1}\_def\_tmpb{#2}\_numberedparparam}
457 \_optdef\_numberedparparam[]{\%
458 \_ea \_printnumberedpar \_csname\_the\_tmpa num\_ea\_endcsname\_ea{\_tmpb}}
459
460 \_public \_numberedpar ;

```

The `\_printnumberedpar` `\theXnum`  $\{ \langle name \rangle \}$  opens numbered paragraph and prints it. The optional parameter is in `\_the\_opt`. You can re-define it if you need another design.

`\_printnumberedpar` needs not to be re-defined if you only want to print Theorems in italic and to insert vertical skips (for example). You can do this by the following code:

```

\def\theorem {\medskip\bgroup\it \numberedpar A{Theorem}}
\def\endtheorem {\par\egroup\medskip}

\theorem Let  $M$  be... \endtheorem

```

sections.opm

```

478 \_def \_printnumberedpar #1#2{\_par
479 \_noindent\_wlabel #1%
480 {\_bf #2 \_numprint{#1}\_istokseempty\_opt\_iffalse \_space \_the\_opt \_fi.}\_space
481 \_ignorespaces
482 }

```

## 2.27 Lists, items

lists.opm

```

3 \_codedecl \begitems {Lists: begitems, enditems <2023-10-20>} % preloaded in format

```

`\_aboveliskip` is used above the list of items,

`\_belowliskip` is used below the list of items,

`\_setlistskip` sets the skip dependent on the current level of items,

`\_listskipab` is `\ilistskipamount` or `\olistskipamount`.

lists.opm

```

12 \_def\_aboveliskip {\_removelastskip \_penalty-100 \_vskip\_listskipab}
13 \_def\_belowliskip {\_penalty-200 \_vskip\_listskipab}
14 \_newskip\_listskipab
15
16 \_def\_setlistskip {\%
17 \_ifnum \_ilevel = 1 \_listskipab = \olistskipamount \_relax
18 \_else \_listskipab = \ilistskipamount \_relax
19 \_fi}

```

The `\itemnum` is locally reset to zero in each group declared by `\begitems`. So nested lists are numbered independently. Users can set initial value of `\itemnum` to another value after `\beitems` if they want. Each level of nested lists is indented by the new `\iindent` from left. The default item mark is `\_printitem`.

The `\begitems` runs `\_aboveliskip` only if we are not near below a title, where a vertical skip is placed already and where the `\penalty` 11333 is. It activates `*` and defines it as `\_startitem`.

The `\enditems` runs `\_isnextchar\_par{\\_noindent}` thus the next paragraph is without indentation if there is no empty line between the list and this paragraph (it is similar behavior as after display math).

```

38 \_newcount\_itemnum \_itemnum=0
39 \_newtoks\_printitem
40
41 \_def\_begitems{\_par
42   \_bgroup
43   \_advance \_ilevel by1
44   \_setlistskip
45   \_ifnum\_lastpenalty<10000 \_aboveliskip \_fi
46   \_itemnum=0 \_edef*\_relax\_ifmode*\_else\_ea\_startitem\_fi}
47   \_advance\_leftskip by\_iindent
48   \_printitem=\_defaultitem
49   \_the\_everylist \_relax
50 }
51 \_def\_enditems{\_par\_belowliskip\_egroup \_isnextchar\_par{}\{\_noindent}}
52
53 \_def\_startitem{\_par \_ifnum\_itemnum>0 \_vskip\_itemskipamount \_fi
54   \_advance\_itemnum by1
55   \_the\_everyitem \_noindent\_llap{\_the\_printitem}\_ignorespaces
56 }
57 \_public \_begitems \_enditems \_itemnum ;

```

`\novspaces` sets `\listskipab` and `\itemskipamount` to 0pt. Moreover, it deactivates `\setlistskip` (for inner lists).

```

64 \_def\_novspaces {\_removelastskip
65   \_listskipab=\_zoskip \_itemskipamount=\_zoskip \_let\_setlistskip=\_relax}
66 \_public \novspaces ;

```

Various item marks are saved in `\_item:<letter>` macros. You can re-define then or define more such macros. The `\style <letter>` does `\_printitem={\_item:<letter>}`. More exactly: `\begitems` does `\_printitem=\_defaultitem` first, then `\style <letter>` does `\_printitem={\_item:<letter>}` when it is used and finally, `\startitem` alias `*` uses `\_printitem`.

```

77 \_def\_style#1{%
78   \_ifcsname\_item:#1\_endcsname \_printitem=\_ea{\_csname\_item:#1\_endcsname}%
79   \_else \_printitem=\_defaultitem \_fi
80 }
81 \_sdef{\_item:o}{\_raise.4ex\_hbox{\$\_scriptscriptstyle\_bullet$} }
82 \_sdef{\_item:-}{- }
83 \_sdef{\_item:n}{\_the\_itemnum. }
84 \_sdef{\_item:N}{\_the\_itemnum) }
85 \_sdef{\_item:i}{(\_romannumeral\_itemnum) }
86 \_sdef{\_item:I}{\_uppercase\_ea{\_romannumeral\_itemnum}\_kern.5em}
87 \_sdef{\_item:a}{\_athe\_itemnum) }
88 \_sdef{\_item:A}{\_uppercase\_ea{\_athe\_itemnum)) }
89 \_sdef{\_item:x}{\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
90 \_sdef{\_item:X}{\_raise.2ex\_fullrectangle{1ex}\_kern.5em}
91 \_sdef{\_item:d}{\_aftergroup\_dword}
92 \_def\_dword#1#2{{\_bf #2 }\_ignorespaces} % #1 is \_ignorespaces from \_startitem

```

`\_athe{<num>}` returns the `<num>`'s lowercase letter from the alphabet.

`\_fullrectangle{<dimen>}` prints full rectangle with given `<dimen>`.

```

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.

```

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 <2026-05-20>} % 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<char>` saves original catcode of previously declared `<char>` (if such character was declared) using `\_savedttchar` and `\_savedttcharc` values. Then new such values are stored. The declared character is activated by `\_adev` as a macro (active character) which opens a group, does `\_setverb` and other settings and reads its parameter until second the same character. This is done by the `\_readverb` macro. Finally, it prints scanned `<text>` by `\_printinverbatim` and closes group. Suppose that `\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   \_adev{#1}{\_begingroup \_setverb \_adev{ }{\_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.

A user can write `\begtt \ttline=0` and we want to interpret this setting locally too. This is the reason of creating the `\_restorettline` macro before the argument #1 of `\_begtti` is processed and using the trick `\ifnum\ttline=\maxdimen` after this processing. See the code below.

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.

The `^^J` is appended to verbatim data because we need to be sure that the data are finished by this character. When `\endtt` is preceded by spaces then we need to close these spaces by `^^J` and such line is not printed due to a trick used in `\_printverb`.

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.

```
131 \_def\begtt{\_par \_begingroup \_edef\%##1\_relax{\_relax}\_begtti}
132 \_eoldef \_begtti#1{\_wipepar \_setxsize
133 \_vskip\_parskip \_ttskip
134 \_setverb
135 \_edef{ }\_dsp}\_edef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
136 \_def\t{\_hskip \_dimexpr\_tabspace em/2\_relax}%
137 \_protrudechars=0 % disable protrusion
138 \_xdef\_restorettline{\_global\_ttline=\_the\_ttline\_relax}\_ttline=\_maxdimen
139 \_the\_everytt \_relax #1\_relax
140 \_ifnum\_ttline=\_maxdimen \_restorettline \_else \_aftergroup\_restorettline \_fi
141 \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
142 \_def\_testcommentchars##1\_iftrue{\_iffalse}\_let\_hicomments=\_relax
143 \_ttfont \_savemathsb \_endlinechar=^^J
144 \_startverb
145 }
146 \_ea\_def\_ea\_startverb \_ea#\_ea1\_csstring\endtt#2^^J{%
147 \_prepareverbdata\_tmpb{#1^^J}%
148 \_ea\_printverb \_tmpb\_fin
149 \_par \_restoremathsb
150 \_endgroup \_ttskip
151 \_isnextchar\_par{}{\_noindent}%
152 }
153 \_def\_prepareverbdata#1#2{\_def#1{#2}}
```

The `\_printverb` macro calls `\_printverblines{<line>}` repeatedly to each scanned line of verbatim text. The `\_printverb` is used from `\begtt... \endtt` and from `\verbinp` 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`.

`\_putttenpalty` puts `\_ttenpalty` before second and next lines, but not before first line in each `\begtt... \endtt` environment.

`\_nograd{<line\_number>}` allows to select only code but not `<line\_number>`s by mouse in several PDF viewers.

The `\_ttline` is increased here in the `\_printverb` macro because of comments-blocks: the `\_printverblines` is not processed in comments-blocks but we need to count the `\_ttline`.

```

181 \def\printverb #1^^J#2{%
182   \ifx\printverblinenum\relax \else \incr\ttline \fi
183   \testcommentchars #1\relax\relax\relax
184   \iftrue
185     \ifx\fin#2\printcomments\fi
186   \else
187     \ifx\vcomments\empty\else \printcomments \def\vcomments{}\fi
188     \ifx\fin#2%
189       \bgroup \adef{ }{}\def\t{}% if the last line is empty, we don't print it
190       \ifcat&#1&\egroup \ifx\printverblinenum\relax \else \decr\ttline \fi
191       \else\egroup \printverblines{#1}\fi
192     \else
193       \printverblines{#1}%
194     \fi
195   \fi
196   \unless\ifx\fin#2\afterfi{\printverb#2}\fi
197 }
198 \def\printverblines#1{\putttpenalty \indent \printverblinenum \kern\ttshift #1\par}
199 \def\initverblinenum{\tenrm \thefontscale[700]\ea\ea\let\ea\sevenrm\the\font}
200 \def\printverblinenum{\llap{\sevenrm \nograd\the\ttline}\kern.9em}
201 \def\putttpenalty{\def\putttpenalty{\penalty\ttpenalty}}
202 \def\nograd#1{\pdfliteral page{/Span<</ActualText<>>BDC}#1\pdfliteral page{EMC}}

```

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.

```

218 \def\verbinp #1(#2) #3 {\par \def\tmpa{#3}%
219   \def\tmpb{#1}% cmds used in local group
220   \ifx\vifilename\tmpa \else
221     \openin\vifile={#3}%
222     \global\viline=0 \glet\vifilename=\tmpa
223     \ifeof\vifile
224       \opwarning{\string\verbinp: file "#3" unable to read}
225       \ea\ea\ea\skiptorelax
226     \fi
227   \fi
228   \viscanparameter #2+\relax
229 }
230 \def\skiptorelax#1\relax{}
231
232 \def \viscanparameter #1+#2\relax{%
233   \if$#2$\viscanminus(#1)\else \viscanplus(#1+#2)\fi
234 }
235 \def \viscanplus(#1+#2){%
236   \if$#1$\tmpnum=\viline
237   \else \ifnum#1<0 \tmpnum=\viline \advance\tmpnum by-#1
238     \else \tmpnum=#1
239     \advance\tmpnum by-1
240     \ifnum\tmpnum<0 \tmpnum=0 \fi % (0+13) = (1+13)
241   \fi \fi
242   \edef\vinolines{\the\tmpnum}%
243   \if$#2$\def\vidolines{0}\else\edef\vidolines{#2}\fi
244   \doverbinp
245 }
246 \def \viscanminus(#1-#2){%
247   \if$#1$\tmpnum=0
248   \else \tmpnum=#1 \advance\tmpnum by-1 \fi
249   \ifnum\tmpnum<0 \tmpnum=0 \fi % (0-13) = (1-13)
250   \edef\vinolines{\the\tmpnum}%
251   \if$#2$\tmpnum=0
252   \else \tmpnum=#2 \advance\tmpnum by-\vinolines \fi
253   \edef\vidolines{\the\tmpnum}%

```

```

254 \doverbinput
255 }
256 \def\doverbinput{%
257   \tmpnum=\vinolines
258   \advance\tmpnum by-\viline
259   \ifnum\tmpnum<0
260     \openin\vfiling=\vifilingname}%
261     \global\viline=0
262   \else
263     \edef\vinolines{\the\tmpnum}%
264   \fi
265   \vskip\parskip \ttskip \wipepar \setxsize
266   \begingroup
267   \setverb \adef{ }{\dsp}\adef^^I{\t}\parindent=\ttindent \parskip=Opt
268   \def\t{\hskip \dimexpr\tabspace em/2\relax}%
269   \protrudechars=0 % disable protrusion
270   \xdef\restoretline{\global\ttline=\the\ttline\relax}\ttline=\maxdimen
271   \the\everytt\relax \tmpb\relax
272   \ifnum\ttline=\maxdimen \restoretline \else \aftergroup\restoretline \fi
273   \ifnum\ttline<-1 \let\printverblinenum=\relax \else \initverblinenum \fi
274   \ttfont \savemathsb \endlinechar=^^J \tmpnum=0
275   \loop \ifeof\vfiling \tmpnum=\vinolines\space \fi
276     \ifnum\tmpnum<\vinolines\space
277       \vireadline \advance\tmpnum by1 \repeat %% skip lines
278     \edef\ttlinesave{\global\ttline=\the\ttline}%
279     \ifnum\ttline=-1 \ttline=\viline \else \let\ttlinesave=\relax \fi
280     \tmpnum=0 \def\tmpb{}%
281     \ifnum\vidolines=0 \tmpnum=-1 \fi
282     \ifeof\vfiling \tmpnum=\vidolines\space \fi
283     \loop \ifnum\tmpnum<\vidolines\space
284       \vireadline
285       \ifnum\vidolines=0 \else\advance\tmpnum by1 \fi
286       \ifeof\vfiling \tmpnum=\vidolines\space \else \visaveline \fi %% save line
287     \repeat
288     \ea\prepareverbdata \ea \tmpb\ea{\tmpb^^J}%
289     \catcode\ =10 \catcode\ =9 % used in \commentchars comments
290     \ea\printverb \tmpb\fin
291     \ttlinesave
292     \par \restoremathsb
293   \endgroup
294   \ttskip
295   \isnextchar\par{{}\noindent}%
296 }
297 \def\vireadline{\read\vfiling to \tmp \incr\viline }
298 \def\visaveline{\ea\addto\ea\tmpb\ea{\tmp}}
299
300 \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

```

310 \def\savemathsb{\ifmathsb \mathsboff \def\restoremathsb{\mathsb}\fi}
311 \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 to 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. 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.

```

363 \_def\_vcomments{}
364 \_let\_hicomments=\_relax
365
366 \_def\_commentchars#1#2{%
367   \_def\_testcommentchars ##1##2##3\_relax ##4\_iftrue{\_ifnum % not closed in this macro
368     \_ifx #1##1\_ifx#2##21\_fi\_fi 0>0
369     \_ifx\_relax##3\_relax \_addto\_vcomments{\_endgraf}% empty comment=\_enfgraf
370     \_else \_addto\_vcomments{##3 }\_fi}%
371   \_def\_hicomments{\_replfromto{\b\n#1#2}{^^J}{\w{#1#2###1}^^J}}% used in \_hisyntax
372 }
373 \_def\_testcommentchars #1\_iftrue{\_iffalse} % default value of \_testcommentchar
374 \_def\_printcomments{\_ttskip
375   {\_catcodetable0 \_rm \_everypar={}%
376     \_noindent \_ignorespaces \_scantextokens\_eaf\_vcomments}\_par}%
377   \_ttskip
378 }
379 \_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.

```

390 \_def \_visiblesp{\_ifx\_initunifonts\_relax \_def\_dsp{\_char9251 }%
391   \_else \_def\_dsp{\_char32 }\_fi}
392 \_let\_dsp=\_ % primitive "direct space"
393
394 \_public \visiblesp ;
```

## 2.28.2 Listings with syntax highlighting

The user can write

```
\begtt \_hisyntax{C}
...
\_endtt
```

to colorize the code using C syntax. The user can also write `\everytt={\_hisyntax{C}}` to have all verbatim listings colorized.

`\_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:

```

3 \_codedecl \_hisyntaxc {Syntax highlighting for C sources <2023-03-02>}
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     {alignas}{alignof}{auto}{bool}{break}{case}{char}{const}%
29     {constexpr}{continue}{default}{do}{double}{else}{enum}{extern}%
30     {false}{float}{for}{goto}{if}{inline}{int}{long}{nullptr}%
31     {register}{restrict}{return}{short}{signed}{sizeof}{static}%
32     {static_assert}{struct}{switch}{thread_local}{true}{typedef}%
33     {typeof}{typeof_unqual}{union}{unsigned}{void}{volatile}{while}%
34     {_Alignas}{_Alignof}{_Atomic}{_BitInt}{_Bool}{_Complex}%
35     {_Decimal128}{_Decimal32}{_Decimal64}{_Generic}{_Imaginary}%
36     {_Noreturn}{_Static_assert}{_Thread_local}
37     \_do {\_replthis{\n#1\n}{\z K{#1}}}
38   \_replthis{.}{\n.\n}                                     % numbers
39   \_foreach 0123456789
40     \_do {\_replfromto{\n#1}{\n}{\c#1##1\e}}
41   \_replthis{\e.\c}{.}
42   \_replthis{\e.\n}{.\e}
43   \_replthis{\n.\c}{\c.}
44   \_replthis{e\e\o+\c}{e+}\_replthis{e\e\o-\c}{e-}
45   \_replthis{E\e\o+\c}{E+}\_replthis{E\e\o-\c}{E-}
46   \_def\o#1{\z O{#1}}
47   \_def\c#1\e{\z N{#1}}
48 }

```

OpTeX provides `hisyntax-{c, lua, python, tex, html, kt}.opm` files. You can take inspiration from these files and declare more languages.

Users can re-declare default colors by `\_hicolors={\langle list of color declarations \rangle}`. This value has precedence over `\_hicolors<name>` values declared in the `hicolors-<name>.opm` file. For example `\_hicolors={\_hicolor S \Brown}` causes all strings in brown color.

Another way to set non-default colors is to declare `\newtoks\_hicolors<name>` (without the `_` prefix) and set the color palette there. It has precedence before `\_hicolors<name>` (with the `_` prefix) declared in the `hicolors-<name>.opm` file. You must re-declare all colors used in the corresponding `hisyntax-<name>.opm` file.

### Notes for hi-syntax macro writers

The file `hisyntax-<name>.opm` is read only once and in a TeX group. If there are definitions then they must be declared as global.

The file `hisyntax-<name>.opm` must (globally) declare `\_hisyntax<name>` token list where the action over verbatim text is declared typically by using the `\replfromto` or `\replthis` macros.

The verbatim text is prepared by the *pre-processing phase*, then `\_hisyntax<name>` is applied and then the *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` is the pattern for matching 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. Both are removed in the post-processing phase.

Special macros are working only in a group when processing the verbatim text.

- `\n` represents nothing but it should be used as a boundary of words as mentioned above.
- `\t` represents a tabulator. It is prepared as `\n\t\n` because it can be at the boundary word boundary.
- `\x <letter>{<text>}` can be used as replacing text. Consider the example

```
\replfromto{/*}{*/}{\x C{/*#1*/}}
```

This replaces all C comments `/*...*/` by `\x C{/*...*/}`. But C comments may span multiple lines, i.e. the `^^J` should be inside it.

The macro `\x <letter>{<text>}` is replaced by one or more occurrences of `\z <letter>{<text>}` in the post-processing phase, each parameter `<text>` of `\z` is from from a single line. Parameters not crossing line boundary are represented by `\x C{<text>}` and replaced by `\z C{<text>}` without any change. But:

```
\x C{<text1>^^J<text2>^^J<text3>}
```

is replaced by

```
\z C{<text1>}^^J\z C{<text2>}^^J\z C{<text3>}
```

`\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 for each line separately (e.g. for C comments spanning multiple lines).

- `\y {<text>}` is replaced by `\<text>` in the post-processing phase. It should be used for macros without a parameters. You cannot use unprotected macros as replacement text before the post-processing phase, because the post-processing phase is based on the expansion of the whole verbatim text.

hi-syntax.opm

```
3 \_codedecl \hisyntax {Syntax highlighting of verbatim listings <2022-04-04>} % preloaded in format
```

The macros `\replfromto` and `\replthis` manipulate the verbatim text that is already stored in the `\_tmpb` macro.

`\replfromto {<from>}{<to>}{<replacement>}` finds the first occurrence of `<from>` and the first occurrence of `<to>` following it. The `<text>` between them is packed into `#1` and available to `<replacement>` which ultimately replaces `<text>`.

`\replfromto` continues by finding next `<from>`, then, next `<to>` repeatedly over the whole verbatim text. If the verbatim text ends with opening `<from>` but has no closing `<to>`, then `<to>` is appended to the verbatim text automatically and the last part of the verbatim text is replaced too.

The first two parameters are expanded before use of `\replfromto`. You can use `\csstring%` or something else here.

hi-syntax.opm

```
23 \_def\replfromto #1#2{\_edef\_tmpa{{#1}{#2}}\_ea\_replfromtoE\_tmpa}
24 \_def\replfromtoE#1#2#3{% #1=from #2=to #3=replacement
25   \_def\replfrom##1#1#2{\_addto\_tmpb{##1}%
26     \_ifx\_fin##2\_ea\_replstop \_else \_afterfi{\_replto##2}\_fi}%
27   \_def\replto##1#2#2{%
28     \_ifx\_fin##2\_afterfi{\_replfin##1}\_else
29     \_addto\_tmpb{##3}%
30     \_afterfi{\_replfrom##2}\_fi}%
31   \_def\replfin##1#1\_fin{\_addto\_tmpb{##3}\_replstop}%
32   \_edef\_tmpb{\_ea}\_ea\_replfrom\_tmpb#1\_fin#2\_fin\_fin\_relax
33 }
34 \_def\replstop#1\_fin\_relax{}
35 \_def\_finrepl{}
```

The `\replthis {<pattern>}{<replacement>}` replaces each `<pattern>` by `<replacement>`. 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 `<from>`, `<to>` and `<pattern>` are not found when they are hidden in braces `{...}`. E.g.

```
\replfromto{/*}{*/}{\x C{/*#1*/}}
```



replaces all C comments by `\x C{...}`. The patterns inside `{...}` are not used by next usage of `\replfromto` or `\replthis` macros.

The `\_xscan` macro replaces occurrences of `\x` by `\z` in the post-processing phase. The construct `\x <letter>{<text>}` expands to `\_xscan {<letter>}{<text>}^^J`. If #3 is `\_fin` then it signals that something wrong happens, the `<from>` was not terminated by legal `<to>` when `\replfromto` did work. We must to fix this by using the `\_xscanR` macro.

hi-syntax.opm

```
63 \def\_xscan#1#2^^J#3{\_ifx\_fin#3 \_ea\_xscanR\_fi
64 \z{#1}{#2}%
65 \_ifx^#3\_else ^^J\_afterfi{\_xscan{#1}{#3}\_fi}
66 \def\_xscanR#1\_fi#2~{^^J}
```

The `\hicolor <letter> <color>` defines `\_z:<letter>{<text>}` as `{<color><text>}`. It should be used in the context of `\x <letter>{<text>}` macros.

hi-syntax.opm

```
74 \def\_hicolor #1#2{\_sdef\_z:#1}{##1{#2##1}}
```

`\hisyntax{<name>}` re-defines default `\_prepareverbdata<macro><verbtex>`, but in order to do it does more things: It saves `<verbtex>` to `\_tmpb`, appends `\n` around spaces and `^^J` characters in pre-processing phase, opens `hisyntax-<name>.opm` file if `\_hisyntax<name>` is not defined. Then `\_the\_hisyntax<name>` is processed. Finally, the post-processing phase is realized by setting appropriate values to the `\x` and `\y` macros and doing `\_edef\_tmpb{\_tmpb}`.

hi-syntax.opm

```
87 \def\_hisyntax#1{\_def\_prepareverbdata##1##2{%
88 \_let\n=\_relax \_let\b=\_relax \_def\t{\n\_noexpand\t\n}\_let\_start=\_relax
89 \_adef{ }{\n\_noexpand\ \n}\_edef\_tmpb{\_start^^J##2\_fin}%
90 \_replthis{^^J}{\n^^J\b\n}\_replthis{\b\n\_fin}{\_fin}%
91 \_let\x=\_relax \_let\y=\_relax \_let\z=\_relax \_let\t=\_relax
92 \_hicomments % keeps comments declared by \commentchars
93 \_endlinechar=`^M
94 \_lowercase{\_def\_tmpa{#1}}%
95 \_ifcname\_hialias:\_tmpa\_endcsname \_edef\_tmpa{\_cs{\_hialias:\_tmpa}}\_fi
96 \_ifx\_tmpa\_empty \_else
97 \_unless \_ifcname\_hisyntax\_tmpa\_endcsname
98 \_isfile{hisyntax-\_tmpa.opm}\_iftrue \_opinput {hisyntax-\_tmpa.opm} \_fi\_fi
99 \_ifcname\_hisyntax\_tmpa\_endcsname
100 \_ifcname hicolors\_tmpa\_endcsname
101 \_cs{\_hicolors\_tmpa}=\_cs{hicolors\_tmpa}%
102 \_fi
103 \_ea\_the \_csname\_hisyntax\_tmpa\_endcsname % \_the\_hisyntax<name>
104 \_the\_hicolors % colors which have precedece
105 \_else\_opwarning{Syntax "\_tmpa" undeclared (no file hisyntax-\_tmpa.opm)}
106 \_fi\_fi
107 \_replthis{\_start\n^^J}{\_replthis{^^J\_fin}{^^J}%
108 \_def\n{\_def\b{\_adef{ }{\_dsp}%
109 \_bgroup \_lccode`~=\` \_lowercase{\_egroup\_def\ {\_noexpand~}}%
110 \_def\w####1{####1}\_def\x####1####2{\_xscan{####1}####2^^J}%
111 \_def\y####1{\_ea \_noexpand \_csname ####1\_endcsname}%
112 \_edef\_tmpb{\_tmpb}%
113 \_def\z####1{\_cs{\_z:####1}}%
114 \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
115 \_localcolor
116 }}
117 \_public \hisyntax \hicolor ;
```

Aliases for languages can be declared like this. When `\hisyntax{xml}` is used then this is the same as `\hisyntax{html}`.

hi-syntax.opm

```
124 \_sdef{\_hialias:xml}{html}
125 \_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 \_codeldecl \inspic {Graphics <2026-03-16>} % 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.

All \inspic macros are surrounded in \hbox in order user can write \moveright\inspic ... or something similar.

graphics.opm

```
14 \_def\_inspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inspicB\_inspicA}
15 \_def\_inspicA #1 {\_inspicB {#1}}
16 \_def\_inspicB #1{%
17   \_pdfximage \_ifdim\_picwidth=\_zo \_else width\_picwidth\_fi
18   \_ifdim\_picheight=\_zo \_else height\_picheight\_fi
19   \_the\_picparams {\_the\_picdir#1}%
20   \_pdfrefximage\_pdflastximage\_egroup}
21
22 \_public \inspic ;
```

Inkscape can save a picture to \*.pdf file and labels for the picture to \*.pdf\_tex file. The second file is in L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X file by plain T<sub>E</sub>X macros when \inkinspic is used. These macros are stored in the \\_inkdefs tokens list and it is used locally in the group. The solution is borrowed from OPmac trick 0032.

graphics.opm

```
34 \_def\_inkinspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}
35 \_def\_inkinspicA #1 {\_inkinspicB {#1}}
36 \_def\_inkinspicB #1{%
37   \_ifdim\_picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
38   \_tmptoks={#1}%
39   \_the\_inkdefs
40   \_opinput {\_the\_picdir #1\_tex}% file with labels
41   \_egroup}
42
43 \_newtoks\_tmptoks
44 \_newtoks\_inkdefs \_inkdefs={%
45   \_def\makeatletter#1\makeatother{%
46     \_def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_fin \_inspic{\_the\_tmptoks}\_hss}%
47     \_def\_inkscanpage#1page=#2,#3\_fin{\_ifx,#2,\_else\_picparams{page#2}\_fi}%
48     \_def\put(#1,#2)#3{\_nointerlineskip\_vbox to\_zo{\_vss\_hbox to\_zo{\_kern#1\_picwidth
49       \_pdfsave\_hbox to\_zo{#3}\_pdfrestore\_hss}\_kern#2\_picwidth}}%
50     \_def\begin#1{\_csname\_begin#1\_endcsname}%
51     \_def\_beginpicture(#1,#2){\_vbox\_bgroup
52       \_hbox to\_picwidth{\_kern#2\_picwidth \_def\end##1{\_egroup}}%
53       \_def\_beginabular[#1]#2#3\end#4{%
54         \_vtop{\_def\\{\_cr}\_tabiteml{\_tabitemr{\_table{#2}{#3}}}%
55         \_def\color[#1]#2{\_scancolor #2,%
56         \_def\_scancolor#1,#2,#3,{\_pdfliteral{#1 #2 #3 rg}}%
57         \_def\makebox(#1)[#2]#3{\_hbox to\_zo{\_csname\_mbx:#2\_endcsname{#3}}}%
58         \_sdef{\_mbx:lb}#1{#1\_hss}\_sdef{\_mbx:rb}#1{\_hss#1}\_sdef{\_mbx:b}#1{\_hss#1\_hss}%
59         \_sdef{\_mbx:lt}#1{#1\_hss}\_sdef{\_mbx:rt}#1{\_hss#1}\_sdef{\_mbx:t}#1{\_hss#1\_hss}%
60         \_def\rotatebox#1#2{\_pdfrotate{#1}#2}%
61         \_def\lineheight#1{%
62         \_def\setlength#1#2{%
63         \_def\transparent#1{\_transparency\_exprA[0]{(1-#1)*255} }%
64         % Inkscape may generate \textbf{\textit{\textsc{TEXT}}}%
65         \_def\textbf#1{\_begingroup\_let\_it\_bi\_bf #1\_endgroup}%
66         \_def\textit#1{\_begingroup\_it #1\_endgroup}%
67         \_def\textsl#1{\_begingroup\_trycs{slant}{\_it #1\_endgroup}%
68       }
69 \_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

```

78 \def\pdfscale#1#2{\pdfsetmatrix{#1 0 0 #2}}
79
80 \def\gonfunc#1#2{%
81   \directlua{tex.print(string.format('\pcent.4f',math.#1(3.14159265*(#2)/180)))}%
82 }
83 \def\sine{\gonfunc{sin}}
84 \def\cos{\gonfunc{cos}}
85
86 \def\pdfrotate#1{\pdfsetmatrix{\cos{#1} \sin{#1} \sin{#1}-180} \cos{#1}}
87
88 \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

```

102 \def\multiplyMxV #1 #2 #3 #4 {% matrix * (vvalX, vvalY)
103   \tmpdim = #1\ vvalX \advance\tmpdim by #3\ vvalY
104   \vvalY = #4\ vvalY \advance\vvalY by #2\ vvalX
105   \vvalX = \tmpdim
106 }
107 \def\multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
108   \vvalX=#1pt \vvalY=#2pt \ea\multiplyMxV \currmatrix
109   \edef\tmpb{\ea\ignorept\the\vvalX\space \ea\ignorept\the\vvalY}%
110   \vvalX=#3pt \vvalY=#4pt \ea\multiplyMxV \currmatrix
111   \edef\currmatrix{\tmpb\space
112     \ea\ignorept\the\vvalX\space \ea\ignorept\the\vvalY\space}%
113 }
114 \def\transformbox#1#2{\hbox bdir0{\setbox0=\dirhbox{#2}}%
115   \dimendef\vvalX 11 \dimendef\vvalY 12 % we use these variables
116   \dimendef\newHt 13 \dimendef\newDp 14 % only in this group
117   \dimendef\newLt 15 \dimendef\newRt 16
118   \ifnum\inboxdirection>1 \tbreset0\fi
119   \setbox0=\hbox{\box0}\pretransform{#1}%
120   \kern-\newLt \vrule height\newHt depth\newDp width\zo
121   \ht0=\zo \dp0=\zo
122   \pdfsave#1\rlap{\box0}\pdfrestore \kern\newRt}%
123 }
124 \def\pretransform #1{\def\currmatrix{1 0 0 1}%
125   \def\pdfsetmatrix##1{\edef\tmpb{##1 }\ea\multiplyMxM \tmpb\unskip}%
126   \let\pdfsetmatrix=\pdfsetmatrix #1%
127   \setnewHtDp Opt \ht0 \setnewHtDp Opt -\dp0
128   \setnewHtDp \wd0 \ht0 \setnewHtDp \wd0 -\dp0
129   \protected\def \pdfsetmatrix {\pdfextension setmatrix}%
130   \let\pdfsetmatrix=\pdfsetmatrix
131 }
132 \def\setnewHtDp #1 #2 {%
133   \vvalX=#1\relax \vvalY=#2\relax \ea\multiplyMxV \currmatrix
134   \ifdim\vvalX<\newLt \newLt=\vvalX \fi \ifdim\vvalX>\newRt \newRt=\vvalX \fi
135   \ifdim\vvalY>\newHt \newHt=\vvalY \fi \ifdim\vvalY>\newDp \newDp=-\vvalY \fi
136 }
137
138 \def\rotbox#1#2{%
139   \isequal{90}{#1}\iftrue \rotboxA{#1}{\kern\ht0 \tmpdim=\dp0}{\vfill}{#2}%
140   \else \isequal{-90}{#1}\iftrue \rotboxA{#1}{\kern\dp0 \tmpdim=\ht0}{#2}%
141   \else \transformbox{\pdfrotate{#1}}{#2}%
142   \fi \fi
143 }
144 \def\rotboxA #1#2#3#4{\hbox bdir0{\setbox0=\dirhbox{#4}}#2%
145   \vbox to\wd0{#3\wd0=\zo \dp0=\zo \ht0=\zo
146     \pdfsave\pdfrotate{#1}\box0\pdfrestore\vfill}%
147   \kern\tmpdim
148 }}
149 \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  $\langle \text{up} \rangle$  and  $\langle \text{right} \rangle$ , puts the  $\langle \text{text} \rangle$  and returns back. The current point is unchanged after this macro ends. The reference point of the  $\langle \text{text} \rangle$  is the left edge of its baseline if `\inboxdirection=0` else it is the right edge of the baseline of the  $\langle \text{text} \rangle$ . This rule is ignored when the optional parameter [l] (left edge) or [r] (right edge) of the `\puttext` macro is given. For example `\puttext[r] -5mm 0mm {text}`.

`\tbreset` $\langle \text{number} \rangle$  supposes that the box  $\langle \text{number} \rangle$  is created in dir modes 2 or 3 (top to bottom texts) and resets it to a `\hbox` in dirmode 1. I.e. it reverts the meaning of `\wd` and `\ht+\dp`. It is used in `\puttext` and similar macros and it decides, where the baseline of vertically printed text is. You can re-define it, but this default macro puts the baseline 1em under the top of the resulting box.

graphics.opm

```

172 \def\scantwodimens{%
173   \directlua{tex.print(string.format('\pcent d){\pcent d}',
174     token.scan_dimen(),token.scan_dimen()))}%
175 }
176 \optdef\puttext[]{\ea\ea\ea\puttextA\scantwodimens}
177 \long\def\puttextA#1#2#3{\setbox0=\dirhbox{#3}\dimen1=#1sp \dimen2=#2sp \puttextB}
178 \def\puttextB{%
179   \ifvmode
180     \ifdim\prevdepth>\zo \vskip-\prevdepth \relax \fi
181     \nointerlineskip
182     \fi
183     \ifnum\inboxdirection>1 \tbreset0\fi
184     \if r\the\opt\relax \setbox0=\hbox bdir1{\box0}\fi
185     \if l\the\opt\relax \setbox0=\hbox bdir0{\box0}\fi
186     \wd0=\zo \ht0=\zo \dp0=\zo
187     \vbox bdir0 to\zo{\kern-\dimen2 \hbox to\zo{\kern\dimen1 \box0\hss}\vss}%
188 }
189 \def\tbreset#1{%
190   \ifnum\inboxdirection=3 \ifdim\dimexpr \ht#1+\dp#1>1.2em % \vbox probably inside
191     \ht#1=\dimexpr\ht#1+\dp#1-.3em \dp#1=.3em \fi\fi % LuaTeX bug for dirmode 3 ??
192   \setbox#1=\hbox bdir1{\lower\dimexpr\wd#1-1em\box#1}%
193 }

```

`\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

```

205 \def\putpic{\ea\ea\ea\putpicA\scantwodimens}
206 \def\putpicA#1#2{\dimen1=#1sp \dimen2=#2sp \ea\ea\ea\putpicB\scantwodimens}
207 \def\putpicB#1#2#3{\setbox0=\hbox{\picwidth=#1sp \picheight=#2sp \inspic{#3}}\puttextB}
208
209 \newbox\bgbox
210 \def\backgroundpic#1{%
211   \setbox\bgbox=\hbox{\picwidth=\pdfpagewidth \picheight=\pdfpageheight \inspic{#1}}%
212   \pgbackground={\copy\bgbox}
213 }
214 \def\nospec{0pt}
215 \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

```

231 \def\circle#1#2{\expr{.5*#1} 0 m
232   \expr{.5*#1} \expr{.276*#2} \expr{.276*#1} \expr{.5*#2} 0 \expr{.5*#2} c
233   \expr{-.276*#1} \expr{.5*#2} \expr{-.5*#1} \expr{.276*#2} \expr{-.5*#1} 0 c
234   \expr{-.5*#1} \expr{-.276*#2} \expr{-.276*#1} \expr{-.5*#2} 0 \expr{-.5*#2} c
235   \expr{.276*#1} \expr{-.5*#2} \expr{.5*#1} \expr{-.276*#2} \expr{.5*#1} 0 c h}
236
237 \def\oval#1#2#3{0 \expr{-(#3)} m \expr{#1} \expr{-(#3)} 1

```

```

238 \_expr{(#1)+.552*(#3)} \_expr{-(#3)} \_expr{(#1)+(#3)} \_expr{-.552*(#3)}
239 \_expr{(#1)+(#3)} 0 c
240 \_expr{(#1)+(#3)} \_expr{#2} 1
241 \_expr{(#1)+(#3)} \_expr{(#2)+.552*(#3)} \_expr{(#1)+.552*(#3)} \_expr{(#2)+(#3)}
242 \_expr{#1} \_expr{(#2)+(#3)} c
243 0 \_expr{(#2)+(#3)} 1
244 \_expr{-.552*(#3)} \_expr{(#2)+(#3)} \_expr{-(#3)} \_expr{(#2)+.552*(#3)}
245 \_expr{-(#3)} \_expr{#2} c
246 \_expr{-(#3)} 0 1
247 \_expr{-(#3)} \_expr{-.552*(#3)} \_expr{-.552*(#3)} \_expr{-(#3)} 0 \_expr{-(#3)} c h
248
249 \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`, `\lwidth`, `\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 `\setflcolors` uses the `\setcolor` macro to separate filling (non-stroking) color and stroking color. The `\coc` macro means “create oval or circle” and it expands to the stroking primitive `S` or filling primitive `f` or both `B`. Only boundary stroking is performed after `\fcolor=\relax`. You cannot combine `\fcolor=\relax` with `\shadow=Y`.

graphics.opm

```

266 \_newdimen \_lwidth
267 \_def\fcolor{\_let\fcolorvalue}
268 \_def\lcolor{\_let\lcolorvalue}
269 \_def\shadow{\_let\shadowvalue}
270 \_def\overlapmargins{\_let\overlapmarginsvalue}
271 \_def\ratio{\_isnextchar ={\_ratioA}{\_ratioA=}}
272 \_def\ratioA =#1 {\_def\ratiovalue{#1}}
273 \_def\touppervalue#1{\_ifx#1n\_let#1=N\_fi}
274
275 \_def\setflcolors#1{% use only in a group
276 \_def\setcolor##1##2##3{##1 ##2}%
277 \_edef#1{\_unless\_ifx\fcolorvalue\_relax \fcolorvalue \_fi}%
278 \_def\setcolor##1##2##3{##1 ##3}%
279 \_edef#1{#1\_space\lcolorvalue\_space}%
280 }
281 \_optdef\inoval[]{\_vbox bdir0\_bgroup
282 \_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
283 \_shadow=N \_overlapmargins=N \_hhkern=Opt \_vvkern=Opt
284 \_the\_ovalparams \_relax \_the\_opt \_relax
285 \_touppervalue\overlapmarginsvalue \_touppervalue\shadowvalue
286 \_ifx\overlapmarginsvalue N%
287 \_advance\_hsize by-2\_hhkern \_advance\_hsize by-2\_roundness \_fi
288 \_setbox0=\dirhbox\_bgroup\_bgroup \_aftergroup\_inova1A \_kern\_hhkern \_let\_next=%
289 }
290 \_def\inova1A{\_egroup % of \setbox0=\hbox\bgroup
291 \_ifnum\_inboxdirection>1 \_tbreset0\_fi
292 \_ifdim\_vvkern=\_zo \_else \_ht0=\_dimexpr\_ht0+\_vvkern \_relax
293 \_dp0=\_dimexpr\_dp0+\_vvkern \_relax \_fi
294 \_ifdim\_hhkern=\_zo \_else \_wd0=\_dimexpr\_wd0+\_hhkern \_relax \_fi
295 \_ifx\overlapmarginsvalue N\_dimen0=\_roundness \_dimen1=\_roundness
296 \_else \_dimen0=-\_hhkern \_dimen1=-\_vvkern \_fi
297 \_setflcolors\_tmp
298 \_hbox{\_kern\_dimen0
299 \_vbox to\_zo{\_kern\_dp0
300 \_ifx\shadowvalue N\_else
301 \_edef\_tmpb{\_bp{\_wd0+\_lwidth}}{\_bp{\_ht0+\_dp0+\_lwidth}}{\_bp{\_roundness}}}%
302 \_doshadow\_oval
303 \_fi
304 \_pdfliteral{q \_bp{\_lwidth} w \_tmp
305 \_oval{\_bp{\_wd0}}{\_bp{\_ht0+\_dp0}}{\_bp{\_roundness}} \_coc\_space Q}\_vss}%
306 \_ht0=\_dimexpr\_ht0+\_dimen1 \_relax \_dp0=\_dimexpr\_dp0+\_dimen1 \_relax
307 \_box0
308 \_kern\_dimen0}%
309 \_egroup % of \vbox\bgroup
310 }

```

```

311 \_optdef\incircle[]{\_vbox bdir0\_bgroup
312 \_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
313 \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt
314 \_ea\_the \_ea\_circleparams \_space \_relax
315 \_ea\_the \_ea\_opt \_space \_relax
316 \_toupervalue\_overlapmarginsvalue \_toupervalue\_shadowvalue
317 \_setbox0=\_dirhbox\_bgroup\_bgroup \_aftergroup\_incircleA \_kern\_hhkern \_let\_next=%
318 }
319 \_def\_incircleA {\_egroup % of \_setbox0=\_hbox\_bgroup
320 \_ifnum\_inboxdirection>1 \_tbreset0\_fi
321 \_wd0=\_dimexpr \_wd0+\_hhkern \_relax
322 \_ht0=\_dimexpr \_ht0+\_vvkern \_relax \_dp0=\_dimexpr \_dp0+\_vvkern \_relax
323 \_ifdim \_ratiovalue\_dimexpr \_ht0+\_dp0 > \_wd0
324 \_dimen3=\_dimexpr \_ht0+\_dp0 \_relax \_dimen2=\_ratiovalue\_dimen3
325 \_else \_dimen2=\_wd0 \_dimen3=\_expr{1/\_ratiovalue}\_dimen2 \_fi
326 \_setflcolors\_tmp
327 \_ifx\_overlapmarginsvalue N\_dimen0=\_zo \_dimen1=\_zo
328 \_else \_dimen0=-\_hhkern \_dimen1=-\_vvkern \_fi
329 \_hbox{\_kern\_dimen0
330 \_ifx\_shadowvalue N\_else
331 \_edef\_tmpb{\_bp{\_dimen2+\_lwidth}}{\_bp{\_dimen3+\_lwidth}}{}}%
332 \_doshadow\_circlet
333 \_fi
334 \_pdfliteral{q \_bp{\_lwidth} w \_tmp \_mv{\_bp{.5\_wd0}}{\_bp{(\_ht0-\_dp0)/2}}
335 {\_circle{\_bp{\_dimen2}}{\_bp{\_dimen3}} \_coc} Q}%
336 \_ifdim\_dimen1=\_zo \_else
337 \_ht0=\_dimexpr \_ht0+\_dimen1 \_relax \_dp0=\_dimexpr \_dp0+\_dimen1 \_relax \_fi
338 \_box0
339 \_kern\_dimen0}
340 \_egroup % of \_vbox\_bgroup
341 }
342 \_def\_circlet#1#2#3{\_circle{#1}{#2}}
343 \_def\_coc{\_ifx\_fcolorvalue\_relax S\_else \_ifdim\_lwidth=0pt f\_else B\_fi\_fi}
344
345 \_public \_inoval \_incircle \_ratio \_lwidth \_fcolor \_lcolor \_shadow \_overlapmargins ;

```

Just before defining shadows, which require special graphics states, we define means for managing these graphics states and other PDF page resources (graphics states, patterns, shadings, etc.). Our mechanism, defined mostly in Lua (see 2.39.5, uses single dictionary for each PDF page resource type (extgstate, etc.) for all pages (\pdfpageresources just points to it).

The macro `\addextgstate{<PDF name>}{<PDF dictionary>}` is a use of that general mechanism and shall be used for adding more graphics states. It must be used *after* `\dump`. It's general variant defined in Lua is `\_addpageresource{<resource type>}{<PDF name>}{<PDF dictionary>}`. You can use `\pageresources` or `\_pageresources` if you need to insert resource entries to manually created PDF XObjects.

graphics.opm

```

363 \_def\_addextgstate{\_addpageresource{ExtGState}}
364
365 \_public \_addextgstate ;
366 \_def\_pageresources{\_pageresources}
367 \_def\_addpageresource{\_addpageresource}

```

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  $2 \times \_shadowb$  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

```

378 \_def\_shadowlevels{9} % number of layers for blur effect
379 \_def\_shadowdarknessA{0.025} % transparency of first shadowlevels/2 layers
380 \_def\_shadowdarknessB{0.07} % transparency of second half of layers
381 \_def\_shadowmoveto{1.8 -2.5} % vector defines shifting layer (in bp)
382 \_def\_shadowb{1} % 2*shadowb = blurring area thickness
383
384 \_def\_insertshadowresources{%
385 \_addextgstate{op1}{<</ca \_shadowdarknessA>>}}%
386 \_addextgstate{op2}{<</ca \_shadowdarknessB>>}}%
387 \_glet\_insertshadowresources=\_relax
388 }

```



The `\doshadow{<curve>}` does the shadow effect.

graphics.opm

```

394 \_def\doshadow#1{\_vbox{%
395   \_insertshadowresources
396   \_tmpnum=\_numexpr (\_shadowlevels-1)/2 \_relax
397   \_edef\_tmpfin{\_the\_tmpnum}%
398   \_ifnum\_tmpfin=0 \_def\_shadowb{0}\_def\_shadowstep{0}%
399   \_else \_edef\_shadowstep{\_expr{\_shadowb/\_tmpfin}}\_fi
400   \_def\_tmpa#1#2#3{\_def\_tmpb
401     {#1{#1+2*\_the\_tmpnum*\_shadowstep}{#2+2*\_the\_tmpnum*\_shadowstep}{#3}}}%
402   \_ea \_tmpa \_tmpb
403   \_def\_shadowlayer{%
404     \_ifnum\_tmpnum=0 /op2 gs \_fi
405     \_tmpb\_space f
406     \_immediateassignment\_advance\_tmpnum by-1
407     \_ifnum-\_tmpfin<\_tmpnum
408       \_ifx#1\_oval 1 0 0 1 \_shadowstep\_space \_shadowstep\_space cm \_fi
409       \_ea \_shadowlayer \_fi
410   }%
411   \_pdfliteral{q /op1 gs 0 g 1 0 0 1 \_shadowmoveto\_space cm
412     \_ifx#1\_circlet 1 0 0 1 \_bp{.5\_wd0} \_bp{(\_ht0-\_dp0)/2} cm
413     \_else 1 0 0 1 -\_shadowb\_space -\_shadowb\_space cm \_fi
414     \_shadowlayer Q}
415 }}

```

A generic macro `\_clipinpath<x> <y> <curve> <text>` declares a clipping path by the `<curve>` shifted by the `<x>`, `<y>`. The `<text>` is typeset when such clipping path is active. Dimensions are given by bp without the unit here. The macros `\_clipinoval<x> <y> <width> <height> {<text>}` and `\_clipincircle<x> <y> <width> <height> {<text>}` are defined here. These macros read normal T<sub>E</sub>X dimensions in their parameters.

graphics.opm

```

426 \_def\_clipinpath#1#2#3#4{% #1=x-pos[bp], #2=y-pos[bp], #3=curve, #4=text
427   \_hbox bdir0 {\_setbox0=\_hbox{\_dirhbox{#4}}}%
428   \_tmpdim=\_wd0 \_wd0=\_zo
429   \_pdfliteral{q \_mv{#1}{#2}{#3 W n}}%
430   \_box0\_pdfliteral{Q}\_kern\_tmpdim
431 }%
432 }
433
434 \_def\_clipinoval {\_ea\_ea\_ea\_clipinovalA\_scantwodimens}
435 \_def\_clipinovalA #1#2{%
436   \_def\_tmp{#1/65781.76}{#2/65781.76}}%
437   \_ea\_ea\_ea\_clipinovalB\_scantwodimens
438 }
439 \_def\_clipinovalB{\_ea\_clipinovalC\_tmp}
440 \_def\_clipinovalC#1#2#3#4{%
441   \_ea\_clipinpath{#1-(#3/131563.52)+(\_bp{\_roundness})}{#2-(#4/131563.52)+(\_bp{\_roundness})}%
442   {\_oval{#3/65781.76-(\_bp{2\_roundness})}{#4/65781.76-(\_bp{2\_roundness})}{\_bp{\_roundness}}}%
443 }
444 \_def\_clipincircle {\_ea\_ea\_ea\_clipincircleA\_scantwodimens}
445 \_def\_clipincircleA #1#2{%
446   \_def\_tmp{#1/65781.76}{#2/65781.76}}%
447   \_ea\_ea\_ea\_clipincircleB\_scantwodimens
448 }
449 \_def\_clipincircleB#1#2{%
450   \_ea\_clipinpath\_tmp{\_circle{#1/65781.76}{#2/65781.76}}%
451 }
452 \_public \_clipinoval \_clipincircle ;

```

## 2.30 The \table macro, tables and rules

### 2.30.1 The boundary declarator :

The `<declaration>` part of `\table{<declaration>}{<data>}` includes column declarators (letters) and other material: the | or (`<cmd>`). 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 (*cmd*) 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 (*setting*) followed by column declarator. But if the column is not first, you must use :(*setting*). Example. We have three centered columns, the second one have to be in bold font and the third one have to be in red: `\table{c:(\bf)c:(\Red)c}{\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 (*cmds*) 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=⟨value⟩`) in the *declaration* 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`, `\crlri` 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⟨size⟩{⟨declaration⟩}{⟨data⟩}` uses stretchability or shrinkability of all spaces between columns generated by `\tabskip` value and eventually by `\tabskip1`, `\tabskipr` values. See example below.
- `\table pxtoto⟨size⟩{⟨declaration⟩}{⟨data⟩}` expands the columns declared by `p{⟨size⟩}`, if the *size* is given by a virtual `\tsize` unit. See the example below.

Example of `\table to⟨size⟩`:

```
\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 pxtoto⟨size⟩` (means “paragraphs expanded to”):

```
\table pxtoto\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{<coefficient>\tsize}` columns in the table when `ppto` keyword is used.

```
\table ppto13cm {r p{3.5\tsize} p{2\tsize} p{\tsize} l}{<data>}
```

This gives the ratio of widths of individual paragraphs in the table 3.5:2:1.

## 2.30.4 \eqbox: boxes with equal width across the whole document

The `\eqbox` [*label*]{<text>} behaves like `\hbox{<text>}` 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` [*label*]{<text>} behaves like `\hbox to <dim:label> {\hss <text>\hss}`, where <dim:label> 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 Implementation of the \table macro and friends

```
3 \_codedecl \table {Macros for tables <2024-11-21>} % 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 `ppto<size>` to #1 and `\_tablew` sets the `to<size>` to the `\_tablew` macro. If `ppto<size>` is used then `\_tablew` is empty and `\_tmpdim` includes given <size>. The `\_ifppto` 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 \_ifppto
25 \_def\_table#1#{\_tablebox\_bgroup \_tablew#1\_empty\_fin
26 \_bgroup \_catcodetable\_optexcatcodes \_tableA}
27 \_def\_tablew#1#2\_fin{\_pptofalse
28 \_ifx#1\_empty \_def\_tablew{}\\_else
29 \_ifx#1p \_def\_tablew{\\_tablew#2\_fin \_else \_def\_tablew{#1#2}\_fi\_fi}
30 \_def\_tablew xto#1\_fin{\_tmpdim=#1\_relax \_pptotrue}
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`.

```
42 \_def\_tableA#1{\_egroup
43 \_the\_thistable \_global\_thistable={}%
44 \_ea\_ifx\_ea\_the\_tabstrut\_setbox\_tstrutbox=\_null
45 \_else \_setbox\_tstrutbox=\_hbox{\_the\_tabstrut}%
```

table.opm

```

46     \setbox\tstrutbox=\hbox{\_vrule width\_zo
47       height\_dimexpr\_ht\tstrutbox+\_tablinespace
48       depth\_dimexpr\_dp\tstrutbox+\_tablinespace}%
49     \offinterlineskip
50     \lineskip=-2\_tablinespace
51   \_fi
52   \_colnum=0 \_let\_addtabitem=\_addtabitemx
53   \_def\_tmpa{\_tabdata={\_colnum1\_relax}\_scantabdata#1\_relax
54     \_the\_everytable \_tableB
55 }

```

The `\_tableB` saves `<data>` to `\_tmpb`. The `#1` parameter is re-scanned in order to set potential `#` characters inside it as others. This is needed because of `\gdef\_tmpb` and next `\replstrings`. They prefix each macro `\crl` (etc.) by `\_crrc`. See `\_tabreplstrings`. The whole `\_tableB` macro is hidden in `{...}` in order to there may be `\table` in `\table` and we want to manipulate with `&` and `\cr` as with normal tokens in the `\_tabreplstrings`, not as the item delimiters of an outer `\table`.

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. All columns where `p{... \tsize}` is used, are created as empty in this first pass. So, the `\wd0` is the width of all other columns. The `\_tsizesum` includes the sum of `\tsize`’s in `\hsize` units after first pass. The desired table width is stored in the `\_tmpdim`, so `\_tmpdim-\wd0` is the rest which have to be filled by `\tsizes`. Then the `\tsize` is re-calculated 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.

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

```

94 \_long\_def\_tableB #1{%
95   {\_catcode\#=12 % see OpTeX trick 0117:
96     \directlua{tex.sprint(-1,"\_luaescapestring{\_unexpanded{\_gdef\_tmpb{#1}}}")}%
97     \tablereplstrings \edef\_tabskipmid{\_the\_tabskip}\_tabskip=\_tabskipl
98     \_ifpxto
99       \edef\_tsizes{\_global\_tsizesum=\_the\_tsizesum \_gdef\_noexpand\_tsizelast{\_tsizelast}}%
100       \_tsizesum=\_zo \_def\_tsizelast{0}%
101       \_tsize=-\_hsize \_setbox0=\_vbox{\_tablepxpreset \_halign \_tableC}%
102       \_advance\_tmpdim by-\_wd0
103       \_ifdim \_tmpdim >\_zo \_else \_tsizesum=\_zo \_fi
104       \_ifdim \_tsizesum >\_zo \_tsize =\_expr{\_number\_hsize/\_number\_tsizesum}\_tmpdim
105       \_else \_tsize=\_zo \_fi
106       \_tsizes % restoring values if there is a \table pxto inside a \table pxto.
107       \_setbox0=\_null \_halign \_tableC
108     \_else
109       \_halign\_tablew \_tableC
110     \_fi
111     \_glet\_tmpb\_undefined
112   }\_egroup % \_tablebox\_bgroup is in the \_table macro
113 }
114 \_def\_tableC{\_ea{\_the\_tabdata\_tabskip=\_tabskipr\_cr \_scantextokens\_ea{\_tmpb\_crrc}}}

```

`\_tabreplstrings` replaces each `\crl` etc. to `\crrc\crl`. The reason is: we want to use macros that scan its parameter to a delimiter written in the right part of the table item declaration. The `\crrc` cannot be hidden in another macro in this case.

table.opm

```

123 \_def\_tablereplstrings{%
124   \_replstring\_tmpb{\crl}{\_crrc\crl}\_replstring\_tmpb{\crl1}{\_crrc\crl1}%
125   \_replstring\_tmpb{\crl1}{\_crrc\crl1}\_replstring\_tmpb{\crl11}{\_crrc\crl11}%
126   \_replstring\_tmpb{\crlp}{\_crrc\crlp}%
127 }
128

```

```

129 \def\tablexpreset{} % can be used to de-activate references to .ref file
130 \newbox\tstrutbox % strut used in table rows
131 \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\vvkern\vrule\tabstrutA
&\the\tabiteml{\hfil#\unsskip\hfil}\the\tabitemr\tabstrutA
&\the\tabiteml{\relax#\unsskip\hfil}\the\tabitemr\vrule\tabstrutA
ddlinedata: &\dditem &\dditem\vvittem &\dditem &\dditem

```

The second result in the `\ddlinedata` macro is a template of one row of the table used by `\crli` macro.

```

151 \def\scantabdata#1{\let\next=\scantabdata
152 \ifx\relax#1\let\next=\relax
153 \else\ifx|#1\addtabvrule
154 \else\ifx|#1\def\next{\scantabdataE}%
155 \else\ifx|#1\def\next{\scantabdataF}%
156 \else\isinlist{123456789}#1\iftrue \def\next{\scantabdataC#1}%
157 \else \ea\ifx\csname\tabdeclare#1\endcsname \relax
158 \ea\ifx\csname\paratdeclare#1\endcsname \relax
159 \opwarning{tab-declarator "#1" unknown, ignored}%
160 \else
161 \def\next{\ea\scantabdataB\csname\paratdeclare#1\endcsname}\fi
162 \else \def\next{\ea\scantabdataA\csname\tabdeclare#1\endcsname}%
163 \fi\fi\fi\fi\fi\fi\next
164 }
165 \def\scantabdataA#1{\addtabitem
166 \ea\addtabdata\ea{#1\tabstrutA\tabskip\tabskipmid\relax}\scantabdata}
167 \def\scantabdataB#1#2{\addtabitem
168 \ea\addtabdata\ea{#1{#2}\tabstrutA\tabskip\tabskipmid\relax}\scantabdata}
169 \def\scantabdataC{\def\tmpb{}\afterassignment\scantabdataD\tmpnum=}
170 \def\scantabdataD#1{\loop\ifnum\tmpnum>0\advance\tmpnum by-1\addto\tmpb{#1}\repeat
171 \ea\scantabdata\tmpb}
172 \def\scantabdataE#1{\addtabdata{#1}\scantabdata}
173 \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}
\tab{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.

```

193 \newcount\colnum % number of current column in the table
194 \public \colnum ;
195
196 \def\addtabitemx{\ifnum\colnum>0
197 \addtabdata{&}\addto\ddlinedata{&\dditem}\fi
198 \advance\colnum by1 \let\tmpa=\relax
199 \ifnum\colnum>1 \etoksapp\tabdata{\colnum\the\colnum\relax}\fi}
200 \def\addtabdata{\toksapp\tabdata}

```

This code converts `||` or `|` from `\table <declaration>` to the *<converted declaration>*.

```

206 \def\addtabvrule{%
207 \ifx\tmpa\vrule \addtabdata{\kern\vvkern}%
208 \ifnum\colnum=0 \addto\vleft{\vvitem}\else\addto\ddlinedata{\vvitem}\fi
209 \else \ifnum\colnum=0 \addto\vleft{\vvitemA}\else\addto\ddlinedata{\vvitemA}\fi\fi

```

```

210 \let\tmpa=\vrule \addtabdata{\vrule}%
211 }
212 \def\tabstrutA{\_copy\tstrutbox}
213 \def\vleft{}
214 \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.

Note, that the `##` with fills are in group. The reason can be explained by following example:

```
\table{|c|c|}{\crl \Red A & B \crl}
```

We don’t want vertical line after red A to be in red.

table.opm

```

233 \def\_tabdeclarec{\_the\_tabiteml\_hfil{##}\_unsskip\_hfil\_the\_tabitemr}
234 \def\_tabdeclarel{\_the\_tabiteml{##}\_unsskip\_hfil\_the\_tabitemr}
235 \def\_tabdeclarer{\_the\_tabiteml\_hfil{##}\_unsskip\_the\_tabitemr}

```

The `\_paramtabdeclarep{<data>}` is invoked when `p{<data>}` declarator is used. First, it saves the `\hsize` value and then it runs `\_tablepar`. The `\_tablepar` macro behaves like `\_tableparbox` (which is `\vtop`) in normal cases. But there is a special case: if the first pass of `p` to table is processed then `\hsize` is negative. We print nothing in this case, i.e. `\_tableparbox` is `\ignoreit` and we advance the `\_tsizesum`. The auxiliary macro `\_tsizelast` is used to do advancing only in the first row of the table. `\_tsizesum` and `\_tsizelast` are initialized in the `\_tableB` macro.

table.opm

```

250 \def\_paramtabdeclarep#1{\_hsize=#1\_relax
251 \_the\_tabiteml\_tablepar{\_tableparB##\_tableparC}\_the\_tabitemr
252 }
253 \def\_tablepar{%
254 \_ifdim\_hsize<0pt
255 \_ifnum\_tsizelast<\_colnum\_global\_advance\_tsizesum by-\_hsize
256 \_xdef\_tsizelast{\_the\_colnum}\_fi
257 \_let\_tableparbox=\ignoreit
258 \_fi
259 \_tableparA\_tableparbox
260 }
261 \_let\_tableparbox=\vtop
262 \_let\_tableparA=\_empty
263 \_newdimen\_tsizesum
264 \_def\_tsizelast{0}

```

The `\_tableparB` initializes the paragraphs inside the table item and `\_tableparC` closes them. They are used in the `\_paramtabdeclarep` macro. The first paragraph is no indented.

table.opm

```

272 \def\_tableparB{%
273 \_baselineskip=\_normalbaselineskip\_lineskiplimit=\_zo\_noindent
274 \_unless\_ifx\tabstrutA\_empty\_raise\_ht\tstrutbox\_null\_fi
275 \_hskip\_zo\_relax
276 }
277 \def\_tableparC{%
278 \_unsskip
279 \_unless\_ifx\tabstrutA\_empty
280 \_ifvmode\_vskip\_dp\tstrutbox\_else\_lower\_dp\tstrutbox\_null\_fi
281 \_fi
282 }

```

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 TeX algorithm but the right space must be removed by macros. This is a reason why we recommend to use `\_unsskip` after each `##` in your definition of “declaration letters”. This macro isn’t only the primitive `\unskip` because we allow usage of plain TeX `\hideskip` macro: `&\hideskip text\hideskip&`.

table.opm

```
294 \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.



```

301 \_let\_fL=\_raggedright
302 \_def\_fR{\_leftskip=0pt plus 1fill \_relax}
303 \_def\_fC{\_leftskip=0pt plus1fill \_rightskip=0pt plus 1fill \_relax}
304 \_def\_fX{\_leftskip=0pt plus1fil \_rightskip=0pt plus-1fil \_parfillskip=0pt plus2fil \_relax}
305 \_public \fL \fR \fC \fX ;

```

The `\fS` macro is more tricky. The `\_tableparbox` isn't printed immediately, but `\setbox2=` is prefixed by the macro `\_tableparA`, which is empty by default (used in `\_tablepar`). The `\_tableparD` is processed after the box is set: it checks if there is only one line and prints `\hbox to\hsize{\hfil<this line>\hfil}` in this case. In other cases, the box2 is printed.

```

316 \_def\_fS{\_relax
317 \_ifdim\_hsize<0pt \_else \_def\_tableparA{\_setbox2=}\_fi
318 \_addto\_tableparC{\_aftergroup\_tableparD}%
319 }
320 \_def\_tableparD{\_setbox0=\_vbox{\_unvcopy2 \_unskip \_global\_setbox1=\_lastbox}%
321 \_ifdim\_ht0>0pt \_box2 \_setbox0=\_box1
322 \_else \_hbox to\hsize{\_hfil \_unhbox1\_unskip\_unskip\_hfil}\_setbox0=\_box2 \_fi
323 }
324 \_public \fS ;

```

The family of `\_cr*` macros `\crl`, `\crl1`, `\crl1i`, `\crl1i`, `\crlp` and `\tskip` (*dimen*) is implemented here. The `\_zerotabrule` is used to suppress the negative `\lineskip` declared by `\tablinespace`.

```

334 \_def\_crl{\_crrc\_noalign{\_hrule}}
335 \_def\_crl1{\_crrc\_noalign{\_hrule\_kern\_hhkern\_hrule}}
336 \_def\_zerotabrule {\_noalign{\_hrule height\_zo width\_zo depth\_zo}}
337
338 \_def\_crl1i{\_crrc \_zerotabrule \_omit
339 \_gdef\_dditem{\_omit\_tablinefil}\_gdef\_vvitem{\_kern\_vvkern\_vrule}\_gdef\_vvitemA{\_vrule}%
340 \_vvleft\_tablinefil\_ddlinedata\_crrc \_zerotabrule}
341 \_def\_crl1i{\_crl1\_noalign{\_kern\_hhkern}\_crl1i}
342 \_def\_tablinefil{\_leaders\_hrule\_hfil}
343
344 \_def\_crlp#1{\_crrc \_zerotabrule \_noalign{\_kern-\_drulewidth}%
345 \_omit \_xdef\_crlplist{#1}\_xdef\_crlplist{\_ea}\_crlpA\_crlplist,\_fin,%
346 \_global\_tmpnum=0 \_gdef\_dditem{\_omit\_crlpD}%
347 \_gdef\_vvitem{\_kern\_vvkern\_kern\_drulewidth}\_gdef\_vvitemA{\_kern\_drulewidth}%
348 \_vvleft\_crlpD\_ddlinedata \_global\_tmpnum=0 \_crrc \_zerotabrule}
349 \_def\_crlpA#1,{\_ifx\_fin#1\_else \_crlpB#1-\_fin,\_ea\_crlpA\_fi}
350 \_def\_crlpB#1#2-#3,{\_ifx\_fin#3\_xdef\_crlplist{\_crlplist#1#2,}\_else\_crlpC#1#2-#3,\_fi}
351 \_def\_crlpC#1-#2-#3,{\_tmpnum=#1\_relax
352 \_loop \_xdef\_crlplist{\_crlplist\_the\_tmpnum,}\_ifnum\_tmpnum<#2\_advance\_tmpnum by1 \_repeat}
353 \_def\_crlpD{\_incr\_tmpnum \_edef\_tmpa{\_noexpand\_isinlist\_noexpand\_crlplist{\_the\_tmpnum,}}%
354 \_tmpa\_iftrue \_kern-\_drulewidth \_tablinefil \_kern-\_drulewidth\_else\_hfil \_fi}
355
356 \_def\_tskip{\_afterassignment\_tskipA \_tmpdim}
357 \_def\_tskipA{\_gdef\_dditem{\_gdef\_vvitem{\_gdef\_vvitemA{\_gdef\_tabstrutA{\_}}
358 \_vbox to\_tmpdim}\_ddlinedata \_crrc
359 \_zerotabrule \_noalign{\_gdef\_tabstrutA{\_copy\_tstrutbox}}}}
360
361 \_public \crl \crl1 \crl1i \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.

```

369 \_def\_mspan{\_omit \_afterassignment\_mspanA \_mscount=}
370 \_def\_mspanA[#1]#2{\_loop \_ifnum\_mscount>1 \_cs{\_span}\_omit \_advance\_mscount-1 \_repeat
371 \_count1=\_colnum \_colnum=0 \_def\_tmpa{\_tabdata={}\_scantabdata#1\_relax
372 \_colnum=\_count1 \_setbox0=\_vbox{\_halign\_ea{\_the\_tabdata\_cr#2\_cr}%
373 \_global\_setbox8=\_lastbox}%
374 \_setbox0=\_hbox{\_unhbox8 \_unskip \_global\_setbox8=\_lastbox}%
375 \_unhbox8 \_ignorespaces}
376 \_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.

table.opm

```
388 \def\vs span#1#2#{\vs panA{#1#2}}
389 \def\vs panA#1#2{\_vtop to\_zo{\_hbox{\_lower\_dimexpr
390 #1\_dimexpr(\_ht\_tstrutbox+\_dp\_tstrutbox)/2\_relax
391 \_dimexpr(\_ht\_tstrutbox+\_dp\_tstrutbox)/2\_relax\_hbox{#2}}\_vss}}
392 \_public \vs pan ;
```

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

```
403 \_newdimen\_drulewidth \_drulewidth=0.4pt
404 \_let\_orihrule=\_hrule \_let\_orivrule=\_vrule
405 \_def\_rulewidth{\_afterassignment\_rulewidthA \_drulewidth}
406 \_def\_rulewidthA{\_edef\_hrule{\_orihrule height\_drulewidth}%
407 \_edef\_vrule{\_orivrule width\_drulewidth}%
408 \_let\_rulewidth=\_drulewidth
409 \_public \vrule \hrule \rulewidth;}
410 \_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

```
420 \_long\_def\_frame#1{%
421 \_hbox{\_vrule\_vtop{\_vbox{\_hrule\_kern\_vbkern
422 \_hbox{\_kern\_hhkern{#1}\_kern\_hhkern}%
423 }\_kern\_vbkern\_hrule}\_vrule}}
424 \_public \frame ;
```

`\eqbox` and `\eqboxsize` are implemented here. The widths of all `\eqboxes` are saved to the `.ref` file in the format `\_Xe qbox{<label>}{<size>}`. The `.ref` file is read again and maximum box width for each `<label>` is saved to `\_eqb:<label>`.

table.opm

```
433 \_def\_Xe qbox#1#2{%
434 \_ifcsname\_eqb:#1\_endcsname
435 \_ifdim #2>\_cs{\_eqb:#1}\_relax \_sdef{\_eqb:#1}{#2}\_fi
436 \_else \_sdef{\_eqb:#1}{#2}\_fi
437 }
438 \_def\_eqbox #1[#2]#3{\_setbox0=\_hbox{{#3}}%
439 \_openref \_immediate\_wref \_Xe qbox{{#2}}{\_the\_wd0}}%
440 \_ifcsname\_eqb:#2\_endcsname
441 \_hbox to\_cs{\_eqb:#2}{\_ifx r#1\_hfill\_fi\_hss\_unhbox0\_hss\_ifx l#1\_hfill\_fi}%
442 \_else \_box0 \_fi
443 }
444 \_def\_eqboxsize [#1]#2{\_trycs{\_eqb:#1}{#2}}
445
446 \_public \eqbox \eqboxsize ;
```

## 2.31 Balanced multi-columns

multicolumns.opm

```
3 \_codedecl \begmulti {Balanced columns <2024-10-31>} % preloaded in format
```

`\_betweencolumns` or `\_leftofcolumns` or `\_rightofcolumns` include a material printed between columns or left of all columns or right of all columns respectively. The `\_betweencolumns` must include a stretchability or a material with exactly `\colsep` width. You can redefine these macros. For example the rule between columns can be reached by `\_def\_betweencolumns{\hss\vrule\hss}`.

`\_multiskip` puts its material at the start and at the end of `\begmulti... \endmulti`.

multicolumns.opm

```
16 \_def\_betweencolumns{\_hss} \_def\_leftofcolumns{} \_def\_rightofcolumns{}
17 \_def\_multiskip{\_medskip} % space above and below \begmulti... \endmulti
```

The code used here is documented in detail in the “`TEXbook naruby`”, pages 244–246, free available, <http://petr.olsak.net/tbn.html>, but in Czech. Roughly speaking, macros complete all material

between `\begmulti<num-columns>` and `\endmulti` into one `\vbox 6`. Then the macro measures the amount of free space at the current page using `\pagegoal` and `\pagetotal` and does `\vsplit` of `\vbox 6` to columns with 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

```

38 \_def\begmulti #1 {\_par\_bgroup\_wipeepar
39 \_ifnum\_lastpenalty>10000 \_vskip4.5\_baselineskip\_penalty9999 \_vskip-4.5\_baselineskip \_fi
40 \_multiskip \_def\_Ncols{#1}
41 \_setbox6=\_vbox\_bgroup\_bgroup \_let\_setxhsize=\_relax \_penalty-99
42 %% \hsize := column width = (\hsize+colsep) / n - \colsep
43 \_setbox0=\_hbox{\_leftofcolumns\_rightofcolumns}%
44 \_advance\_hsize by-\_wd0 \_advance\_hsize by\_colsep
45 \_divide\_hsize by\_Ncols \_advance\_hsize by-\_colsep
46 }
47 \_def\endmulti{\_vskip-\_prevdepth\_vfil
48 \_ea\_egroup\_ea\_egroup\_ea\_baselineskip\_the\_baselineskip\_relax
49 \_dimen0=.8\_maxdimen \_tmpnum=\_dimen0 \_divide\_tmpnum by\_baselineskip
50 \_splittopskip=\_baselineskip
51 \_setbox1=\_vsplit6 toOpt % initialize first \splittopskip in \box6
52 %% \dimen1 := the free space on the page
53 \_penalty0 % initialize \_pagegoal
54 \_ifdim\_pagegoal=\_maxdimen \_setcolsize\_vsize
55 \_else \_setcolsize{\_dimexpr\_pagegoal-\_pagetotal}\_fi
56 \_ifdim \_dimen1<2\_baselineskip \_vfil\_break \_setcolsize\_vsize \_fi
57 \_setsplitsize % sets \_dimen0 = \_ht6
58 \_divide\_dimen0 by\_Ncols \_relax
59 %% split the material to more pages?
60 \_ifdim \_dimen0>\_dimen1 \_splitpart
61 \_else \_balancecolumns \_fi % only balancing
62 \_multiskip \_egroup
63 }

```

Splitting columns...

multicolumns.opm

```

69 \_def\_makecolumns{\_bgroup % full page, destination height: \_dimen1
70 \_vbadness=20000 \_splitmaxdepth=\_maxdepth \_dimen6=\_wd6
71 \_createcolumns
72 \_printcolumns
73 \_egroup
74 }
75 \_def\_splitpart{%
76 \_makecolumns % full page
77 \_vskip Opt plus 1fil minus\_baselineskip \_break
78 \_setsplitsize % sets \_dimen0 = \_ht6
79 \_divide\_dimen0 by\_Ncols \_relax
80 \_ifx\_balancecolumns\_flushcolumns \_advance\_dimen0 by-.5\_vsize \_fi
81 \_setcolsize\_vsize \_dimen2=\_dimen1
82 \_advance\_dimen2 by-\_baselineskip
83 %% split the material to more pages?
84 \_ifvoid6 \_else
85 \_ifdim \_dimen0>\_dimen2 \_ea\_ea\_ea \_splitpart
86 \_else \_balancecolumns % last balancing
87 \_fi \_fi
88 }

```

Final balancing of the columns.

multicolumns.opm

```

94 \_def\_balancecolumns{\_bgroup \_setbox7=\_copy6 % destination height: \_dimen0
95 \_ifdim\_dimen0>\_baselineskip \_else \_dimen0=\_baselineskip \_fi
96 \_vbadness=20000 \_dimen6=\_wd6 \_dimen1=\_dimen0
97 \_def\_trybalance{\_createcolumns
98 \_ifvoid6 \_else

```

```

99      \advance \dimen1 by.2\baselineskip
100      \setbox6=\copy7
101      \ea \trybalance \fi}\trybalance
102      \printcolumns
103      \egroup
104 }

```

`\setcolsize` $\langle dimen \rangle$  sets initial value `\dimen1`= $\langle size \rangle$  which is used as height of columns at given page. The correction `\splittopskip`—`\topskip` is done if the columns start at the top of the page.

`\createcolumns` prepares columns with given height `\dimen1` side by side to the `\box1`.

`\printcolumns` prints the columns prepared in `\box1`. The first `\hbox{}` moves typesetting point to the next baseline. Next negative skip ensures that the first line from split columns is at this position.

multicolumns.opm

```

119 \def\setcolsize #1{\dimen1=#1\relax
120   \ifdim\dimen1=\vsize
121     \advance \dimen1 by \splittopskip \advance \dimen1 by-\topskip \fi
122 }
123 \def\createcolumns{%
124   \setbox1=\hbox{\leftofcolumns}\tmpnum=0
125   \loop \ifnum\Ncols>\tmpnum
126     \advance\tmpnum by1
127     \setbox1=\hbox{\unhbox1
128       \ifvoid6 \hbox to\dimen6{\hss}\else \vsplit6 to\dimen1 \fi
129       \ifnum\Ncols=\tmpnum \rightofcolumns \else \betweencolumns \fi}%
130   \repeat
131 }
132 \def\printcolumns{%
133   \hbox{\nobreak\vskip-\splittopskip \nointerlineskip
134     \hbox to\hsize{\unhbox1}%
135 }
136 \public \begmulti \endmulti ;

```

The accumulated `\box6` (before its splitting to pages and columns) can be very large and we cannot measure it with classical T<sub>E</sub>X methods because we can get “dimension too large” error or arithmetic overflow. The T<sub>E</sub>X limit for dimensions is about 5.75 m. For example, the `\box6` of the index in this document has about 6.5 m height.

The `\rawht` $\langle box-num \rangle$  macro solves this problem. It expands to the height plus depth of given `\vbox` as an integer in points (i.e. the fraction part of points is stripped and unit isn’t appended). The limit  $2^{31}$  of such an integer allows us to measure boxes with many kilometers height.

The trick used in the `\setsplitsize` macro sets `\dimen0` to the maximum of `.8\maxdimen` and actual `\ht6`, because 13100 is `.8\maxdimen` in points. This is sufficient because if the `\dimen0` is big then we get  $n$  columns for full page. This is not the last page where exact calculations for balancing is needed.

multicolumns.opm

```

158 \def\setsplitsize{%
159   \ifnum \rawht6 > 13100 \dimen0 = .8\maxdimen
160   \else \dimen0=\ht6 \def\rawht6{0}\fi
161 }
162 \def\rawht{\directlua{optex.raw_ht()}}

```

## 2.32 Citations, bibliography

### 2.32.1 Macros for citations and bibliography preloaded in the format

cite-bib.opm

```

3 \codedec \cite {Cite, Bibliography <2026-05-06>} % preloaded in format

```

Registers used by `\cite`, `\bib` macros are declared here. The `\bibnum` counts the bibliography items from one. The `\bibmark` is used when `\nonumcitations` is set.

cite-bib.opm

```

11 \newcount\bibnum           % the bibitem counter
12 \newtoks\bibmark          % the bibmark used if \nonumcitations
13 \newcount\lastcitenum \lastcitenum=0 % for \shortcitations
14 \public \bibnum \bibmark ;

```

`\bib` expands to `\bibpart/`. By default, `\bibpart` is empty, so internal links are in the form `cite:/\number`. If `\bibpart` is set to  $\langle bibpart \rangle$ , then internal links are `cite:\langle bibpart \rangle/\langle number \rangle`.

```
23 \_def\_bibp{\_the\_bibpart/} % unique name for each bibliography list
```

`\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,,}
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}{\_bibnn{#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:

- *<label>* was not declared by `\_Xbib` before and it is first such a *<label>* in the document: Then `\_bib:<bibpart>/<label>` is undefined and we save label using `\_addcitelist`, write warning on the terminal and define `\_bib:<bibpart>/<label>` as empty.
- *<label>* was not declared by `\_Xbib` before but it was used previously in the document: Then `\_bib:<bibpart>/<label>` is empty and we do nothing (only data to `\_savedcites` are saved).
- *<label>* was declared by `\_Xbib` before and it is first such *<label>* used in the document: Then `\_bib:<bibpart>/<label>` includes `\_bibnn{<number>}&` and we test this case by the command `\if &\_bibnn{<number>}&`. This is true when `\_bibnn{<number>}` expands to empty. The *<label>* is saved by `\_addcitelist` and `\_bib:<bibpart>/<label>` is re-defined directly as *<number>*.
- *<label>* was declared by `\_Xbib` and it was used previously in the document. Then we do nothing (only data to `\_savedcites` are saved).

The `\_citeA` macro runs repeatedly over the whole list of *<labels>*.

cite-bib.opm

```
99 \_def\_citeA #1#2,{\_if#1,\_else
100 \_if **1\_addcitelist{*}\_sxdef{\_bib:\_bibp*}{\_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 \_sxddef{\_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  $\backslash\_ctwrite:\langle bibpart \rangle/$ . 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 T<sub>E</sub>X run.

cite-bib.opm

```

137 \_def\_addcitelist#1{%
138 \_unless \_ifcsname\_ctlst:\_bibp\_endcsname \_sxddef{\_ctlst:\_bibp}{\_fi
139 \_ifcsname\_ctwrite:\_bibp\_endcsname \_openref \_immediate\_wref\_Xcite{\_bibp}{#1}}%
140 \_else \_global \_ea\_addto \_csname\_ctlst:\_bibp\_endcsname {\_citeI[#1]}\_fi
141 }
142 \_def\_Xcite#1#2{%
143 \_unless \_ifcsname\_ctlstB:#1\_endcsname \_sxddef{\_ctlstB:#1}{\_fi
144 \_global \_ea\_addto \_csname\_ctlstB:#1\_endcsname {\_citeI[#2]}%
145 }

```

The  $\langle bib-marks \rangle$  (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  $\backslash\_shortcitations$  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  $\backslash\_shortcitations$  sets simply  $\backslash\_lastcitenum=1$ . The macros for  $\langle bib-marks \rangle$  printing follows (sorry, without detail documentation). They are documented in opmac-d.pdf (but only in Czech).

cite-bib.opm

```

161 \_def\_printsavedcites{\_sortcitesA
162 \_chardef\_tmpb=0 \_ea\_citeB\_savedcites,%
163 \_ifnum\_tmpb>0 \_printdashcite{\_the\_tmpb}\_fi
164 }
165 \_def\_sortcitesA{}
166 \_def\_sortcitations{%
167 \_def\_sortcitesA{\_edef\_savedcites{300000,\_ea}\_ea\_sortcitesB\_savedcites,%
168 \_def\_tmpa####1300000,{\_def\_savedcites{####1}\_ea\_tmpa\_savedcites}%
169 }
170 \_def\_sortcitesB #1,{\_if $#1$%
171 \_else
172 \_mathchardef\_tmpa=#1
173 \_edef\_savedcites{\_ea}\_ea\_sortcitesC \_savedcites\_end
174 \_ea\_sortcitesB
175 \_fi
176 }
177 \_def\_sortcitesC#1,{\_ifnum\_tmpa<#1\_edef\_tmpa{\_the\_tmpa,#1}\_ea\_sortcitesD
178 \_else\_edef\_savedcites{\_savedcites#1,}\_ea\_sortcitesC\_fi}
179 \_def\_sortcitesD#1\_end{\_edef\_savedcites{\_savedcites\_tmpa,#1}}
180
181 \_def\_citeB#1,{\_if $#1$ \_else
182 \_if?#1\_relax??%
183 \_else
184 \_ifnum\_lastcitenum=0 % only comma separated list
185 \_printcite{#1}%
186 \_else
187 \_ifx\_citesep\_empty % first cite item
188 \_lastcitenum=#1\_relax

```



```

189     \_printcite{#1}%
190     \_else           % next cite item
191     \_advance\_lastcitenum by1
192     \_ifnum\_lastcitenum=#1\_relax % cosecutive cite item
193     \_mathchardef\_tmpb=\_lastcitenum
194     \_else % there is a gap between cite items
195     \_lastcitenum=#1\_relax
196     \_ifnum\_tmpb=0 % previous items were printed
197     \_printcite{#1}%
198     \_else
199     \_prindashcite{\_the\_tmpb}\_printcite{#1}\_chardef\_tmpb=0
200 \_fi\_fi\_fi\_fi\_fi
201 \_ea\_citeB\_fi
202 }
203 \_def\_shortcitations{\_lastcitenum=1 }
204
205 \_def\_printcite#1{\_citesep
206   \_ilink[cite:\_bibp#1]{\_citelinkA{#1}}\_def\_citesep{\_hskip.2em\_relax}}
207 \_def\_prindashcite#1{\_ifmmode-\_else\_hbox{--}\_fi\_ilink[cite:\_bibp#1]{\_citelinkA{#1}}}
208 \_def\_citesep{}
209
210 \_def\_nonumcitations{\_lastcitenum=0\_def\_sortcitesA{\_def\_etalchar##1{$^{##1}$}%
211   \_def\_citelinkA##1{\_trycs{\_bim:\_bibp##1}
212     {##1\_opwarning{\_noexpand\nonumcitations + empty bibmark. Maybe bad bib-style}}}%
213 }
214 \_def\_citelinkA{}
215
216 \_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.

cite-bib.opm

```

232 \_def\_bib{#1}{\_def\_tmp{\_isnextchar={\_bibA{#1}}{\_bibmark={\_bibB{#1}}}%
233   \_nospaceafter\_tmp} % ignore optional space
234 \_def\_bibA{#1}=#2{\_bibmark={#2}\_bibB{#1}}
235 \_def\_bibB{#1}{\_par \_bibskip
236   \_bibgl\_advance\_bibnum by1
237   \_noindent \_def\_tmpb{#1}\_dbib{#1}\_wbib{#1}{\_the\_bibnum}{\_the\_bibmark}%
238   \_printbib \_ignorespaces
239 }
240 \_def\_dbib#1{\_dest[cite:\_bibp\_the\_bibnum]\_printlabel{#1}}
241 \_def\_wbib#1#2#3{%
242   \_ifx\_wref\_wrefrelax\_else \_immediate\_wref\_Xbib{\_the\_bibpart}{#1}{#2}{#3}\_fi
243   \_unless \_ifcsname bib:\_bibp#1\_endcsname \_Xbib{\_the\_bibpart}{#1}{#2}{#3}\_fi
244 }
245 \_let\_bibgl=\_relax
246
247 \_public \bib ;

```

The `\_printbib` prints the bib-entry itself. You can re-define it if you want a different design. The `\_prtib` starts in horizontal mode after `\_noindent` and after the eventual hyperlink destination is inserted. By default, the `\_printbib` sets the indentation by `\_hangindent` and prints numeric *<bib-marks>* by `\_llap{[\_the\_bibnum]}`. If `\_nonumcitations` then the `\_citelinkA` is not empty and *<bib-marks>* (`\_the\_bibnum` nor `\_the\_bibmark`) are not printed. The text of bib-entry follows. User can create this text manually using `\bib` command or it is generated automatically from a `.bib` database by `\usebib` command.

The vertical space between bib-entries is controlled by `\_bibskip` macro.

cite-bib.opm

```

264 \_def \_printbib {\_hangindent=\_iindent
265   \_ifx\_citelinkA\_empty \_hskip\_iindent \_llap{[\_the\_bibnum]} \_fi
266 }

```

```
267 \def \bibskip {\ifnum\bibnum>0 \smallskip \fi}
```

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.

`cite-bib.opm`

```
277 \def\usebib{\_par \opinput {usebib.opm} \_usebib}
278 \def\usebib{\_usebib}
```

`\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.

`cite-bib.opm`

```
288 \def\nobibwarnlist{,}
289 \def\nobibwarning[#1]{\_global\_addto\nobibwarnlist{#1,}}
290 \public \nobibwarning ;
```

## 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 sorted usually by authors and years), `⟨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 document). 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` file 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.

Users may redefine declarations and the formatting rules given by the macros from the style file. Such a re-definition have to be included in the `\bibtexhook` token list, because the `\usebib` macro opens group, reads the macros from the style file, then executes `\bibtexhook` (it is empty by default), then reads data from the `.bib` files, then prints the desired records and finally, it closes the group.

For example, `\bibtexhook={\oldaccents}` can be set if your old `.bib` files use an obscure notation for accents like `{\“o}`. Recommendation: converting such old `.bib` files to Unicode encoding is much more conceptual solution of this problem.

## 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. More field-names can behave equally if the `\_fieldalias{⟨new-field-name⟩}{⟨given-field-name⟩}` is used in a style file. If a `⟨new-field-name⟩` is declared by this command and it is used in the `.bib` file then the effect is the same as if it was used the `⟨given-field-name⟩`.

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 page <http://en.wikipedia.org/wiki/BibTeX>. The set of entry types listed on this www 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  $\langle entry-type \rangle$ . The printing rules for  $\langle entry-type \rangle$  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 2.32.6.

The printing rules for each  $\langle entry-type \rangle$  must be declared by `\_sdef{\_print:⟨entry-type⟩}` in `bib-⟨style⟩.opm` file. The  $\langle entry-type \rangle$  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  $\langle if defined \rangle$  is executed if  $\langle field-name \rangle$  is declared in .bib file for the entry which is currently processed. Else the part  $\langle if not defined \rangle$  is processed. The part  $\langle if defined \rangle$  can include the `*` parameter which is replaced by the value of the  $\langle field-name \rangle$ .
- The part  $\langle if not defined \rangle$  can include the `\_bibwarning` command if the  $\langle field-name \rangle$  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  $\langle if non-empty \rangle$  part is executed if `\macro` is non-empty. The `*` parameter can be used, it is replaced by the `\macro`.
- `\_bprintv [⟨field1⟩,⟨field2⟩,...] {⟨if defined⟩} {⟨if not defined⟩}`. The part  $\langle if defined \rangle$  is executed if  $\langle field1 \rangle$  or  $\langle field2 \rangle$  or ... is defined, else the second part  $\langle if not defined \rangle$  is executed. There is one field name or the list field names separated by commas. The parts cannot include any 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  $\langle entry-type \rangle$ . The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by `\_printbib` macro used in OpTeX (and may be redefined by the author or document designer).

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 TeX 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 processed 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
- `\_namecount`: 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.⟨identifier⟩}`, where  $\langle identifier \rangle$  is an identifier of the phrase and the phrase itself is defined by `\_sdef{\_mt:bib.⟨identifier⟩:⟨language⟩}{⟨phrase⟩}`. See section 2.37.2 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 2.37.2.

The `sortedby` field is declared by `\readbibs` as a special field where sorting phrase can be specified. If it is present then it has precedence before default sorting phrase generated by `\_preparebibsoring` from the lastname, firstnames of the first author and from the year. 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`.

If you want to declare a different sorting rule, you can re-define the `\_preparebibsorting` macro. The example is in the [OpTeX trick 0113](#).

## 2.32.4 Direct reading of .bib files

`\readbibs`  $\langle bib-bases \rangle$  is internally used (by `\usebib`) for reading `.bib` databases in BibTeX format. The  $\langle bib-bases \rangle$  is comma separated list of file names (without `.bib` extension, without spaces). These files are read and `\readbibs` defines macros `\_be:⟨bibpart⟩/⟨label⟩`, where  $\langle label \rangle$  is the label of the reference record. These macros include key-value pairs  $[\langle field name \rangle]\{\langle field data \rangle\}$ . The first pair is  $[@]\{\langle entry type \rangle\}$ . For example, if we have in the `.bib` file:

```
@Book { tbn,
  author = "Petr Olšák",
  TITLe   = {\TeX{}}book naruby},
  publisher = "Konvoj",
  year    = 2001,
}
```

and the `\bibpart` is empty (default value) then the `\_be:/tbn` macro is defined with the content:

```
[@]{BOOK}[author]{Olšák}{Petr}{}}[authornum]{1}[title]{\TeX{}}book naruby}%
[publisher]{Konvoj}[year]{2001}
```

If you do `\slet\tmp{\_be:/tbn}` then you can print the data (for example) by:

```
\ea\foreach \tmp \do [#1]#2{\wterm{field-name: "#1", data: "#2"}}
```

or you can do `\ea\foreach \tmp \do [#1]#2{\sdef{bib-field:#1}{#2}}` to enable direct access to the scanned data.

Note that entry type and field names are converted lower-case by the `\readbibs` macro.

There are two special entry types: `@COMMENT{⟨ignored text⟩}` and `@TEXCODE{⟨processed text⟩}`. The  $\langle ignored text \rangle$  is ignored, the  $\langle processed text \rangle$  is executed by TeX. The definitions of macros used in other entries in data of fields can be here. If the `\usebib` is used then the  $\langle processed text \rangle$  is executed inside a TeX group, so the assignment is locally valid only during creating the reference list. The BibTeX's `@STRING{}` isn't supported. All others entry types are interpreted as a *reference entry* and they are interpreted as described above. An optional balanced text between entries in `.bib` files is ignored.

If the macro `\_be:⟨bibpart⟩/*` is defined then the `\readbibs` macro reads all entries from `.bib` files and creates `\_citelist`. If the `\_be:⟨bibpart⟩/*` is undefined then the `\readbibs` macro reads only entries with  $\langle label \rangle$  where `\_be:⟨bibpart⟩/⟨label⟩` is set to the empty macro. After reading, the macros `\_be:⟨bibpart⟩/⟨label⟩` are globally re-defined as described above.

The `\readbibs` macro doesn't convert fields data, but there are two exceptions: author and editor fields. These fields have very specific format with various alternatives, see <https://nwalsh.com/tex/texhelp/bibtex-23.html>. Shortly speaking, more authors are divided by the `and` keyword and names of a single author must be separated to four subfields:  $\langle Lastnames \rangle$ ,  $\langle Firstnames \rangle$ ,  $\langle Von \rangle$ ,  $\langle Junior \rangle$ . Only the  $\langle Lastnames \rangle$  subfield must be nonempty. The input can look like

```
Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
```

and both these variants are converted to `{Vinci}{Leonardo Piero}{da}{}`. The  $\langle Von \rangle$  part is recognized as a word with only lowercase letters. In general, the name can be written without commas:  $\langle Firstnames \rangle$   $\langle Von \rangle$   $\langle Lastnames \rangle$  or with single comma:  $\langle Von \rangle$   $\langle Lastnames \rangle$ ,  $\langle Firstnames \rangle$  or with two commas:  $\langle Von \rangle$   $\langle Lastnames \rangle$ ,  $\langle Junior \rangle$ ,  $\langle Firstnames \rangle$  and all these variants are converted to the quaternion  $\{\langle Lastnames \rangle\}\{\langle Firstnames \rangle\}\{\langle Von \rangle\}\{\langle Junior \rangle\}$  by the `\readbibs` macro. If there are more than single author, then each author is saved in four subfields side by side, so you have 4 or 8 or 12 etc. subfields in the author/editor data field. You can read them by `\foreach ⟨author-data⟩\do #1#2#3#4{...}`.

## 2.32.5 The usebib.opm macro file loaded when \usebib is used

```
3 \_codedecl \readbibs {Reading bib databases <2026-05-10>} % loaded on demand by \usebib
```

usebib.opm

First, we implement the scanner of .bib files. Unfortunately, the format of these files isn't TeX friendly, so we must do more work. `\readbibs`  $\langle bib-bases \rangle$  reads  $\langle bib-bases \rangle$  files (i.e. the BibTeX format).

usebib.opm

```

12 \_newcount\_aunum
13 \_newcount\_NameCount
14 \_gdef\_eaddto#1#2{\_ea\_addto\_ea#1\_ea{#2}}
15
16 \_gdef\_readbibs #1{%
17   \_ifcsname\_be:\_bibp*\_endcsname \_def\_citelist{}\_fi % \_citelist will be created
18   \_begingroup
19     \_everyeof{@{}}\_foreach#1,\_do#1,{%
20       \_isfile{##1.bib}\_iftrue \_ea\_nextat\_input{##1.bib}
21       \_else \_opwarning{\_string\usebib: Missing ##1.bib file}
22       \_fi}%
23   \_endgroup
24 }
25 \_public \_readbibs ;

```

The `\_nextat` macro skips the text in the .bib file to the next @, and starts the `\_bibentry` macro which reads  $\langle entry\ type \rangle \langle data \rangle$  from the .bib file. Each reference entry is converted to the `\_entrydata` macro and then `\_glet \_be:\langle bib-part \rangle / \langle label \rangle = \_entrydata` is done. The `\_entrydata` includes key-value pairs, as described in the section 2.32.4.

usebib.opm

```

36 \_long\_gdef\_nextat#1@{\_bgroup\_catcode`=9 \_ea\_egroup\_bibentry}
37 \_gdef\_bibentry #1#{\_ifx~#1~\_else \_afterfi{\_bibentryA{#1}}\_fi}
38 \_gdef\_bibentryA #1#2{\_lowercase{\_def\_entrytype{#1}}}%
39   \_ismacro\_entrytype{comment}\_iffalse % comment is ignored
40   \_ismacro\_entrytype{texcode}\_iftrue % TeX code is processed
41   \_endgroup #2\_begingroup \_everyeof{@{}}\_else
42   \_ismacro\_entrytype{string}\_iftrue % string is reported as unsupported
43   \_opwarning{\_string\usebib: @STRING entry isn't supported, try to use @TEXCODE}%
44   \_else
45     \_edef\_entrydata{[@]{\_entrytype}}%
46     {\_bibentryB #2\_fin}% % read a "normal" bib entry
47   \_fi\_fi\_fi
48   \_nextat
49 }
50 \_gdef\_bibentryB #1#2,\_3\_fin{\_def\_citekey{#1#2}\_def\_bibentryC{\_nextfield #3,\_fin}%
51   \_ifcsname\_be:\_bibp*\_endcsname
52     \_bibentryC
53     \_global\_addto\_citelist{\_citeI[#1#2]}
54   \_else
55     \_ea\_ifx \_begincsname\_be:\_bibp#1#2\_endcsname \_empty
56     \_bibentryC
57   \_fi\_fi
58 }
59 \_gdef\_bibentryF {% finalize entry
60   \_preparebibsortng
61   \_global\_ea\_let \_csname\_be:\_bibp\_citekey\_endcsname = \_entrydata
62 }

```

`\_nextfield` reads next field name and saves it to the `\_fieldname` and then reads field data and saves it to the `\_fielddata`.

usebib.opm

```

69 \_gdef\_nextfield #1{\_ifx,#1\_ea\_nextfield % skip commas from previous field
70   \_else \_ifx\_fin#1\_ea\_ea\_ea\_bibentryF % finalize bib entry
71   \_else \_def\_fieldname{#1}\_ea\_ea\_ea\_nextfieldA % first letter of fieldname found
72   \_fi\_fi
73 }
74 \_gdef\_nextfieldA #1{% next letters of field name until = is found
75   \_ifx=#1\_afterfi{\_nospacefuturelet\_next\_nextfieldB}%
76   \_else \_addto\_fieldname{#1}%
77   \_ea\_nextfieldA \_fi
78 }
79 \_gdef\_nextfieldB {% reading field data
80   \_casesof\_next
81     " {\_nextfieldC}% name = "data",
82   \_bgroup {\_nextfieldD}% name = {data},
83   \_finc {\_nextfieldE}% name = data,

```

```

84 }
85 \gdef\nextfieldC "#1"{\nextfieldD{#1}}
86 \gdef\nextfieldD #1{\def\fielddata{#1}\nextfieldF}
87 \gdef\nextfieldE #1,{\nextfieldD{#1}}
88
89 \gdef\nextfieldF{% finalize field
90   \lowercase\ea{\ea\def\ea\fieldname\ea{\fieldname}}% case insensitive field name
91   \ifcsname _fia:\fieldname\endcsname \edef\fieldname{\cs{_fia:\fieldname}}\fi
92   \eaddto\entrydataf{\ea{\fieldname}}%
93   \ismacro\fieldname{author}\iftrue \ea\auscan\ea{\fielddata}[author]\else
94   \ismacro\fieldname{editor}\iftrue \ea\auscan\ea{\fielddata}[editor]\else
95   \eaddto\entrydataf{\ea{\fielddata}}\fi\fi
96   \sdef{fd:\fieldname\ea}\ea{\fielddata}%
97   \nextfield
98 }

```

`\_fielddalias{⟨new-name⟩}{⟨given-name⟩}` defines `\_fia:⟨new_name⟩` as `⟨given-name⟩`.

usebib.opm

```
105 \gdef\_fielddalias#1#2{\lowercase{\sdef\_fia:#1}{#2}}
```

The `\_auscan{⟨authors/editors-names⟩}[⟨field-name⟩]` reads the specific BibTeX format mentioned in section 2.32.4 and converts them to `{⟨Lastname⟩}{⟨Firstname⟩}{⟨Von⟩}{⟨Junior⟩}` for each author/editor. The result includes 4k subfields (where k is number of the authors/editors) and it is saved to the `\_entrydata` and the `[authornum]{k}` or `[editornum]{k}` is added.

The `\_auscanA` macro does the loop over authors separated by `and`. Each single author has its `\_tmpb` macro with X and x. Each letter corresponds to single word of the name (X: begins with uppercase, x: begins with lowercase). For example Leonardo Piero da Vinci has `\_tmpb` macro `XXxX..` If there are commas in after some words, then these commas are in `\_tmpb` macro too, for example da Vinci, Piero Leonardo has its `\_tmpb` macro `xX,XX..` The number of commas is saved to `\_tmpnum`. The `\_auscanB` macro does a slight modifications of the `\_tmpb` macro as mentioned in comments. Then the macro `\_auscanD(tpmb-pattern);{⟨WordA⟩}{⟨WordB⟩}{⟨WordC⟩}...` is executed. It saves given words due to the `\_tmpb` pattern to the macros `\_Lastname`, `\_Firstname`, `\_Von`, `\_Junior` in a loop. Finally, the contents of these macros are saved to `\_fielddata` and then to the `\_entrydata`.

usebib.opm

```

130 \gdef\_auscan#1[#2]{\def\_auname{}\def\_fielddata{}\def\_tmpb{}\_aunum=0\_tmpnum=0
131   \_auscanA #1 and {}
132   \eaddto\entrydataf{\ea{\fielddata}[#2num]}%
133   \eaddto\entrydataf{\ea{\_the\_aunum}}%
134 }
135 \gdef\_auscanA #1 {%
136   \ifx^#1~\else
137     \isequal{#1}{and}\iftrue
138       \incr\_aunum
139       \addto\_tmpb{.}%
140       \auscanB
141       \ea\_auscanX \_auname
142       \def\_auname{}\def\_tmpb{}\_tmpnum=0
143     \else
144       \lowercase{\isequal{#1}{#1}}\iftrue \addto\_tmpb{x}\else \addto\_tmpb{X}\fi
145       \def\_tmp{#1~}\isinlist\_tmp{,}%
146       \iftrue
147         \ea\_auscanC\_tmp \addto\_tmpb{,}\incr\_tmpnum
148       \else
149         \def\_tmp{#1}%
150       \fi
151       \eaddto\_auname{\ea{\_tmp}}%
152       \fi
153       \ea\_auscanA
154     \fi
155 }
156 \gdef\_auscanB{%
157   \ifcase\_tmpnum % 0 commas: XXX. -> XX:X. ; XxxXX. -> X:xxXX. ; xXX. -> :xXX. ; First:Last
158   \isinlist\_tmpb{x}\iffalse \replstring\_tmpb{X.}{:X.}\else
159   \ea\_auscanT\_tmpb;\iffalse \replstring\_tmpb{Xx}{X:x}\fi\fi
160   \or % 1 comma: XX,XXX -> XX,,XXX, Junior part is empty
161   \replstring\_tmpb{,}{,,}\_tmpnum=2
162   \fi % 2 commas: XX,XX,XXX no changes, generic format: Last, Junior, First

```



```

163 \_def\_Firstname{}\\_def\_Lastname{}\\_def\_Von{}\\_def\_Junior{}%
164 }
165 \_gdef\_auscanT #1#2;\_iffalse{\_ifx#1x\_def\_tmpb{:x#2}\_else} % xXX. -> :xXX.
166 \_gdef\_auscanX {\_ea\_auscanD\_tmpb;}
167 \_gdef\_auscanD #1#2;{%
168 \_def\_tmpb{#2}%
169 \_casesof #1
170 . {\_auscanF} % dot is last character, do final job
171 , {\_decr\_tmpnum \_auscanX} % Lastname->Junior or Junior->Firstname
172 : {\_tmpnum=2 \_auscanX} % Firstname->Lastname
173 X {\_auscanE\_Firstname\_Junior\_Lastname} % add data due to the \tmpnum value
174 x {\_auscanE\_Firstname\_Junior\_Von} % Von instead Lastname
175 \_finc {}%
176 }
177 \_gdef\_auscanE#1#2#3{\_ifcase\_tmpnum \_ea\_auscanS\_ea#1\_or \_ea\_auscanS\_ea#2\_else
178 \_ea\_auscanS\_ea#3\_fi}
179 \_gdef\_auscanS#1#2{% #1=\Firstname or \Lastname or etc., #2=word to be inserted
180 \_ifx#1\_empty \_def#1{#2}\_else \_addto#1{ #2}\_fi
181 \_auscanX
182 }
183 \_gdef\_auscanF{% final work of \auscanX
184 \_eaddto\_fielddata{\_ea{\_Lastname}}\_eaddto\_fielddata{\_ea{\_Firstname}}%
185 \_eaddto\_fielddata{\_ea{\_Von}}\_eaddto\_fielddata{\_ea{\_Junior}}%
186 }
187 \_gdef\_auscanC #1,~{\_def\_tmp{#1}} % removing final comma: Word,^ -> Word

```

The `\_citelist` includes `\_citeI[⟨label⟩]` commands. The `\usebib` macro runs this lists in order to print references. Each `\_citeI[⟨label⟩]` prints single bib entry given by the `⟨label⟩`. It opens a group, sets macros `\_fd:⟨field-name⟩` to `⟨field-data⟩` and runs `\_printentry`. Finally, it closes TeX group, so all macros `\_fd:⟨field-name⟩` have their initial (undefined) value.

The `\_getfield[⟨field-name⟩]` macro does `\def\macro{⟨field-data⟩}`. If the field isn't declared then the `\macro` is empty.

usebib.opm

```

200 \_gdef\_citeI[#1]{%
201 \_begingroup
202 \_ea\_ifx \_begincsname\_be:\_bibp#1\_endcsname \_empty
203 \_opwarning{\_string\usebib: entry [#1] isn't found in .bib}%
204 \_global\_slet{\_bes:#1}{\_relax}%
205 \_else
206 \_ea\_ea\_ea\_foreach \_csname\_be:\_bibp#1\_endcsname \_do{##1}##2{\_sdef\_fd:##1}{##2}}%
207 \_def\_entrykey{#1}%
208 \_printentry
209 \_fi
210 \_endgroup
211 }
212 \_gdef\_getfield[#1]#2{%
213 \_ifcsname\_fd:#1\_endcsname
214 \_ea\_ea\_ea \_def \_ea\_ea\_ea #2\_ea\_ea\_ea {\_csname\_fd:#1\_endcsname}%
215 \_else \_def#2{}%
216 \_fi
217 }

```

`\_preparebibsorting` is called repeatedly for each bib entry when its reading from `.bib` file is finished. Its main goal is to do `\gdef\_bes:⟨citekey⟩ {\;⟨sorting-rule⟩^^⟨citekey⟩}`. Note that the part of the control sequence name after `^^` is ignored during sorting. The default `\_preparebibsorting` macro creates `⟨sorting-rule⟩` in the form: `⟨Lastnames⟩ ⟨Firstnames⟩ ⟨Von⟩ ⟨Junior⟩` of the first author followed by `⟨year⟩` from year field.

`\_dobibsorting\_citelist` sorts the `\_citelist` and runs it.

usebib.opm

```

230 \_gdef\_preparebibsorting{%
231 \_getfield[sortedby]\_sortedby
232 \_ifx\_sortedby\_empty % explicitly given [sortedby] field has precedence
233 \_def\_sortedby{}%
234 \_getfield[author]\_tmp % sorting by author firstly
235 \_ifx\_tmp\_empty \_else \_ea\_preparebibsortingA\_tmp \_fin \_fi
236 \_getfield[year]\_tmp % soering by year secondly
237 \_ifx\_tmp\_empty \_else \_eaddto\_sortedby{\_tmp}\_fi

```

```

238     \edef\sortedby{\sortedby}% we need to run macros aka \"e etc.
239     \edef\sortedby{\ea\removeoutbraces\sortedby {\_fin}}}% remove braces
240     \_fi
241     \_sxdef{\_bes:\citekey\_ea}\_ea{\_csname;\_detokenize\_ea{\_sortedby}^^\_citekey\_endcsname}%
242 }
243 \_gdef\preparebibsoringA#1#2#3#4#5\_fin {% names of the first author used by sorting:
244     \_def\sortedby{#1 }%                Lastname
245     \_ifx^#2\_else \_addto\_sortedby{#2 }\_fi % Firstname
246     \_ifx^#3\_else \_addto\_sortedby{#3 }\_fi % Von
247     \_ifx^#4\_else \_addto\_sortedby{#4 }\_fi % Junior
248 }
249 \_gdef\_dobibsorting{%
250     {\_def\_citeI[#1]{\_ea\_citeIs\_csname\_bes:##1\_endcsname{##1}}}%
251     \_edef\_citelist{\_ea}\_citelist % converting \_citelist
252     \_dosorting\_citelist \_ea}%
253 }
254 \_gdef\_citeIs#1#2{\_eaddto\_citelist{#1}%
255     \_ea\_gdef#1{\_citeI[#2]\_ea\_glet#1=\_undefined \_glet#1=\_undefined}%
256 }

```

The `\_thisentry` macro prints bibliographic reference entry. It prints  $\langle bibnum \rangle$  or  $\langle bemark \rangle$  (including hyperlinks) and they are followed by printing the entry data. The format is given by the `\_printbib` macro and by `\_print:⟨entrytype⟩` declared in the bib-style file.

usebib.opm

```

265 \_gdef\_thisentry {\_par \_bibskip
266     \_bibgl\_incr\_bibnum
267     \_isdefined{\_bim:\_bibp\_the\_bibnum}\_iftrue
268     \_edef\_tmpb{\_csname\_bim:\_bibp\_the\_bibnum\_endcsname}%
269     \_bibmark=\_ea{\_tmpb}%
270     \_else \_bibmark={}\_fi
271     \_edef\_tmpb{\_entrykey}%
272     \_noindent \_dbib\_entrykey
273     \_printbib
274     {%
275         \_getfield[@]\_entrytype
276         \_csname\_print:BEGIN\_endcsname
277         \_isdefined{\_print:\_entrytype}\_iftrue
278         \_csname\_print:\_entrytype\_endcsname
279         \_else
280             \_ifx\_entrytype\_empty \_else
281                 \_opwarning{Entrytype @\_entrytype\_space from [\_entrykey] undefined}%
282                 \_csname\_print:misc\_endcsname
283             \_fi\_fi
284             \_csname\_print:END\_endcsname
285             \_wbib \_entrykey {\_the\_bibnum}{\_the\_bibmark}%
286     }\_par
287 }

```

`\_printentry` is equivalent to the `\_thisentry` macro by default, it means that no filter is applied and all cited entries are printed. We can re-define `\_printentry` in order to implement a filter which prints only selected entries. It can be usable if we want to divide the bibliography list to more sections. For example, suppose that a `test.bib` file includes several entries with `entrysubtype` field and it can have a value `primary`. We want to print these entries in a special subsection and then to print others entries. We can do following:

```

\def\isfield[#1]#2{\_getfield[#1]\tmp \_ismacro\tmp{#2}}

\nonum\secc Primary Sources
\def\_printentry{\isfield[entrysubtype]{primary}\iftrue \_thisentry \_fi}
\usebib/s (simple) test % only entrysubtype=primary entries are printed

\nonum\secc Secondary Sources
\def\_printentry{\isfield[entrysubtype]{primary}\iffalse \_thisentry \_fi}
\usebib/s (simple) test % other cited entries are printed

```

The `\_bprinta`, `\_bprintb`, `\_bprintc`, `\_bprintv` commands used in the style files:

```

316 \_gdef\_bprinta {\_bprintb*}
317 \_gdef\_bprintb #1[#2#3]{%
318   \_def\_bibfieldname{#2#3}%
319   \_if!#2\_relax
320     \_def\_bibfieldname{#3}%
321     \_getfield[#3]\_bibfield
322     \_getfield[#3num]\_namecount % total persons in the author/editor fields
323     \_ifx\_bibfield\_empty\_else
324       \_def\_bibfield{\_loopauthors{#3}}% read author/editor field in a loop
325       \_fi
326     \_else
327       \_getfield[#2#3]\_bibfield
328       \_fi
329     \_if^#1^%
330       \_ifx\_bibfield\_empty \_ea\_ea\_ea \_doemptyfield
331       \_else \_ea\_ea\_ea \_dofullfield \_fi
332     \_else \_ea \_bprintaA
333     \_fi
334 }
335 \_gdef\_dofullfield#1#2{\_def\_dofield##1{#1}\_ea\_dofield\_ea{\_bibfield}}
336 \_gdef\_doemptyfield#1#2{\_def\_dofield##1{#2}\_ea\_dofield\_ea{\_bibfield}}
337 \_gdef\_bprintaA #1#2{\_ifx\_bibfield\_empty #2\_else\_bprintaB #1**\_fin\_fi}
338 \_gdef\_bprintaB #1*#2*#3\_fin{\_ifx^#3^#1\_else\_ea\_bprintaC\_ea{\_bibfield}{#1}{#2}\_fi}
339 \_gdef\_bprintaC #1#2#3{#2#1#3}
340 \_gdef\_bprintc#1#2{\_bprintcA#1#2**\_relax}
341 \_gdef\_bprintcA#1#2*#3*#4\_relax{\_ifx#1\_empty \_else \_if^#4^#2\_else#2#1#3\_fi\_fi}
342 \_gdef\_bprintv [#1]#2#3{\_def\_tmpa{#2}\_def\_tmpb{#3}\_bprintvA #1,,}
343 \_gdef\_bprintvA #1,{%
344   \_if^#1^\_tmpb\_else
345     \_getfield[#1]\_tmp
346     \_ifx \_tmp\_empty
347     \_else \_tmpa \_def\_tmpb{}\_def\_tmpa{}%
348     \_fi
349   \_ea \_bprintvA
350   \_fi
351 }

```

`\_loopauthors{<field-name>}` does a loop over all authors/editors in the `author` or `editor` field. The `\_namecount` (total number of authors/editors) was defined in `\_bprintb`. Then for each author/editor it do:

- Set `\_NameCount` to the position number of the currently processed author/editor.
- Define `\_Lastname`, `\_Firstname`, `\_Junior`, `\_Von`, `\_After` macros.
- Run `\_authorname` or `\_editorname` macro (defined in the bib style file).

```

365 \_gdef\_loopauthors #1{%
366   \_NameCount=0
367   \_ea\_ea\_ea\_foreach\_csname \_fd:#1\_endcsname \_do ##1##2##3##4{%
368     \_advance\_NameCount by1
369     \_def\_Lastname{##1}\_def\_Firstname{##2}\_def\_Von{##3}\_def\_Junior{##4}%
370     \_csname \_#1ini\_endcsname \_csname \_#1name\_endcsname
371   }}
372 \_gdef\_authorini{} % ready for \_AbbreviateFirstname or similar...
373 \_gdef\_editorini{}

```

`\_bibwarning` can be used if the mandatory field is missing. Note that `\_nobibwarnlist` is used here, it is set by `\_nobibwarning` macro.

```

380 \_gdef\_bibwarning{%
381   \_ea\_isinlist \_ea\_nobibwarnlist\_ea{\_ea,\_entrykey,}\_iffalse
382   \_opwarning{Missing field "\_bibfieldname" in [\_entrykey]}\_fi}

```

`\_AbbreviateFirstname`, `\_RetrieveFieldIn`, `\_RetrieveField` are here only for backward compatibility with previous macros based on the librarian package. The `\_CreateField`, `\_SortingOrder`, and `\_SpecialSort` are dummy macros because the sorting is implemented by a slightly different way than in librarian package.

```

392 \_gdef\_AbbreviateFirstname{\_addto\_authorini{\_abbrevnames\_Firstname}}
393 \_gdef\_abbrevnames#1{% Karolina Pondelickova-Maslova -> K. P.-M.
394   \_edef#1{\_ea\_foreach #1 \_do ##1##2 {##1.%
395     \_foreach ##2-{}\_do #####2{\_ifx^####2\_else-####2.\_fi} }~%
396   }\_replstring#1{ ~}{}%
397 }
398 \_gdef\_RetrieveFieldIn#1{\_getfield[#1]}
399 \_gdef\_RetrieveField#1{\_trycs{\_fd:#1}{}}
400 \_gdef\_CreateField#1{}
401 \_gdef\_SortingOrder#1#2{}
402 \_gdef\_SpecialSort#1{}

```

The `\usebib` command is defined as `\input{usebib.opm}\_usebib` in the format. So, the command is re-defined here and it is run again with the new meaning.

The `\usebib` macro defined here reads `\_ctlst:<bibpart>/` and `\_ctlstB:<bibpart>/` (they include a list of `\_citeI[<label>]`) and merges them to a single `\_citelist`. The `\_be:<bibpart>/<label>` is set to empty for each member of the `\_citelist`. Then the style file is read in a group, the `\readbibs` macro reads given `.bib` files and resulting `\_citelist` is processed: i.e. the macros `\_citeI` print desired entries.

```

416 \_gdef\_usebib/#1 (#2) #3 {%
417   \_ifcsname\_ctlst:\_bibp\_endcsname
418     \_slet{\_citelist}{\_ctlst:\_bibp}\_else\_def\_citelist{}\_fi
419   \_ea\_foreach\_citelist\_do ##1[##2]{\_sdef{\_be:\_bibp##2}{}}%
420   \_ifcsname\_ctlstB:\_bibp\_endcsname
421     \_ea\_ea\_ea\_foreach\_csname\_ctlstB:\_bibp\_endcsname\_do ##1[##2]{%
422       \_ifcsname\_be:\_bibp##2\_endcsname
423       \_else\_addto\_citelist{\_citeI[##2]}\_sdef{\_be:\_bibp##2}{}%
424       \_fi
425     }%
426   \_fi
427   \_sxdef{\_ctwrite:\_bibp}{}%
428   \_ifx\_printentry\_undefined\_let\_printentry=\_thisentry\_fi
429   \_ifx\_citelist\_empty
430     \_opwarning{No cited items. \_noexpand\_usebib ignored}%
431   \_else
432     \_bgroup
433     \_par
434     \_emergencystretch=.3\_hsize
435     \_def\_optexbibstyle{#2}%
436     \_setctable\_optexcatcodes
437     \_input bib-#2.opm
438     \_the\_bibtexhook
439     \_ifcsname\_mt:bib.and:\_cs{\_lan:\_the\_language}\_endcsname\_else
440       \_opwarning{\_string\_usebib: No phrases for language
441         "\_cs{\_lan:\_the\_language}" (using "en")}%
442     \_language=0 \_chardef\_documentlanguage=0
443     \_fi
444     \_ifx#1c\_def\_preparebibsoring{}\_def\_dobibsortng{}\_fi
445     \_readbibs {#3}%
446     \_dobibsortng\_citelist
447     \_restorectable
448   \_egroup
449   \_fi
450 }

```

## 2.32.6 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␣=␣"value"`, or `name␣=␣{value}`. No matter which form is used. If the value is pure numeric then

you can say simply `name=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 = "auctal",
```

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 = "auctal 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\_ Engineering",
```

output: CZECH TECHNICAL UNIVERSITY IN PRAGUE, Faculty of Electrical Engineering.

### 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 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 in use).

### 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 selected language. Example:

```
citedate = "2004/05/21",
```

Output when `en` is used: [cit. 2004-05-21].

Output when `cs` is used: [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 filed 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 `@ONLINE` entry

It is intended for online publications.

Fields: author, title(!), howpublished, ednote, publisher, accessed, doi, url(!), note.

- 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 `@MICS` 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 `sortedby` (or `key` field with the same effect). It can be used for sorting instead of the “Lastname Firstname(s)” of the first author. If the `sortedby` field is unset then the “Lastname Firstname(s)” is used for sorting normally. Example:

```
author      = "Světla Čmejrková",
sortedby    = "Czzmejrkoa 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 `sortedby_="@"`, 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 BibTeX:

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.

## 2.32.7 Implementation of the bib-iso690 style

```

3 \_codedecl \_undefined {BIB style (iso690) <2025-04-15>} % 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
17 \_ifx\_undefined\_let\:=\_maybetod \_fi % for backward compatibility

```

Option field.

```

23 \_CreateField {option}
24 \_def\_isbiboption#1#2{\_edef\_tmp{\_noexpand\_isbiboptionA{#1}}\_tmp}
25 \_def\_isbiboptionA#1{\_def\_tmp##1 #1 ##2\_relax{%
26   \_if^##2\_csname iffalse\_ea\_endcsname \_else\_csname iftrue\_ea\_endcsname \_fi}%
27   \_ea\_tmp\_biboptionsi #1 \_relax}
28 \_def\_bibopt[#1]#2#3{\_isbiboption{#1}\_iftrue\_def\_tmp{#2}\_else\_def\_tmp{#3}\_fi\_tmp}
29 \_def\_biboptionvalue#1#2{\_def\_tmp##1 #1:##2 ##3\_relax{\_def#2{##2}}}%
30   \_ea\_tmp\_biboptionsi #1: \_relax}
31
32 \_def\_readbiboptions{%
33   \_RetrieveFieldIn{option}\_biboptionsi
34   \_toks1=\_ea{\_biboptionsi}%
35   \_edef\_biboptionsif\_space \_the\_toks1 \_space \_the\_biboptions \_space}%
36 }

```

Formatting of Author/Editor lists.

```

42 \def\firstauthorformat{%
43   \upper{\_Lastname}\_bprintc\_Firstname{, *}\_bprintc\_Von{ *}\_bprintc\_Junior{, *}%
44 }
45 \def\otherauthorformat{%
46   \_bprintc\_Firstname{* }\_bprintc\_Von{* }\_upper{\_Lastname}\_bprintc\_Junior{, *}%
47 }
48 \def\_commonname{%
49   \_ifnum\_NameCount=1
50     \firstauthorformat
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   \_ifx\_authlist\_undefined \_edef\_authlist{\_Lastname,\_Firstname,\_Von,\_Junior}%
60   \else \_edef\_authlist{\_authlist;\_Lastname,\_Firstname,\_Von,\_Junior}\_fi
61   \_ifnum\_NameCount>0\_namecount\_relax\_else \_commonname \_fi
62   \_ifnum\_NameCount=0\_namecount\_relax \_maybeetal \_fi
63 }
64 \def\_editorname{%
65   \_ifnum\_NameCount>0\_namecount\_relax\_else \_commonname \_fi
66   \_ifnum\_NameCount=0\_namecount\_relax \_maybeetal \_fi
67 }
68
69 \def\_prepareauedoptions#1{%
70   \def\_maybeetal{\\_csname lb@abbreviatefalse\_endcsname
71   \_biboptionvalue{#1max}\_authormax
72   \_biboptionvalue{#1min}\_authormin
73   \_biboptionvalue{#1pre}\_authorpre
74   \_biboptionvalue{#1print}\_authorprint
75   \_isbiboption{#1etal}\_iftrue \_def\_maybeetal{\_Mtext{bib.etal}}\_fi
76   \_biboptionvalue{#1trim}\_autrim
77   \_let\_namecount=\_namecount
78   \_ifx\_authormax\_empty \_else
79     \_ifnum 0\_authormax<0\_namecount
80     \_edef\_namecount{\_ifx\_authormin\_empty\_authormax\_else\_authormin\_fi}%
81     \def\_maybeetal{\_Mtext{bib.etal}}%
82   \_fi\_fi
83   \_ifx\_autrim\_empty \_def\_autrim{10000}\_fi
84   \_ifnum\_autrim=0 \_def\_autrim{10000}\_fi
85   \_ifnum 0\_namecount<\_autrim\_relax \_else \_AbbreviateFirstname \_fi
86 }
87 \def\_maybeetal{}
88
89 \_ifx\_upper\_undefined
90   \_ifx\_caps \_undefined \_def\_upper{\_uppercase\_ea}\_else
91     \_def\_upper#1{{\caps\_rm #1}}\_fi
92   \_fi
93 \_let\_upper=\upper

```

Preparing bib-mark (used when `\nonumcitations`). The `\_setbibmark` is run at the end of each record. The `\_authlist` includes Lastname,Firstname,Von,Junior of all authors separated by semicolon (no semicolon at the end of the list). If bibmark isn't declared explicitly then we create it by the `\_createbibmark<year>;<authors-list>;,;,\_fin` macro. It outputs first Lastname (and adds "et al." if the second author in the `<authors-list>` is non-empty). Then comma and `<year>` is appended. A user can redefine the `\_createbibark` macro in the `\bibtexhook` tokens list, if another bibmark format is needed. The macro `\_createbibmark` must be expandable. See also [OpTeX trick 0104](#).

```

110 \def\_setbibmark{%
111   \_ifx\_authlist\_undefined \_def\_authlist{,;}\_fi
112   \_RetrieveFieldIn{bibmark}\_tmp
113   \_ifx\_tmp\_empty
114     \_RetrieveFieldIn{year}\_tmp
115     \_edef\_tmp{\_ea\_createbibmark\_expanded{\_tmp;\_authlist};,;,\_fin}\_fi
116   \_bibmark=\_ea{\_tmp}%

```



```

117 }
118 \def\createbibmark #1;#2,#3;#4,#5\fin{% #1=year #2=LastName #3=FirstName #4=nextAuthor
119     #2\ifx^#4^_\else \Mtext{bib.etal}\_fi \ifx^#1^_\else, #1\_fi
120 }

```

Setting phrases.

bib-iso690.opm

```

126 \def\bibconjunctionand{\Mtext{bib.and}}
127 \def\preurl{\Mtext{bib.available}}
128 \let\predoi=\preurl
129 \def\postedition{\_mtext{bib.edition}}
130 \def\Inclause{In:~}
131 \def\prevolume{\_mtext{bib.volume}}
132 \def\prenumber{\_mtext{bib.number}}
133 \def\prepapes{\_mtext{bib.prepages}}
134 \def\posteditor{\_ifnum0\_namecountraw>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

```

141 \chardef\_documentlanguage=\_language
142 \def\_Mtext#1{\_csname\_mt:#1:\_csname\_lan:\_the\_documentlanguage\_endcsname\_endcsname}
143
144 \CreateField {lang}
145 \def\_setlang#1{\_ifx#1\_empty \_else
146     \_setbox0=\_vbox{\_langinput{#1}}%
147     \_ifcsname\_mt:bib.and:#1\_endcsname
148         \_language=\_csname\_#1Patt\_endcsname \_relax
149     \_else \opwarning{No phrases for "#1" used by [\EntryKey] in .bib}%
150     \_fi\_fi
151 }

```

Sorting.

bib-iso690.opm

```

157 \_fieldalias {key} {sortedby}

```

Supporting macros.

bib-iso690.opm

```

163 \def\bibwarninga{\_bibwarning}
164 \def\bibwarningb{\_bibwarning}
165
166 \def\_docitedate #1/#2/#3/#4\_relax{[\_Mtext{bib.citedate}]%
167     \_if^#2^#1\_else
168         \_if^#3^#1/#2\_else
169             \_cs{\_cs{\_lan:\_the\_documentlanguage}dateformat}#1/#2/#3\_relax
170     \_fi\_fi ]%
171 }
172 \def\_doyear#1{
173     \_biboptionvalue{yearprint}\_yearprint
174     \_ifx\_yearprint\_empty#1\_else\_def\YEAR{#1}\_yearprint\_fi
175 }
176 \def\_preparenumbering{%
177     \_def\VOL{\_RetrieveField{volume}}%
178     \_def\NO{\_RetrieveField{number}}%
179     \_def\PP{\_RetrieveField{pages}}%
180 }
181 \def\_prepareednote{%
182     \_def\EDN{\_RetrieveField{edition}}%
183     \_def\ADDR{\_RetrieveField{address}}%
184     \_def\PUBL{\_RetrieveField{publisher}}%
185     \_def\YEAR{\_RetrieveField{year}}%
186     \_def\AU{\_bprintb[!author]{\_doauthor0{####1}}{}}%
187     \_def\ED{\_bprintb[!editor]{\_doeditor0{####1}}{}}%
188     \_preparenumbering
189 }
190 \def\_doedition#1{%
191     \_biboptionvalue{editionprint}\_editionprint
192     \_ifx\_editionprint\_empty#1\_postedition\_else\_def\ED{#1}\_editionprint\_fi
193 }
194 \def\_doauthor#1#2{\_prepareauoptions{au}\_let\_iseditorlist=\_undefined
195     \_if1#1\_def\AU{#2}\_else\_let\_authorprint=\_empty\_fi

```

```

196 \_ifx\_authorprint\_empty #2\_else\_authorprint\_fi
197 }
198 \_def\_doeditor#1#2{\_prepareauedoptions{ed}\_let\_firstauthorformat=\_otherauthorformat
199 \_if1#1\_defED{#2}\_else\_let\_authorprint=\_empty\_fi
200 \_ifx\_authorprint\_empty #2\_posteditor\_else\_authorprint\_fi
201 }

```

Entry types.

bib-iso690.opm

```

207 \_sdef\_print:BEGIN){%
208 \_readbiboptions
209 \_biboptionvalue{titlepost}\_titlepost
210 \_isbiboption{unpublished}\_iftrue\_let\_bibwarninga=\_relax\_let\_bibwarningb=\_relax\_fi
211 \_isbiboption{nowarn}\_iftrue\_let\_bibwarning=\_relax\_fi
212 \_isbiboption{urlalso}\_iftrue\_def\_preurl{\_Mtext{bib.availablealso}}\_fi
213 \_RetrieveFieldIn{lang}\_langentry\_setlang\_langentry
214 }
215 \_sdef\_print:END){%
216 \_bprinta [note] {*\_}{}%
217 \_setbibmark
218 }
219 \_def\_bookgeneric#1{%
220 \_bprinta [howpublished] {[*].\ }{ }%
221 \_bprintb [edition] {\_doedition{##1}\ }{ }%
222 \_bprinta [ednote] {*\_}{ }%
223 \_bprinta [address] {*\_bprintv[publisher]{:}{\_bprintv[year]{,}{.}\ }{\_bibwarninga}%
224 \_bprinta [publisher] {*\_bprintv[year]{,}{.}\ }{\_bibwarninga}%
225 \_bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}{.}\ }{ }%
226 {\_bibwarning}%
227 \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{:}{.}\ }{ }%
228 \_bprinta [citedate] {\_docitedate*///\_relax.\ }{ }%
229 #1%
230 \_bprinta [series] {*\_}{ }%
231 \_bprinta [isbn] {ISBN~*.\ }{\_bibwarningb}%
232 \_bprinta [issn] {ISSN~*.\ }{ }%
233 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{ }%
234 \_bprintb [url] {\_preurl\_url{##1}. }{ }%
235 }
236 \_sdef\_print:book){%
237 \_bprintb [!author] {\_doauthor1{##1}\ }{\_bibwarning}%
238 \_bprintb [title] {\_em##1\_bprintc\_titlepost{:} \_bprintv[howpublished]{:}{.}\ }%
239 {\_bibwarning}%
240 \_bookgeneric}{%
241 }
242 \_sdef\_print:article){%
243 \_biboptionvalue{journalpost}\_journalpost
244 \_bprintb [!author] {\_doauthor1{##1}\ }{\_bibwarning}%
245 \_bprinta [title] {*\_ \_bprintc\_titlepost{*\_} }{\_bibwarning}%
246 \_bprintb [journal] {\_em##1\_bprintc\_journalpost{:} \_bprintv[howpublished]{:}{.}\ }%
247 {\_bibwarning}%
248 \_bprinta [howpublished] {[*].\ }{ }%
249 \_bprinta [address] {*\_bprintb[publisher]{:}{,}\ }{ }%
250 \_bprinta [publisher] {*\_}{ }%
251 \_bprinta [month] {*\_}{ }%
252 \_bprintb [year] {\_doyear{##1}\_bprintv[volume,number,pages]{,}{:}\ }{ }%
253 \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{:}{.}\ }
254 {\_bprinta [volume] {\_prevolume*\_bprintv[number,pages]{,}{:}\ }{ }%
255 \_bprinta [number] {\_prenumber*\_bprintv[pages]{,}{:}\ }{ }%
256 \_bprintb [pages] {\_prepapes\_hbox{##1}\_bprintv[citedate]{:}{.}\ }%
257 {\_bibwarning}%
258 \_bprinta [citedate] {\_docitedate*///\_relax.\ }{ }%
259 \_bprinta [issn] {ISSN~*.\ }{ }%
260 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{ }%
261 \_bprintb [url] {\_preurl\_url{##1}. }{ }%
262 }
263 \_sdef\_print:inbook){%
264 \_let\_bibwarningb=\_relax
265 \_bprintb [!author] {\_doauthor1{##1}\ }{\_bibwarning}%
266 \_bprinta [title] {*\_}{\_bibwarning}%

```

```

267 \_Incluse
268 \_bprintb [!editor] {\_doeditor1{##1}\:\ }{}%
269 \_bprintb [booktitle] {\_em##1\_bprintc\_titlepost{\:\ }*\_bprintv[howpublished]{\:\}\ }%
270 {\_bibwarning}%
271 \_bookgeneric{\_bprintb [pages] {\_prepages\_hbox{##1}. }{}%
272 }
273 \_slet{\_print:inproceedings}{\_print:inbook}
274 \_slet{\_print:conference}{\_print:inbook}
275
276 \_sdef{\_print:thesis}{%
277 \_bprintb [!author] {\_doauthor1{##1}\:\ }{\_bibwarning}%
278 \_bprintb [title] {\_em##1\_bprintc\_titlepost{\:\ }*\_bprintv[howpublished]{\:\}\ }%
279 {\_bibwarning}%
280 \_bprinta [howpublished] {[*].\ }{}%
281 \_bprinta [address] {*\_bprintv[school]{,}{\_bprintv[year]{,}{.}\ }{\_bibwarning}%
282 \_bprinta [school] {*\_bprintv[year]{,}{.}\ }{\_bibwarning}%
283 \_bprinta [month] {*, }{}%
284 \_bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\:\}\ }{\_bibwarning}%
285 \_bprinta [citedate] {\_docitedate*///\_relax.\ }{}%
286 \_bprinta [type] {*\_bprintv[ednote]{,}{.}\ }%
287 {\_ifx\_thesistype\_undefined\_bibwarning
288 \_else\_thesistype\_bprintv[ednote]{,}{.}\ }{\_fi}%
289 \_bprinta [ednote] {*.\ }{}%
290 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
291 \_bprintb [url] {\_preurl\_url{##1}. }{}%
292 }
293 \_sdef{\_print:phdthesis}{\_def\_thesistype{\_Mtext{bib.phdthesis}}\_cs{\_print:thesis}}
294 \_sdef{\_print:mastersthesis}{\_def\_thesistype{\_Mtext{bib.mastthesis}}\_cs{\_print:thesis}}
295 \_sdef{\_print:bachelorsthesis}{\_def\_thesistype{\_Mtext{bib.bachthesis}}\_cs{\_print:thesis}}
296
297 \_sdef{\_print:online}{%
298 \_bprintb [!author] {\_doauthor1{##1}\:\ }{}%
299 \_bprintb [title] {\_em##1\_bprintc\_titlepost{\:\ }*\_bprintv[howpublished]{\:\}\ }%
300 {\_bibwarning}%
301 \_bprinta [howpublished] {[*].\ }{}%
302 \_bprinta [ednote] {\_prepareednote*\_bprintv[citedate]{\:\}\ }{}%
303 \_bprinta [year] {}{}%
304 \_bprinta [accessed] {\_docitedate*///\_relax.\ }{\_bibwarning}%
305 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
306 \_bprintb [url] {\_preurl\_url{##1}. }{\_bibwarning}%
307 }
308
309 \_sdef{\_print:generic}{%
310 \_bprintb [!author] {\_doauthor1{##1}\:\ }{\_bibwarning}%
311 \_bprintb [title] {\_em##1\_bprintc\_titlepost{\:\ }*\_bprintv[howpublished]{\:\}\ }%
312 {\_bibwarning}%
313 \_bprinta [howpublished] {[*].\ }{}%
314 \_bprinta [ednote] {\_prepareednote*\_bprintv[citedate]{\:\}\ }{\_bibwarning}%
315 \_bprinta [citedate] {\_docitedate*///\_relax.\ }{}%
316 \_bprintb [doi] {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
317 \_bprintb [url] {\_preurl\_url{##1}. }{}%
318 }
319 \_slet{\_print:booklet}{\_print:generic}
320 \_slet{\_print:incollection}{\_print:generic}
321 \_slet{\_print>manual}{\_print:generic}
322 \_slet{\_print:proceedings}{\_print:generic}
323 \_slet{\_print:techreport}{\_print:generic}
324 \_slet{\_print:unpublished}{\_print:generic}
325
326 \_sdef{\_print:misc}{\_let\_bibwarning=\_relax \_cs{\_print:generic}}

```

## 2.33 Sorting and making Index

makeindex.opm

```
3 \_codedec1 \_makeindex {Makeindex and sorting <2024-12-01>} % preloaded in format
```

`\_makeindex` implements sorting algorithm at  $\text{\TeX}$  macro-language level. You need not any external program. The sorting can be used for various other applications, see an example in [Op \$\text{\TeX}\$  trick 0068](#).

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 `\_sortingdatalatin` implements these rules for the languages with latin alphabets. The groups between commas are not distinguished in the first pass. The second pass distinguishes all characters mentioned in the `\_sortingdatalatin` (commas are ignored). The order of letters in the `\_sortingdatalatin` macro is significant for the sorting algorithm.

makeindex.opm

```

27 \_def \_sortingdatalatin {%
28 |,<./,{ },-,&,@,%
29 aAâÄãÀÃäÁÁ,%
30 aÄ,%
31 bB,%
32 cC,%
33 ċĈċĈ,%
34 dDdĎ,%
35 eEëÊëËëÊëËë,%
36 eË,%
37 fF,%
38 gG,%
39 hH,%
40 ^T^U^V,% ch Ch CH
41 iIîïİİîİ,%
42 jJ,%
43 kK,%
44 lLíÍlÍ,%
45 Ł,%
46 mM,%
47 nNñÑ,%
48 ñÑñÑ,%
49 oOöÖóÓôÔ,%
50 pP,%
51 qQ,%
52 rRřŘ,%
53 řŘ,%
54 sSß,%
55 šŠšš,%
56 tTťĤ,%
57 uUûÛüÜúÚûÛü,%
58 vV,%
59 wW,%
60 xX,%
61 yYýŸÿŸ,%
62 zZ,%
63 žŽ,%
64 žŽ,%
65 žŽ,%
66 ^Z,% Hungarian: cz:c^Z, etc., see \_compoundcharshu in lang-data.opm
67 0,1,2,3,4,5,6,7,8,9,',>%
68 }
```

Characters to be ignored during sorting are declared in `\_ignoredcharsgeneric`. These characters are ignored in the first pass without additional condition. All characters are taken into account in the second pass: ASCII characters with code < 65 are sorted first if they are not mentioned in the `\_sortingdata...` macro. Others not mentioned characters have undefined behavior during sorting.

makeindex.opm

```

79 \_def \_ignoredcharsgeneric {.,;?!:"'() []=+-}
```

Sorting is always processed by rules of a given language. The macros `\_sortingdata<lang-tag>`, `\_ignoredchars<lang-tag>` and `\_compoundchars<lang-tag>` declare these rules. The `<lang-tag>` is ISO code of the language: en, cs, de, pl, es for example. The English language is implemented here. Other languages are implemented in the `lang-data.opm` file (see section 2.37.4).

makeindex.opm

```

90 \_let \_sortingdataen = \_sortingdatalatin % English alphabet is subset of Latin
91 \_let \_ignoredcharsen = \_ignoredcharsgeneric
92 \_def \_compoundcharsen {} % English doesn't have compound characters like DZ
```

The `\_compoundchars⟨lang-tag⟩` can declare changes performed before sorting. For example Czech language declares:

```
\_let \_sortingdatacs = \_sortingdatalatin % Czech alphabet is subset of Latin
\_def \_compoundcharscs {ch:^^T Ch:^^U CH:^^V}
```

It transforms two-letters `ch` to single character `^^T` because `ch` is treated as single compound character by Czech rules and `CH` is sorted between `H` and `I`. See `\_sortingdatalatin` where `^^T` is used. This declaration makes more transformations of `Ch` and `CH` too. The declarations of the form `x:y` in the `\_compoundchars⟨lang-tag⟩` are separated by space.

You can declare a transformation from single letter to more letters too. For example German rules sets `ß` equal to `ss` during sorting:

```
\_let \_sortingdatade = \_sortingdatalatin % German alphabet is subset of Latin
\_def \_compoundcharsde {ß:ss}
```

If there are two words equal after first pass of sorting: `Masse` (mass) and `Maße` (measures) for example, then second pass must decide about the order. DIN 5007, section 6.1 says: `ss` must be before `ß` in this case. So, we want to switch off the `\_compoundchars` declaration for the second pass and use the order of `s` and `ß` given in `\_sortingdata`. This is possible if the `\_xcompoundchars⟨lang-tag⟩` is defined. It has precedence in the second pass of sorting. We declare for German:

```
\_def \_xcompoundcharsde {}
```

German rules mention alternative sorting for phone-books or similar lists of names. The letters `ä` `ö` `ü` should be interpreted as `ae`, `oe` and `ue`. So we get `Mueller` < `Müller` < `Muff`. If this rule is not taken into account, we get `Mueller` < `Muff` < `Müller`. The rule can be implemented by:

```
\_def \_compoundcharsde {ß:ss Ä:AE Ö:OE Ü:UE ä:ae ö:oe ü:ue}
```

Because `u` < `ü` in `\_sortingdata` and because `\_xcompoundcharsde` is empty, we have `Mueller` < `Müller` after second pass of the sorting.

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`, `^^J` or `^^M` because these characters have very specific category codes.

If you created `\_sortingdata` etc. for your language, please, send them to me. I am ready to add them to the file `lang-data.opm` in a new `OpTeX` release. See also section 2.37.4.

French sorting rule says: if the words are the same except for accents then accented letters are sorted after unaccented letters but read the words from their end in the second pass. For example correct sorting is: `cote` < `côte` < `coté` < `côté`. This rule can be activated if the control sequence `\_secondpass⟨lang-tag⟩` is set to `\_reversewords`. For example, `lang-data.opm` declares `\_let\_secondpassfr=\_reversewords`.

Preparing to primary pass is performed by the `\_setprimarysorting` macro implemented here. The `⟨lang-tag⟩` is saved to the `\_sortinglang` macro when sorting is initialized in `\_dosorting` (it is typically derived from current `\language` value). The `\_setprimarysorting` is called from `\_dosorting` macro and all processing of sorting is in a group. It sets actual `\_sortingdata`, `\_compoundchars` and `\_ignoredchars` if given language declares them. If not then warning will be printed using `\_nold` macro and English data are used. The `\lccode` of all characters from `\_sortingdata` and `\_ignoredchars` are set. The sorted words will be converted using `\_compoundchars` followed by `\lowercase` before first pass is run.

makeindex.opm

```
164 \_def\_setprimarysorting {%
165   \_ea\_let \_ea\_sortingdata \_csname\_sortingdata\_sortinglang\_endcsname
166   \_ea\_let \_ea\_compoundchars \_csname\_compoundchars\_sortinglang\_endcsname
167   \_ea\_let \_ea\_ignoredchars \_csname\_ignoredchars\_sortinglang\_endcsname
168   \_def\_nold{}%
169   \_ifx \_sortingdata\_relax \_addto\_nold{ sortingdata}%
170   \_let \_sortingdata = \_sortingdataen \_fi
171   \_ifx \_compoundchars\_relax \_addto\_nold{ compoundchars}%
172   \_let \_compoundchars = \_compoundcharsen \_fi
173   \_ifx \_ignoredchars\_relax \_addto\_nold{ ignoredchars}%
174   \_let \_ignoredchars = \_ignoredcharsen \_fi
175   \_ifx\_nold\_empty\_else \_opwarning{Missing\_nold\_space for language (\_sortinglang)}\_fi
176   \_ifx \_compoundchars\_empty\_else
```

```

177 \_edef \_compoundchars {\_detokenize\_ea{\_compoundchars} }\_fi % all must be catcode 12
178 \_def \_act ##1{\_ifx##1\_relax \_else
179 \_ifx##1,\_advance\_tmpnum by1
180 \_else \_lccode`##1=\_tmpnum \_fi
181 \_ea\_act \_fi}%
182 \_tmpnum=65 \_ea\_act \_sortingdata \_relax
183 \_def \_act ##1{\_ifx##1\_relax \_else
184 \_lccode`##1=``^I
185 \_ea\_act \_fi}%
186 \_ea\_act \_ignoredchars \_relax
187 }

```

Preparing to secondary pass is implemented by the `\_setsecondarysorting` macro.

```

193 \_def\_setsecondarysorting {%
194 \_def \_act ##1{\_ifx##1\_relax \_else
195 \_ifx##1,\_else \_advance\_tmpnum by1 \_lccode`##1=\_tmpnum \_fi
196 \_ea\_act \_fi}%
197 \_tmpnum=64 \_ea\_act \_sortingdata \_relax
198 }

```

makeindex.opm

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 part of the string after `^^^` is ignored (you can have the same sorting key for different things) and the compound characters are converted by the `\_docompound` macro.

```

211 \_def \_preparesorting #1{%
212 \_edef \_tmpb {\_ea\_ignoreit\_csstring #1}% \,<string> -> <string>
213 \_edef\_tmpb{\_ea \_stripfromcaret \_tmpb ^^^\_fin}% <string>^^^<ignore> -> <string>
214 \_ea \_docompound \_compoundchars \_relax:{} % replace compound characters
215 \_lowercase \_ea{\_ea\_def \_ea\_tmpb \_ea{\_tmpb}}% convert in respect to \_sortingdata
216 \_ea\_replstring \_ea\_tmpb \_ea{\_csstring\^^I}{}% remove ignored characters
217 }
218 \_def \_docompound #1:#2 {%
219 \_ifx\_relax#1\_else \_replstring\_tmpb {#1}{#2}\_ea\_docompound \_fi
220 }
221 \_def\_stripfromcaret #1^^^#2\_fin{#1}

```

makeindex.opm

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.

```

238 \_newifi \_ifAleB
239
240 \_def\_isAleB #1#2{%
241 \_edef\_tmpb {#1&\_relax#2&\_relax}%
242 \_ea \_testAleB \_tmpb #1#2%
243 }
244 \_def\_testAleB #1#2\_relax #3#4\_relax #5#6{%
245 \_if #1#3\_if #1&\_testAleBsecondary #5#6% goto to the second pass::
246 \_else \_testAleB #2\_relax #4\_relax #5#6%
247 \_fi
248 \_else \_ifnum `#1<`#3 \_AleBtrue \_else \_AleBfalse \_fi
249 \_fi
250 }

```

makeindex.opm

The `\_testAleBsecondary` `\,<string1>` `\,<string2>` is run if the words are equal in the primary pass. It runs `\_setsecondarysorting` if it was not initialized already. Then prepares compared words to `\_tmpa` and `\_tmpb` and corrects them by `\_prepsecondpass` if needed. Finally, the test is recursively done by the macro `\_testAleBsecondaryX` `<converted-string1>``0\_relax``<converted-string2>``1\_relax`



```

261
262 \def\testAleBsecondary#1#2{%
263   \setsecondarysorting \let\setsecondarysorting=\relax
264   \preparesorting#1\let\tmpa=\tmpb \preparesorting#2%
265   \prepsecondpass
266   \edef\tmpb{\tmpa0\relax\tmpb1\relax}%
267   \ea\testAleBsecondaryX \tmpb
268 }
269 \def\testAleBsecondaryX #1#2\relax #3#4\relax {%
270   \if #1#3\testAleBsecondaryX #2\relax #4\relax
271   \else \ifnum `#1<`#3 \AleBtrue \else \AleBfalse \fi
272   \fi
273 }

```

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.

```

283 \def\mergesort #1#2,#3{% by Miroslav Olsak
284   \ifx,#1% % prazdna-skupina,neco, (#2=neco #3=pokracovani)
285     \toksapp\tmptoks{#2,}% % dvojice skupin vyresena
286     \sortreturn{\fif\mergesort#3}% % \mergesort pokracovani
287   \fi
288   \ifx,#3% % neco,prazna-skupina, (#1#2=neco #3=,)
289     \toksapp\tmptoks{#1#2,}% % dvojice skupin vyresena
290     \sortreturn{\fif\mergesort}% % \mergesort dalsi
291   \fi
292   \ifx\fin#3% % neco,konec (#1#2=neco)
293     \if;\_the\tmptoks;% % neco=kompletni setrideny seznam
294       \tmptoks{#1#2}%
295       \sortreturn{\fif\fif\gobbletoend}% % koncim
296     \else % neco=posledni skupina nebo \end
297       \sortreturn{\fif\fif % spojim \indexbuffer+necoa cele znova
298         \tmptoks\ea{\ea}\ea\mergesort\_the\tmptoks#1#2,#3}%
299       \fi\fi % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
300       \isAleB #1#3\ifAleB % p1<p2
301         \toksapp\tmptoks{#1}% % p1 do bufferu
302         \sortreturn{\fif\mergesort#2,#3}% % \mergesort neco1,p2+neco2,
303       \else % p1>p2
304         \toksapp\tmptoks{#3}% % p2 do bufferu
305         \sortreturn{\fif\mergesort#1#2,%} % \mergesort p1+neco1,neco2,
306       \fi
307       \relax % zarazka, na ktere se zastavi \sortreturn
308   }
309 \def\sortreturn#1#2\fi\relax{#1} \def\fif{\fi}
310 \def\gobbletoend #1\fin{}

```

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 current language (chosen for hyphenation patterns) is used for sorting data. If the macro `\_sortinglang` is defined as `\<lang-tag>` (for example `\def\_sortinglang{de}` for German) then this has precedence and current language is not used. Moreover, if you specify `\_asciisortingtrue` then ASCII sorting will be processed and all language sorting data will be ignored.

```

329 \newif \ifasciisorting \asciisortingfalse
330 \def\dosorting #1{%
331   \begingroup
332     \ifasciisorting \def\_sortinglang{ASCII}\fi
333     \ifx\_sortinglang\undefined \edef\_sortinglang{\_cs{\_lan:\_the\_language}}\fi
334     \sortmessage{OpTeX: Sorting \_string#1 (\_sortinglang) ...^^J}%
335     \ismacro\_sortinglang{ASCII}\iftrue
336       \def \_preparesorting#1{\edef\tmpb{\ea\_ignoreit\_csstring#1}}%
337       \let \_setsecondarysorting=\relax

```

```

338     \else
339         \setprimarysorting
340     \fi
341     \def \act##1{\_preparesorting ##1\_edef##1{\_tmpb}}%
342     \ea\_xargs \ea\_act #1;% \preparesorting for first pass of sorting applied
343     \ifcsname _xcompoundchars\_sortinglang\_endcsname
344         \ea\_let \ea\_compoundchars \csname _xcompoundchars\_sortinglang\_endcsname
345     \fi % \compoundchars can differ in the second pass of sorting
346     \csname _secondpass\_sortinglang\_endcsname % activates \reversewords if needed
347     \def \act##1{\_toksapp\_tmptoks{##1,}}%
348     \tmptoks{}%
349     \edef #1{\_ea}\_ea\_xargs \ea\_act #1;% commas between items added, mergesort initialized
350     \tmptoks\_ea{\_ea}\_ea\_mergesort \_the\_tmptoks\_fin,\_fin
351     \ea\_endgroup
352     \ea\_def\_ea#1\_eaf\_the\_tmptoks}%
353 }
354 \let\_sortmessage=\_message % User can do \let\_sortmessage=\_ignoreit, for example

```

French rules needs reverse reading the words in the second pass. The `\_reversewords` is activated in this case and it adds new job to the macro `\_prepsecondpass`: it reverses the letters in the compared words (saved in `\_tmpa` and `\_tmpb`) by the expandable `\_sortrevers` macro. The `\_prepsecondpass` macro is used in the `\_testAleBsecondary` and it is empty by default.

makeindex.opm

```

365 \def\_prepsecondpass{}
366 \def\_reversewords{%
367     \addto\_prepsecondpass{\_edef\_tmpa{\_ea\_sortrevers\_tmpa\_relax}%
368         \_edef\_tmpb{\_ea\_sortrevers\_tmpb\_relax}}%
369 }
370 \def\_sortrevers #1#2\_relax{\_ifx^#2^#1\_else \_sortrevers#2\_relax #1\_fi}

```

The `\makeindex` prints the index. First, it sorts the `\_iilist` second, it prints the sorted `\_iilist`, each item is printed using `\_printindexitem`.

We set `\leftskip=\_iindent` and we suppose that each index entry starts by `\noindent\hskip-\_iindent` (see the macro `\_printii`). Then the next lines of the same index entry (if the page list is broken to more pages) is indented by `\leftskip=\_iindent`.

makeindex.opm

```

383 \def\_makeindex{\_par
384     \ifx\_iilist\_empty \_opwarning{index data-buffer is empty. TeX me again}%
385     \incr\_unresolvedrefs
386     \else
387         \dosorting \_iilist % sorting \_iilist
388         \bgroup
389         \rightskip=0pt plus1fil \exhyphenpenalty=10000 \leftskip=\_iindent
390         \ea\_xargs \ea\_printindexitem \_iilist ;\_par
391         \egroup
392     \fi
393 }
394 \_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

```

404 \def\_printindexitem #1{%
405     \ifcsname _\_csstring #1\_endcsname
406         \ea\_ea\_ea \_printii \csname _\_csstring #1\_endcsname &%
407     \else
408         \ea\_ea\_ea\_printii \ea\_ignoreit \csstring #1&%
409     \fi
410     \ea\_printiipages #1&
411 }

```

`\_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>}{\<word>}` macro is empty by default. It is invoked if first letter of index entry 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 #1}\nobreak\medskip}
```

`\_definefirstii`  $\langle word \rangle$  & macro defines `\_firstii` which is used as the  $\langle letter \rangle$  parameter of the macro `\_newiiletter` and for testing if the “first letter” of the index entry was changed. The `\uppercase` of the real first letter is used by default here. You can re-implement `\_definefirstii` if you want. For example, you want to ignore accents above letters for index sub-headers:

```
\def\_definefirstii#1#2{%
  \uppercase{\bgroup \iicodes \uppercase{\egroup\def\_firstii{#1}}}}
\def\iicodes{}
\def\setiicodes #1#2,{\_ifx^#1\_else
  \foreach #2\do{\_addto\iicodes{\uccode`##1=`#1}}
  \_ea\setiicodes \_fi
}
\setiicodes AÀÁÂÃÄÅ,ĆČ,ĐĎ,ÈÉÊËĚ,ÍĪĬ,ĹĻ,ŌŎŰ,ŔŖ,ŠŠ,ŤŦ,ŮŰÚŮŰŮ,ÝŸ,{},
```

If the first character of the  $\langle word \rangle$  is `<` or `>` then it isn’t printed. It can be used as first character of the index entry in order to put the entry to a group entries sorted `<`before or `>`after whole normal entries.

makeindex.opm

```
447 \_def\_printii #1&{\_definefirstii #1&%
448 \_ifx\_firstii\_lastii\_else
449 \_ea\_newiiletter\_ea{\_firstii}{#1}\_let\_lastii=\_firstii\_fi
450 \_gdef\_currii{#1}\_the\_everyii\_noindent
451 \_hskip-\_iindent \_ignorespaces \_isnextchar<{\_ea\_printiiA\_ignoreit}%
452 {\_isnextchar>{\_ea\_printiiA\_ignoreit}{\_printiiA}{#1//}
453 \_def\_printiiA #1/{\_if^#1\_let\_previi=\_currii \_else
454 \_ea\_scanprevii\_previi/&\_edef\_tmpb{\_detokenize{#1}}%
455 \_ifx\_tmpa\_tmpb \_iiemdash \_else#1 \_gdef\_previi{\_fi
456 \_ea\_printiiA\_fi
457 }
458 \_def\_definefirstii #1#2&{\_uppercase{\_def\_firstii{#1}}}
459 \_def\_iiemdash{\_kern.1em---\_space}
460 \_def\_lastii{}
461 \_def\_newiiletter#1#2{}
462
463 \_def\_scanprevii#1/#2&{\_def\_previi{#2}\_edef\_tmpa{\_detokenize{#1}}}
464 \_def\_previi{} % previous index item
```

`\_printiipages`  $\langle pglst \rangle$  & gets  $\langle pglst \rangle$  in the form  $\langle pg \rangle : \langle type \rangle, \langle pg \rangle : \langle type \rangle, \dots \langle pg \rangle : \langle type \rangle$  and it converts them to  $\langle pg \rangle$ ,  $\langle pg \rangle$ ,  $\langle from \rangle -- \langle to \rangle$ ,  $\langle pg \rangle$  etc. The same pages must be printed only once and continuous consequences of pages must be compressed to the form  $\langle from \rangle - \langle to \rangle$ . Moreover, the consequence is continuous only if all pages have the same  $\langle type \rangle$ . Empty  $\langle type \rangle$  is most common, pages with `b`  $\langle type \rangle$  must be printed as bold and with `i`  $\langle type \rangle$  as italics. Moreover, the  $\langle pg \rangle$  mentioned here are  $\langle gpageno \rangle$ , but we have to print  $\langle pageno \rangle$ . The following macros solve these tasks.

makeindex.opm

```
478 \_def\_printiipages#1&{\_let\_pgtype=\_undefined \_tmpnum=0 \_printpages #1,.,\_par}
479 \_def\_printpages#1:#2,{% state automaton for compriming pages
480 \_ifx,#1,\_uselastpgnum
481 \_else \_def\_tmpa{#2}%
482 \_ifx\_pgtype\_tmpa \_else
483 \_let\_pgtype=\_tmpa
484 \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
485 \_tmpnum=#1 \_returnfi \_fi
486 \_ifnum\_tmpnum=#1 \_returnfi \_fi
487 \_advance\_tmpnum by1
488 \_ifnum\_tmpnum=#1 \_ifx\_lastpgnum\_undefined \_usepgdash\_fi
489 \_edef\_lastpgnum{\_the\_tmpnum:{\_pgtype}}%
490 \_returnfi \_fi
491 \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
492 \_tmpnum=#1
493 \_relax
494 \_ea\_printpages \_fi
495 }
496 \_def\_returnfi #1\_relax{\_fi}
497 \_def\_uselastpgnum{\_ifx\_lastpgnum\_undefined
498 \_else \_ea\_pgprint\_lastpgnum \_let\_lastpgnum=\_undefined \_fi
```

```

499 }
500 \def\usepgcomma{\ifnum\tmpnum>0, \fi} % comma+space between page numbers
501 \def\usepgdash{\hbox{--}} % dash in the <from>--<to> form

```

You can re-define `\pgprint`  $\langle gpageno \rangle : \{ \langle iitype \rangle \}$  if you need to implement more  $\langle iitypes \rangle$ .

```

508 \def\pgprint #1:#2{%
509   \ifx ,#2,\pgprintA{#1}\returnfi \fi
510   \ifx b#2{\bf \pgprintA{#1}}\returnfi \fi
511   \ifx i#2{\it \pgprintA{#1}}\returnfi \fi
512   \ifx u#2\pgu{\pgprintA{#1}}\returnfi \fi
513   \pgprintA{#1}\relax
514 }
515 \def\pgprintA #1{\_ilink[pg:#1]{\_cs{pgi:#1}} % \_ilink[pg:<gpageno>]{<pageno>}
516 \def\pgu#1{\_leavevmode\_vtop{\_hbox{#1}\kern.3ex\_hrule}}

```

makeindex.opm

The `\iindex`  $\langle word \rangle$  puts one  $\langle word \rangle$  to the index. It writes `\_Xindex`  $\langle word \rangle \{ \langle iitype \rangle \}$  to the .ref file. All other variants of indexing macros expand internally to `\iindex`.

```

524 \def\iindex#1{\_isempty{#1}\iffalse
525   \openref{\def~{ }\_ewref\_Xindex{#1}{\_iitypesaved}}\_fi}
526 \public \iindex ;

```

makeindex.opm

The `\_Xindex`  $\langle word \rangle \{ \langle iitype \rangle \}$  stores  $\backslash, \langle word \rangle$  to the `\_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 `\_pgi:  $\langle gpageno \rangle$`  is defined as  $\langle pageno \rangle$ .

```

538 \def \_iilist {}
539 \def \_Xindex #1#2{\_ea\_XindexA \_csname ,#1\_ea\_endcsname \_currpage {#2}}
540 \def \_XindexA #1#2#3#4{% #1=\,<word> #2=<gpageno> #3=<pageno> #4=<iitype>
541   \ifx#1\_relax \_global\_addto \_iilist {#1}%
542     \gdef #1{#2:#4}%
543   \else \_global\_addto #1{, #2:#4}%
544   \fi
545   \sxddef\_pgi:#2{#3}%
546 }

```

makeindex.opm

The implementation of macros `\ii`, `\iid`, `\iis` follows. Note that `\ii` works in the horizontal mode in order to the `\write` whatsit is not broken from the following word. If you need to keep vertical mode, use `\iindex`  $\langle word \rangle$  directly.

The `\iitype`  $\{ \langle type \rangle \}$  saves the  $\langle type \rangle$  to the `\_iitypesaved` macro. It is used in the `\iindex` macro.

```

558 \def\ii #1 {\_leavevmode\_def\_tmp{#1}\_iiA #1,,\_def\_iitypesaved{}}
559
560 \def\_iiA #1,{\_if$#1$\_else\_def\_tmpa{#1}%
561   \_ifx\_tmpa\_iiatsign \_ea\_iiB\_tmp,,\_else\_iindex{#1}\_fi
562   \_ea\_iiA\_fi}
563 \def\_iiatsign{0}
564
565 \def\_iiB #1,{\_if$#1$\_else \_iiC#1/\_relax \_ea\_iiB\_fi}
566 \def\_iiC #1/#2\_relax{\_if$#2$\_else\_iindex{#2#1}\_fi}
567
568 \def\_iid #1 {\_leavevmode\_iindex{#1}\_def\_iitypesaved{}}%
569   \_isnextchar<\_ignoreit>{\_isnextchar>{\_ignoreit}{}}#1\_futurelet\_tmp\_iid}
570 \def\_iidD{\_ifx\_tmp,\_else\_ifx\_tmp.\_else\_space\_fi\_fi}
571
572 \def\_iis #1 #2{{\_def~{ }\_global\_sdef{_,#1}{#2}}\_ignorespaces}
573
574 \def\_iitypesaved{}
575 \def\_iitype #1{\_def\_iitypesaved{#1}\_ignorespaces}
576
577 \public \ii \iid \iis \iitype ;

```

makeindex.opm

## 2.34 Footnotes and marginal notes

fnotes.opm

```
3 \_codedecl \fnote {Footnotes, marginal notes OpTeX <2026-03-19>} % preloaded 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.

fnotes.opm

```
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).

fnotes.opm

```
48 \_def \_printfnotemark {\_quitvmode\_hbox{$\fnotenum$}} % 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\_printfnotemarkA{\_link[fnt:\_the\_gfnotenum]{#1}{\_printfnotemark}}%
54     \_dest[fnf:\_the\_gfnotenum]}%
55   \_def\_printfnotemarkB{\_link[fnf:\_the\_gfnotenum]{#2}{\_printfnotemark}}%
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:<fnotenum>`. 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 \_protected\_def \fnote{\_fnotemark1\_fnotetext}
76 \_protected\_def \fnotemark#1{\_advance\_gfnotenum by#1\_advance\_lfnotenum by#1\_relax
77   \_printfnotemarkA}}
```

The `\fnotetext` calls `\_opfootnote` which is equivalent to plain T<sub>E</sub>X `\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

```

91 \def\fnotetext{\_incr\_gfnotenum\_incr\_lfnotenum % global increment
92   \ifpgfnote \openref \fi
93   \wref \Xfnote}%
94   \ifpgfnote \ifcsname _fn:\_the\_gfnotenum \endcsname \else
95     \opwarning{unknown \noexpand\fnote mark. TeX me again}%
96     \incr\_unresolvedrefs
97   \fi\fi
98   \opfootnote\fnset\printfnotemarkB
99 }
100 \def\fnset{\_everypar={}\_scalemain\_typoscale[800/800]}
101
102 \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⟩` is 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

```

114 \newcount\mnnotenum \mnnotenum=0 % global counter of mnotes
115 \def \Xmnote {\_incr\_mnnotenum\_ea \XmnoteA \currpage}
116 \def \XmnoteA #1#2{% #1=<gpageno> #2=<pageno>
117   \sxddef{\mn:\_the\_mnnotenum}{\_ifodd\_numtype{#2}{#1} \_right \_else \_left \_fi}}
118 \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

```

126 \def \_fixmnotes #1{\_edef\_mnotesfixed{\_cs{\_csstring #1}}
127 \_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 shifted down by `\parskip` and with `\vskip-\baselineskip` followed.

fnotes.opm

```

136 \def\mnote #1{\_ifx^#1\_else \mnoteC#1\_fin \_fi \mnoteD}
137 \def\mnoteC up#1\_fin{\_mnnoteskip=#1\_relax} % \mnote up<dimen> {<text>} syntax
138 \_long\_def\mnoteD#1{%
139   \_ifmode \_vskip\_parskip{\_mnoteA{#1}}\_nobreak\_vskip-\_baselineskip\_vskip-\_parskip \_else
140   \_lower\_dp\_strutbox\_hbox{\_vadjust{\_kern-\_dp\_strutbox \_mnoteA{#1}\_kern\_dp\_strutbox}}%
141   \_fi
142 }
143 \_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

```

153 \_newdimen\_mnoteskip
154 \_public \mnoteskip ;

```

The `\mnoteA` macro does the real work. The `\lrnote{<left>}{<right>}` uses only first or only second parameter depending on the left or right marginal note. The `\mninvert` macro can be redefined for special purposes, see [OpTeX trick 0175](#).

fnotes.opm

```

163 \_long\_def\mnoteA #1{\_incr\_mnnotenum
164   \_ifx\_mnotesfixed\_undefined
165     \_ifcsname _mn:\_the\_mnnotenum \endcsname
166     \_edef\_mnotesfixed{\_mninvert\_cs{\_mn:\_the\_mnnotenum}}%
167     \_else
168     \_opwarning{unknown \noexpand\mnote side. TeX me again}\_openref

```



```

169     \incr\unresolvedrefs
170     \def\mnotesfixed{\_right}%
171     \fi\_fi
172     \hbox to0pt{\_wref\_Xmnote{}\_everypar={}\_resetatrrs
173         \lrmnote{\_kern-\_mnotesize \_kern-\_mnoteindent}{\_kern\_hsize \_kern\_mnoteindent}%
174         \vbox to0pt{\_vss \_setbox0=\_vtop{\_hsize=\_mnotesize
175             \lrmnote{\_leftskip=0pt plus 1fill \_rightskip=0pt}
176                 {\_rightskip=0pt plus 1fil \_leftskip=0pt}%
177                 {\_the\_everymnote\_noindent#1\_endgraf}}}%
178         \_dp0=0pt \_box0 \_kern\_mnoteskip \_global\_mnoteskip=0pt}\_hss}%
179 }
180 \def \lrmnote#1#2{\_ea\_ifx\_mnotesfixed\_left #1\_else #2\_fi}
181 \def\_mninvert{} % should be redefined, see OpTeX trick 0175

```

We don't want to process `\fnote`, `\fnotemark`, `\mnote` in TOC, headlines nor outlines.

fnotes.opm

```

188 \regmacro {\_def\fnote#1{}} {\_def\fnote#1{}} {\_def\fnote#1{}}
189 \regmacro {\_def\fnotemark#1{}} {\_def\fnotemark#1{}} {\_def\fnotemark#1{}}
190 \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` style initialization macro is defined here.

styles.opm

```

32 \_def\_report{
33     \_typosize[11/13.2]
34     \_vsize=\_dimexpr \_topskip + 52\_baselineskip \_relax % added 2020-03-28
35     \_let\_titfont=\_chapfont
36     \_titskip=3ex
37     \_eoldef\_author##1{\_removelastskip\_bigskip
38         {\_leftskip=0pt plus1fill \_rightskip=\_leftskip \_it \_noindent ##1\_par}\_nobreak\_bigskip
39     }
40     \_public \author ;
41     \_parindent=1.2em \_iindent=\_parindent \_ttindent=\_parindent
42     \_footline={\_global\_footline={\_hss\_rmfixed\_folio\_hss}}
43 }

```

The `\letter` style initialization macro is defined here.

The `\letter` defines `\address` and `\subject` macros.

See the files `demo/op-letter-*.tex` for usage examples.

styles.opm

```

53 \def\letter{
54   \def\address{\vtop\bgroup\boxlines \parskip=0pt \let\par=\egroup}
55   \def\subject{{\bf \mtext{subj}: }}
56   \public \address \subject ;
57   \typosize[11/14]
58   \vsize=\dimexpr \topskip + 49\baselineskip \relax % added 2020-03-28
59   \parindent=0pt
60   \parskip=\medskipamount
61   \nopagenumbers
62 }
63 \public \letter \report ;

```

The `\slides` macro reads macro file `slides.opm`, see the section 2.35.2.

styles.opm

```

69 \def\slides{\par
70   \opinput{slides.opm}
71   \edef*\relax\ifmmode*\else\ea\startitem\fi}
72 }
73 \public \slides ;

```

## 2.35.2 \slides style for presentations

slides.opm

```

3 \codedecl \slideshow {Slides style for OpTeX <2025-02-27>} % 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.

slides.opm

```

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 \typosize[16/19]
19 \def\urlfont{}
20 \everytt={\typosize[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 \let\scolor=\Blue % secondary color used in default design
31
32 \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.

slides.opm

```

42 \footlinedist=1mm
43 \footline={\hss \rlap{%
44   \rlap{\Grey\kern.2\hoffset\vrule height6mm depth2mm width.8\hoffset}%
45   \hbox to\hoffset{\White\hss\folio\kern3mm}}}

```

The `\subtit` is defined analogically like `\tit`.

slides.opm

```

51 \eoldef\subtit#1{\vskip20pt {\leftskip=0pt plus1fill \rightskip=\leftskip
52   \subtitfont #1\nbpar}}

```

The `\pshow<num>` prints the text in invisible (transparent) font when `\layernum<num>`. For transparency we need to define special graphics states.

```

60 \def\Transparent {\_transparency255 }
61 \_public \Transparent ;
62
63 \def\_use#1#2{\_ifnum\_layernum#1\_relax#2\_fi}
64 \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+`.

```

76 \_newcount\_gilevel
77 \_def\*{*}
78 \_adef*{\_relax\_ifmode*\_else\_ea\_startitem\_fi} % defined also in styles.opm
79 \_sdef{\_item:X}{\_scolor\_raise.2ex\_fullrectangle{.8ex}\_kern.5em}
80 \_sdef{\_item:x}{\_scolor\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
81 \_style X
82 \_def\_egroups{\_par\_global\_gilevel=\_ilevel \_egroup}
83 \_everylist={\_novspaces \_ifcase\_ilevel \_or \_style x \_else \_style - \_fi
84 \_addto\_egroups{\_egroup}}

```

The default values of `\pg`, i.e. `\pg;`, `\pg+` and `\pg.` are very simple. They are used when `\slideshow` is not specified.

```

91 \_def\_pg#1{\_cs{\_spg:#1}}
92 \_sdef{\_spg:;}{\_vfil\_break \_lfnotenumreset}
93 \_sdef{\_spg:..}{\_endslides}
94 \_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..`

```

106 \_def\_endslides{\_vfill \_supereject \_byehook \_end}
107 \_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 `\_scolor`.

```

115 \_def\_titfont{\_typosize[42/60]\_boldify \_scolor}
116 \_def\_subtitfont{\_typosize[20/30]\_boldify}
117 \_def\_secfont{\_typosize[25/30]\_boldify \_scolor}
118
119 \_nonum \_notoc \_let\_resetnonumnotoc=\_relax
120 \_def\_printsec#1{\_par
121 \_abovetitle{\_penalty-400}\_bigskip
122 {\_secfont \_noindent \_leftskip=0pt plus1fill \_rightskip=\_leftskip
123 \_printrefnum[@\_quad]#1\_nbparg}\_insertmark{#1}%
124 \_nobreak \_belowtitle{\_medskip}%
125 }

```

When `\slideshow` is active then each page is opened by `\setbox\_slidepage=\vbox\bgroup` (roughly speaking) and closed by `\egroup`. The material is `\unvboxed` and saved for the usage in the next usage if `\pg+` is in process. The `\_slidelayer` is incremented instead `\pageno` if `\pg+`. This counter is equal to `\count1`, so it is printed to the terminal and log file next to `\pageno`.

The code is somewhat more complicated when `\layers` is used. Then `\_layered-text` is saved to the `\_layertext` macro, the material before it is in `\_slidepage` box and the material after it is in `\_slidepageB` box. The pages are completed in the `\loop` which increments the `\layernum` register and prints page by the `\_printlayers`

```

143 \_newbox\_slidepage \_newbox\_slidepageB
144 \_countdef\_slidelayer=1
145
146 \_def\_slideshow{\_slidelayer=1 \_slideshowactive
147 \_let\_slideopen=\_relax % first wins
148 \_setbox\_slidepage=\vbox\bgroup\bgroup}

```

```

149
150 \def\slideshowactive{%
151   \sdef{spg:;}{\closepage \global\slidelayer=1 \resetpage \openslide}
152   \sdef{spg:}{\closepage \endslides}
153   \sdef{spg:+}{\closepage \incr\slidelayer \decr\pageno \openslide}
154   \let\layers=\layersactive
155   \slidelinks % to prevent hyperlink-dests duplication
156 }
157 \def\openslide{\setbox\slidepage=\vbox\bgroup\bgroup \setilevel
158   \ifvoid\slidepage \else \unvbox\slidepage \nointerlineskip\lastbox \fi}
159 \def\setilevel{\loop \decr\gilevel \ifnum\gilevel<0 \else \begitem \repeat}
160
161 \def\closepage{\egroups \egroup
162   \ifnum \maxlayers=0 \unvcopy\slidepage \vfil\break
163   \else \begingroup \setwarnslides \layernum=0
164     \loop
165       \ifnum\layernum<\maxlayers \advance\layernum by1
166       \printlayers \vfil\break
167       \ifnum\layernum<\maxlayers \incr\slidelayer \decr\pageno \fi
168     \repeat
169     \global\maxlayers=0
170     \incr\layernum \global\setbox\slidepage=\vbox{\printlayers}%
171   \endgroup
172   \fi}
173 \def\resetpage{%
174   \global\setbox\slidepage=\box\voidbox \global\setbox\slidepageB=\box\voidbox
175   \lfnotenunreset
176 }
177 \def\setwarnslides{%
178   \def\pg##1{\opwarning{\string\pg##1 \layersenv}\def\pg####1{}}%
179   \def\layers##1 {\opwarning{\string\layers\space \layersenv}\def\layers####1{}}%
180 }
181 \def\layersenv{cannot be inside \string\layers...\string\endlayers, ignored}
182
183 \def\printlayers{\unvcopy\slidepage \prevdepth=\dp\slidepage
184   {\layertext \endgraf}%
185   \vskip\parskip
186   \unvcopy\slidepageB
187 }
188 \let\destboxori=\destbox
189
190 \newcount\layernum \newcount\maxlayers
191 \maxlayers=0
192
193 \long\def\layersactive #1 #2\endlayers{%
194   \par\penalty0\egroup\egroup
195   \gdef\layertext{\settinglayer#2}%
196   \global\maxlayers=#1
197   \setbox\slidepageB=\vbox\bgroup\bgroup
198   \setbox0=\vbox{\layernum=1 \globaldefs=-1 \layertext\endgraf}}\prevdepth=\dp0
199 }
200 \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

```

207 \def\slideopen{\let\slideshow=\relax % first wins
208   \sdef{spg:;}{\egroups\vfil\break \lfnotenunreset\bgroup \setilevel}
209   \sdef{spg:}{\egroups\endslides}
210   \sdef{spg:+}{\egroups\bgroup \setilevel}
211   \let\layersopen=\egroup \let\layersclose=\bgroup
212   \bgroup
213 }
214 \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

```
239 \_def\_slidelinks{%
240   \_def \_destbox[##1]{\_edef\_tmp{\_noexpand\_sdestbox[##1]}%
241     \_global\_ea\_addto\_ea\_destboxes\_ea{\_tmp}}%
242   \_def \_pagedest {%
243     \_hbox{\_def\_destheight{25pt}\_sdestbox[pg:\_the\_gpageno]\_destboxes}%
244     \_nointerlineskip \_gdef\_destboxes{}%
245   }%
246   \_ifx \_dest\_destactive \_else \_let\_pagedest=\_relax \_fi
247 }
248 \_let\_sdestbox = \_destbox
249 \_def\_destboxes{} % initial value of \_destboxes
250 \_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 `\_slideshook` (used by the `\label` macro). Moreover, the warning about illegal use of `\bib`, `\usebib` in `\layers` environment is activated.

slides.opm

```
260 \_def\_settinglayer{%
261   \_def\_slideshook ##1##2{%
262     \_def\_bibB[##1]{\_nousebib}\_def\_usebib/##1 (##2) ##3 {\_nousebib}%
263   }
264   \_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

```
272 \_long\_def\_layers #1 #2\endlayers{\_par
273   \_layersopen {\_layernum=\_numexpr#1+1\_relax #2\_endgraf}\_layersclose}
274 \_let\_layersopen=\_relax
275 \_let\_layersclose=\_relax
276
277 \_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

```
287 \_def \_fnotenumpages {\_def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse
288   \_def\_lfnotenumreset{\_global\_lfnotenum=0 }}
289 \_let \_lfnotenumreset=\_relax
290 \_public \_fnotenumpages ;
```

## 2.36 Logos

logos.opm

```
3 \_codedecl \TeX {Logos TeX, LuaTeX, etc. <2025-01-22>} % preloaded in format
```

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.

logos.opm

```
13 \_protected\_def \_TeX {T\_kern-.1667em\_lower.5ex\_hbox{E}\_kern-.125emX\_ignoreslash}
14 \_protected\_def \_OpTeX {Op\_kern-.1em\_TeX}
15 \_protected\_def \_LuaTeX {Lua\_TeX}
16 \_protected\_def \_LuaHBTeX {Lua{\_setfontsize{mag.9}\_resizethefont HB}\_TeX}
17 \_protected\_def \_XeTeX {X\_kern-.125em\_phantom E}
18   \_pdfsave\_rlap{\_pdfscale{-1}{1}\_lower.5ex\_hbox{E}}\_pdfrestore \_kern-.1667em \_TeX}
19
20 \_def\_ignoreslash {\_isnextchar/\_ignoreit{}}
21
22 \_public \_TeX \_OpTeX \_LuaTeX \_LuaHBTeX \_XeTeX \_ignoreslash ;
```

The `\ConTeXt` logo is implemented as in the ConTeXt format itself. The kerning between “Con” and “TeXt” is calculated by measuring the kerning between the letters “T” and “e”.

logos.opm

```
30 \protected\def \ConTeXt{\begingroup
31   Con\setbox0=\hbox{T\kern\zo e}\setbox1=\hbox{Te}{\kern\dimexpr\wd1 -\wd0}%
32   \TeX t\endgroup\ignoreslash}
33
34 \public \ConTeXt ;
```

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.

logos.opm

```
41 \protected\def \LaTeX{\tmpdim=.42ex L\kern-.36em \kern \slantcorr % slant correction
42   \raise \tmpdim \hbox{\theFontscale[710]A}%
43   \kern-.15em \kern-\slantcorr \TeX}
44 \def\slantcorr{\ea\ignorept \the\fontdimen1\font\tmpdim}
45
46 \public \LaTeX ;
```

`\OPmac`, `\CS` and `\csplain` logos.

logos.opm

```
52 \def\OPmac{\leavevmode
53   \lower.2ex\hbox{\theFontscale[1400]O}\kern-.86em P{\em mac}\ignoreslash}
54 \def\CS{$\cal CS\kern-.1667em\lower.5ex\hbox{$\cal S$}\ignoreslash}
55 \def\csplain{\CS plain\ignoreslash}
56
57 \public \OPmac \CS \csplain ;
```

The expandable versions of logos used in Outlines need the expandable `\ignslash` (instead of the `\ignoreslash`).

logos.opm

```
64 \def\ignslash#1{\ifx/#1\else #1\fi}
65 \regmacro {}{}{% conversion for PDF outlines
66   \def\TeX\TeX\ignslash}\def\OpTeX\OpTeX\ignslash}%
67   \def\LuaTeX\LuaTeX\ignslash}\def\XeTeX\XeTeX\ignslash}%
68   \def\LaTeX\LaTeX\ignslash}\def\OPmac\OPmac\ignslash}%
69   \def\ConTeXt\ConTeXt\ignslash}%
70   \def\CS\CS\def\csplain\csplain\ignslash}%
71 }
72 \public \ignslash ;
```

## 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 T<sub>E</sub>X 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`.

uni-lcuc.opm

```
3 \wlog{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 Multilingual phrases and quotation marks

languages.opm

```
3 \_codedecl \_mtext {Languages <2026-05-17>} % preloaded in format
```

Four words are generated by OpTeX macros: “Chapter”, “Table”, “Figure” and “Subject”. These phrases are generated depending on the current value of the `\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-tag⟩}` where `⟨phrase-id⟩` is a label for the declared word and `⟨lang-tag⟩` is a language shortcut declared by `\_preplang`.

languages.opm

```

16 \_def\_mtext#1{\_tryscs{\_mt:#1:\_tryscs{\_lan:\_the\_language}}{en}}
17 {\_csname\_mt:#1:en\_endcsname}}

```

We can declare such language-dependent words by

```

\_sdef{\_mt:chap:en}{Chapter} \_sdef{\_mt:chap:cs}{Kapitola}
\_sdef{\_mt:t:en}{Table} \_sdef{\_mt:t:cs}{Tabulka}

```

etc. but we use more “compact” macro `\_langw⟨lang-tag⟩⟨chapter⟩⟨table⟩⟨figure⟩⟨subject⟩` for declaring them.

languages.opm

```

30 \_def\_langw #1 #2 #3 #4 #5 {%
31 \_sdef{\_mt:chap:#1}{#2}\_sdef{\_mt:t:#1}{#3}\_sdef{\_mt:f:#1}{#4}%
32 \_sdef{\_mt:subj:#1}{#5}%
33 }

```

More phrases are auto-generated in bibliography references. They are declared by

`\_langb⟨lang-tag⟩{⟨and⟩}{⟨et-al⟩}{⟨ed⟩}{⟨cit⟩}{⟨vol⟩}{⟨no⟩}{⟨pp⟩}{⟨p⟩}{⟨ed⟩}{⟨eds⟩}{⟨avail-from⟩}{⟨avali-to⟩}{⟨ba-thesis⟩}{⟨ma-thesis⟩}{⟨phd-thesis⟩}`. It is used similar way as the `\_langw` above. Both these macros are used in `lang-data.opm` file, see the end of section 2.37.3.

languages.opm

```

43 \_def\_langb#1 #2#3#4#5#6#7#8#9{\_def\_mbib##1##2{\_sdef{\_mt:bib.##2:#1}{##1}}%
44 \_mbib{#2}{and}\_mbib{#3}{etal}\_mbib{#4}{edition}\_mbib{#5}{citedate}\_mbib{#6}{volume}%
45 \_mbib{#7}{number}\_mbib{#8}{prepages}\_mbib{#9}{postpages}\_langbA}
46 \_def\_langbA#1#2#3#4#5#6#7{\_mbib{#1}{editor}\_mbib{#2}{editors}\_mbib{#3}{available}%
47 \_mbib{#4}{availablealso}\_mbib{#5}{bachthesis}\_mbib{#6}{masthesis}\_mbib{#7}{phdthesis}}

```

`\today` macro needs auto-generated words for each name of the month.

`\_monthw⟨lang-tag⟩⟨January⟩⟨February⟩...⟨December⟩` is used for decaring them.

The language-dependent format for printing date should be declared like

```
\_sdef{\_mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
```

This example declares date format for English where  $\langle lang-tag \rangle$  is en.

languages.opm

```
60 \_def \_monthw #1 #2 #3 #4 #5 #6 #7 {%
61   \_sdef{\_mt:m1:#1}{#2}\_sdef{\_mt:m2:#1}{#3}\_sdef{\_mt:m3:#1}{#4}%
62   \_sdef{\_mt:m4:#1}{#5}\_sdef{\_mt:m5:#1}{#6}\_sdef{\_mt:m6:#1}{#7}%
63   \_monthwB #1
64 }
65 \_def \_monthwB #1 #2 #3 #4 #5 #6 #7 {%
66   \_sdef{\_mt:m7:#1}{#2}\_sdef{\_mt:m8:#1}{#3}\_sdef{\_mt:m9:#1}{#4}%
67   \_sdef{\_mt:m10:#1}{#5}\_sdef{\_mt:m11:#1}{#6}\_sdef{\_mt:m12:#1}{#7}%
68 }
69 \_def \_today{\_mtext{today}}
70 \_public \_today ;
```

Quotes should be tagged by  $\backslash\langle text \rangle$  and  $\backslash'\langle text \rangle'$  if  $\backslash\langle iso-code \rangle$ quotes is declared at beginning of the document (for example  $\backslash$ enquotes). If not, then the control sequences  $\backslash$ " and  $\backslash$ ' are undefined. Remember, that they are used in another meaning when the  $\backslash$ oldaccents command is used. The macros  $\backslash$ " and  $\backslash$ ' are not defined as  $\backslash$ protected because we need their expansion when  $\backslash$ outlines are created. User can declare quotes by  $\backslash$ quoteschars $\langle clqq \rangle\langle crqq \rangle\langle clq \rangle\langle crq \rangle$ , where  $\langle clqq \rangle \dots \langle crqq \rangle$  are normal quotes and  $\langle clq \rangle \dots \langle crq \rangle$  are alternative quotes. or use  $\backslash$ altquotes to swap between the meaning of these two types of quotes.  $\backslash$ enquotes,  $\backslash$ csquotes,  $\backslash$ frquotes,  $\backslash$ dequotes,  $\backslash$ skquotes are defined here. Languages in general provide the  $\backslash$ quotes declaration macro. It declares the quotation marks depending on the actual selected language. For example,  $\backslash$ eslang  $\backslash$ quotes declares Spanish language including its quotation marks used for  $\backslash\langle text \rangle$  and  $\backslash'\langle text \rangle'$ . The language-dependent quotation marks should be declared by  $\backslash$ quotationmarks  $\langle lang-tag \rangle \{ \langle clqq \rangle\langle crqq \rangle\langle clq \rangle\langle crq \rangle \}$  in the lang-data.opm file.

languages.opm

```
92 \_def \_enquotes {\_quoteschars ""''}
93 \_def \_csquotes {\_quoteschars "","'"}
94 \_def \_frquotes {\_quoteschars \_flqq\_frqq\_flq\_frq}
95 \_let \_dequotes = \_csquotes
96 \_let \_skquotes = \_csquotes
97
98 \_def \_quotes {\_trycs{\_qt:\_trycs{\_lan:\_the\_language}{en}}{\_enquotes}}
99 \_def \_quotationmarks #1 #2{\_sdef{\_qt:#1}{\_quoteschars #2}}
100
101 \_public \_quotes \_enquotes \_csquotes \_frquotes \_dequotes \_skquotes ;
```

The  $\backslash$ quoteschars $\langle lqq \rangle\langle rqq \rangle\langle lq \rangle\langle rq \rangle$  defines  $\backslash$ " and  $\backslash$ ' as  $\backslash$ qqA in normal mode and as expandable macros in outline mode. We want to well process the common cases:  $\backslash$ "~&" or  $\backslash$ "~{~". This is the reason why the quotes parameter is read in verbatim mode and retokenized again by  $\backslash$ scantextokens. We want to allow to quote the quotes mark itself by  $\backslash$ "{~"~". This is the reason why the sub-verbatim mode is used when the first character is { in the parameter.

The  $\backslash$ " is defined as  $\backslash$ qqA $\backslash$ qqB $\langle lqq \rangle\langle rqq \rangle$  and  $\backslash$ ' as  $\backslash$ qqA $\backslash$ qqC $\langle lq \rangle\langle rq \rangle$ . The  $\backslash$ qqA $\backslash$ qqB $\langle clqq \rangle\langle crqq \rangle$  runs  $\backslash$ qqB $\langle lqq \rangle\langle rqq \rangle\langle text \rangle$ .

The  $\backslash$ regquotes $\backslash$ " $\langle L \rangle\langle R \rangle$  does  $\backslash$ def $\backslash$ "#1" $\{ \langle L \rangle\#1\langle R \rangle \}$  for outlines but the " separator is active (because " and ' are active in  $\backslash$ pdfunidef).

languages.opm

```
117 \_def \_quoteschars #1#2#3#4{\_def \_altquotes{\_quoteschars#3#4#1#2}\_public \_altquotes;%
118   \_protected\_def \{"{\_qqA\_qqB#1#2}\_protected\_def \'\{\_qqA\_qqC#3#4}%
119   \_regmacro{}{}{\_regquotes\{"#1#2\_regquotes\'\#3#4}}
120
121 \_def \_qqA#1#2#3{\_bgroup\_setverb \_catcode\` =10
122   \_isnextchar\_bgroup{\_catcode\`{=1 \_catcode\`}=2 #1#2#3}{#1#2#3}}
123 \_def \_qqB#1#2#3'\{\_egroup#1\_scantextokens{#3}#2}
124 \_def \_qqC#1#2#3'\{\_egroup#1\_scantextokens{#3}#2}
125 \_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  $\backslash$ activequotes macro and leave quotes without the first backslash. First, declare  $\backslash\langle iso-code \rangle$ quotes, then  $\backslash$ altquotes (if needed) and finally  $\backslash$ activequotes.

languages.opm

```
135 \_def \_activequotes{\_let\_actqq=\_adeft{\_actqq}\_let\_actq=\_adeft{\_actq}%
136   \_regmacro{}{}{\_adeft{\_actq}\_adeft{\_actq}}}
137
138 \_public \_quoteschars \_activequotes ;
```

## 2.37.3 Languages declaration

lang-decl.opm

```
3 \_codedecl \langlist {Languages declaration <2025-05-12>} % preloaded in format
```

`\_preplang`  $\langle lang-id \rangle$   $\langle LongName \rangle$   $\langle lang-tag \rangle$   $\langle hyph-tag \rangle$   $\langle lr-hyph \rangle$  declares a new language. The parameters (separated by space) are

- $\langle lang-id \rangle$ : language identifier. It should be derived from ISO 639-1 code but additional letters can be eventually added because  $\langle lang-id \rangle$  must be used uniquely in the whole declaration list. The `\_preplang` macro creates the language switch `\_ $\langle lang-id \rangle$ lang` and defines also `\_ $\langle lang-id \rangle$ lang` as a macro which expands to `\_ $\langle lang-id \rangle$ lang`. For example, `\_preplang cs Czech ...` creates `\_cslang` as the language switch and defines `\def\_cslang{\_cslang}`.
- $\langle LongName \rangle$ : full name of the language.
- $\langle lang-tag \rangle$ : language tag, which is used for setting language-dependent phrases and sorting data. If a language have two or more hyphenation patterns but a single phrases set, then we declare this language more than once with the same  $\langle lang-tag \rangle$  but different  $\langle lang-hyph \rangle$ .
- $\langle hyph-tag \rangle$ : a part of the file name where the hyphenation patterns are prepared in Unicode. The full file name is `hyph- $\langle hyph-tag \rangle$ .tex`. If  $\langle hyph-tag \rangle$  is `{}` then no hyphenation patterns are loaded.
- $\langle lr-hyph \rangle$ : two digits, they denote `\lefthyphenmin` and `\righthyphenmin` values.

`\_preplang` allocates a new internal number by `\newlanguage\_ $\langle lang-id \rangle$ Patt` which will be bound to the hyphenation patterns. But the patterns nor other language data are not read at this moment. The `\_ $\langle lang-id \rangle$ lang` is defined as `\_langinit`. When the `\_ $\langle lang-id \rangle$ lang` switch is used firstly in a document then the language is initialized, i.e. hyphenation patterns and language-dependent data are read. The `\_ $\langle lang-id \rangle$ lang` is re-defined itself after such initialization. `\_preplang` does also `\def\_ulan:⟨longname⟩ {⟨lang-id⟩}`, this is needed for the `\uselanguage` macro.

lang-decl.opm

```
37 \_def\_preplang #1 #2 #3 #4 #5#6{% lang-id LongName lang-tag hyph-tag lr-hyph
38   \_ifcsname\_#1lang\_endcsname\_else
39     \_ea\_newlanguage\_csname\_#1Patt\_endcsname
40     \_xdef\_langlist{\_langlist\_space#1(#2)}%
41   \_fi
42   \_lowercase{\_sxdef\_ulan:#2}{#1}%
43   \_slet{#1lang}{\_relax}%
44   \_sxdef {#1lang}{\_cs{#1lang}}%
45   \_sxdef {#1lang}{\_noexpand\_langinit\_cs{#1lang}#1(#2)#3[#4]#5#6}%
46 }
```

The `\_preplang` macro adds  $\langle lang-id \rangle$  ( $\langle LongName \rangle$ ) to the `\_langlist` macro which is accessible by `\langlist`. It can be used for reporting declared languages.

lang-decl.opm

```
53 \_def\_langlist{\_langlist}
54 \_def\_langlist{en(USEnglish)}
```

All languages with hyphenation patterns provided by T<sub>E</sub>Xlive are declared here. The language switches `\cslang`, `\sklang`, `\delang`, `\pllang` and many others are declared. You can declare more languages by `\_preplang` in your document, if you want.

The usage of `\_preplang` with  $\langle lang-id \rangle$  already declared is allowed. The language is re-declared in this case. This can be used in your document before first usage of the `\_ $\langle lang-id \rangle$ lang` switch.

lang-decl.opm

67 %	lang-id	LongName	lang-tag	hyph-tag	lr-hyph
68 <code>\_preplang</code>	enus	USenglishmax	en	en-us	23
69 % Europe:					
70 <code>\_preplang</code>	engb	UKenglish	en	en-gb	23
71 <code>\_preplang</code>	be	Belarusian	be	be	22
72 <code>\_preplang</code>	bg	Bulgarian	bg	bg	22
73 <code>\_preplang</code>	ca	Catalan	ca	ca	22
74 <code>\_preplang</code>	hr	Croatian	hr	hr	22
75 <code>\_preplang</code>	cs	Czech	cs	cs	23
76 <code>\_preplang</code>	da	Danish	da	da	22
77 <code>\_preplang</code>	nl	Dutch	nl	nl	22
78 <code>\_preplang</code>	et	Estonian	et	et	23
79 <code>\_preplang</code>	fi	Finnish	fi	fi	22
80 <code>\_preplang</code>	fis	schoolFinnish	fi	fi-x-school	11
81 <code>\_preplang</code>	fr	French	fr	fr	22

82	\_preplang	de	nGerman	de	de-1996	22
83	\_preplang	deo	oldGerman	de	de-1901	22
84	\_preplang	gsw	swissGerman	de	de-ch-1901	22
85	\_preplang	elm	monoGreek	el	el-monoton	11
86	\_preplang	elp	Greek	el	el-polyton	11
87	\_preplang	grc	ancientGreek	grc	grc	11
88	\_preplang	hu	Hungarian	hu	hu	22
89	\_preplang	is	Icelandic	is	is	22
90	\_preplang	ga	Irish	ga	ga	23
91	\_preplang	it	Italian	it	it	22
92	\_preplang	la	Latin	la	la	22
93	\_preplang	lac	classicLatin	la	la-x-classic	22
94	\_preplang	lal	liturgicalLatin	la	la-x-liturgic	22
95	\_preplang	lv	Latvian	lv	lv	22
96	\_preplang	lt	Lithuanian	lt	lt	22
97	\_preplang	mk	Macedonian	mk	mk	22
98	\_preplang	pl	Polish	pl	pl	22
99	\_preplang	pt	Portuguese	pt	pt	23
100	\_preplang	ro	Romanian	ro	ro	22
101	\_preplang	rm	Romansh	rm	rm	22
102	\_preplang	ru	Russian	ru	ru	22
103	\_preplang	srl	Serbian	srlatn	sh-latn	22
104	\_preplang	src	SerbianCyril	srcyrl	sh-cyrl	22
105	\_preplang	sk	Slovak	sk	sk	23
106	\_preplang	sl	Slovenian	sl	sl	22
107	\_preplang	es	Spanish	es	es	22
108	\_preplang	sv	Swedish	sv	sv	22
109	\_preplang	uk	Ukrainian	uk	uk	22
110	\_preplang	cy	Welsh	cy	cy	23
111	% Others:					
112	\_preplang	af	Afrikaans	af	af	12
113	\_preplang	hy	Armenian	hy	hy	12
114	\_preplang	as	Assamese	as	as	11
115	\_preplang	eu	Basque	eu	eu	22
116	\_preplang	bn	Bengali	bn	bn	11
117	\_preplang	nb	Bokmal	nb	nb	22
118	\_preplang	cop	Coptic	cop	cop	11
119	\_preplang	cu	churchslavonic	cu	cu	12
120	\_preplang	eo	Esperanto	eo	eo	22
121	\_preplang	ethi	Ethiopic	ethi	mul-ethi	11
122	\_preplang	fur	Friulan	fur	fur	22
123	\_preplang	gl	Galician	gl	gl	22
124	\_preplang	ka	Georgian	ka	ka	12
125	\_preplang	gu	Gujarati	gu	gu	11
126	\_preplang	hi	Hindi	hi	hi	11
127	\_preplang	id	Indonesian	id	id	22
128	\_preplang	ia	Interlingua	ia	ia	22
129	\_preplang	kn	Kannada	kn	kn	11
130	\_preplang	kmr	Kurmanji	kmr	kmr	22
131	\_preplang	ml	Malayalam	ml	ml	11
132	\_preplang	mr	Marathi	mr	mr	11
133	\_preplang	mn	Mongolian	mn	mn-cyrl	22
134	\_preplang	nn	Nynorsk	nn	nn	22
135	\_preplang	oc	Occitan	oc	oc	22
136	\_preplang	or	Oriya	or	or	11
137	\_preplang	pi	Pali	pi	pi	12
138	\_preplang	pa	Panjabi	pa	pa	11
139	\_preplang	pms	Piedmontese	pms	pms	22
140	\_preplang	zh	Pinyin	zh	zh-latn-pinyin	11
141	\_preplang	sa	Sanskrit	sa	sa	13
142	\_preplang	ta	Tamil	ta	ta	11
143	\_preplang	te	Telugu	te	te	11
144	\_preplang	th	Thai	th	th	23
145	\_preplang	tr	Turkish	tr	tr	22
146	\_preplang	tk	Turkmen	tk	tk	22
147	\_preplang	hsb	Uppersorbian	hsb	hsb	22
148						
149	\_preplang	he	Hebrew	he	{}	00

`\_prelangmore`  $\langle lang-id \rangle \langle space \rangle \{ \langle text \rangle \}$  declares more activities of the language switch. The  $\langle text \rangle$  is processed whenever `\_lang` is invoked. If `\_prelangmore` is not declared for given language then `\_langdefault` is processed.

You can implement selecting a required script for given language, for example:

```
\_prelangmore ru {\_frenchspacing \_setff{script=cyrl}\selectcyrfont}
\_addto\_langdefault {\_setff{}\selectlatnfont}
```

The macros `\selectcyrfont` and `\selectlatnfont` are not defined in OpTeX. If you follow this example, you have to define them after your decision what fonts will be used in your specific situation.

```
167 \_def\_prelangmore #1 #2{\_ea \_gdef \_csname \_langspecific:#1\_endcsname{#2}}
168
169 \_prelangmore en {\_nonfrenchspacing}
170 \_prelangmore enus {\_nonfrenchspacing}
171 \_def\_langdefault {\_frenchspacing}
```

lang-decl.opm

The `\_langreset` is processed before macros declared by `\_prelangmore` or before `\_langdefault`. If you set something for your language by `\_prelangmore` then use `\_def\_langreset{\_settings}` in this code too in order to return default values for all other languages. See cs part of `lang-data.opm` file for an example.

```
181 \_def\_langreset {}
```

lang-decl.opm

The default `\language=0` is US-English with original hyphenation patterns preloaded in the format (see the end of section 2.10). We define `\_enlang` and `\enlang` switches. Note that if no language switch is used in the document then `\language=0` and US-English patterns are used, but `\nonfrenchspacing` isn't set.

```
192 \_chardef\_enPatt=0
193 \_sdef{\_lan:0}{en}
194 \_sdef{\_ulan:usenglish}{en}
195 \_def\_enlang{\_uselang{en}\_enPatt23} % \lefthyph=2 \righthyph=3
196 \_def\enlang{\_enlang}
```

lang-decl.opm

The list of declared languages are reported during format generation.

```
202 \_message{Declared languages: \_langlist.
203 Use \_string\<lang-id>lang to initialize language,
204 \_string\cslang\space for example.}
```

lang-decl.opm

Each language switch `\_lang` defined by `\_prelang` has its initial state `\_langinit`  $\langle switch \rangle \langle lang-id \rangle (\langle LongName \rangle) \langle lang-tag \rangle [\langle hyph-tag \rangle] \langle lr-hyph \rangle$ . The `\_langinit` macro does:

- The internal language  $\langle number \rangle$  is extracted from `\_the\_langPatt`.
- `\_def \_lan: \langle number \rangle { \langle lang-tag \rangle }` for mapping from `\language` number to the  $\langle lang-tag \rangle$ .
- loads `hyph- \langle hyph-tag \rangle .tex` file with hyphenation patterns when `\language= \langle number \rangle`.
- loads the part of `lang-data.opm` file with language-dependent phrases using `\_langinput`.
- `\_def \_lang { \_uselang{ \langle lang-id \rangle } \_langPatt \langle lr-hyph \rangle }`, i.e. the switch redefines itself for doing a “normal job” when the language switch is used repeatedly.
- Runs itself (i.e. `\_lang`) again for doing the “normal job” firstly.

```
223 \_def\_langinit #1#2(#3)#4[#5]#6#7{% \_switch lang-id(LongName)lang-tag[hyph-file]lr-hyph
224 \_sxdef{\_lan:\_ea\_the\_csname \_#2Patt\_endcsname}{#4}%
225 \_begingroup \_setbox0=\_vbox{% we don't want spaces in horizontal mode
226 \_setctable\_optexcatcodes
227 % loading patterns:
228 \_language=\cs{\_#2Patt}\_relax
229 \_ifx\#5\_else
230 \_wlog{Loading hyphenation for #3: \_string\language=\_the\_language\_space(#5)}%
231 \_let\patterns=\_patterns \_let\hyphenation=\_hyphenation \_def\message#1{%
232 \_isfile {hyph-#5}\_iftrue \_input{hyph-#5}%
233 \_else \_opwarning{No hyph. patterns #5 for #3, missing package?}\_fi
234 \_fi
235 % loading language data:
236 \_langinput{#4}%
```

lang-decl.opm

```

237 } \endgroup
238 \xdef#1{\noexpand\uselang{#2}\_csname\_#2Patt\_endcsname #6#7}%
239 #1% do language switch
240 }

```

`\uselang{⟨lang-id⟩}\_⟨lang-id⟩Patt ⟨pre-hyph⟩⟨post-hyph⟩` is used as “normal job” of the switch. It sets `\language`, `\lefthyphenmin`, `\righthyphenmin`. Finally, it runs data from `\preplangmore` or runs `\langdefault`.

lang-decl.opm

```

249 \_def\uselang#1#2#3#4{\_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\_relax
250 \_langreset \_def\_langreset{\_trycs{\_langspecific:#1}{\_langdefault}%
251 }

```

The `\uselanguage {⟨LongName⟩}` macro is defined here (for compatibility with e-plain users). Its parameter is case insensitive.

lang-decl.opm

```

258 \_def\uselanguage#1{\_def\_tmp{#1}\_lowercase{\_cs{\_trycs{\_ulan:#1}{0x}lang}}
259 \_sdef{\_Oxlang}{\_opwarning{\_string\uselanguage{\_tmp}: Unknown language name, ignored}}
260 \_public \uselanguage ;

```

## 2.37.4 Data for various languages

The “language data” include declarations of rules for sorting (see section 2.33), language-dependent phrases and quotation marks (see section 2.37.2). The language data are collected in the single `lang-data.opm` file. Appropriate parts of this file is read by `\langinput{⟨lang-tag⟩}`. First few lines of the file looks like:

lang-data.opm

```

3 \_codedecl \_langdata {Language dependent data <2026-05-17>} % only en, cs preloaded in format
4
5 \_langdata en {English} % -----
6 \_langw en Chapter Table Figure Subject
7 \_langb en {, and} {et al.} {\,ed.} {cit.~} {Vol.~} {No.~} {pp.~} {-p.} {,-ed.} {,-eds.}
8 {Available from} {Available also from}
9 {Bachelor's Thesis} {Master's Thesis} {Ph.D. Thesis}
10 \_monthw en January February March April May June
11 July August September October November December
12 \_sdef{\_mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
13 \_quotationmarks en {"'"}
14
15 %\_let \_sortingdataen = \_sortingdatalatin % set already, see section 2.33, makeindex.opm
16 %\_let \_ignoredcharsen = \_ignoredcharsgeneric
17 %\_def \_compoundcharsen {}
18
19 \_langdata cs {Czech} % -----
20 % Chapter Table Figure Subject
21 \_langw cs Kapitola Tabulka Obrázek Věc
22 % {, and} {et al.} {\,ed.} {cit.~} {Vol.~} {No.~} {pp.~} {-p.} {,-ed.} {,-eds.}
23 % {Available from} {Available also from}
24 % {Bachelor's Thesis} {Master's Thesis} {Ph.D. Thesis}
25 \_langb cs {a} {a-kol.} {\,vyd.} {vid.~} {ročník~} {č.~} {s.~} {-s.} {,-editor} {,-editoři}
26 {Dostupné na} {Dostupné též na}
27 {Bakalářská práce} {Diplomová práce} {Disertační práce}
28 % January February March April May June
29 % July August September October November December
30 \_monthw cs ledna února března dubna května června
31 července srpna září října listopadu prosince
32 \_sdef{\_mt:today:cs}{\_the\_day.~\_mtext{m\_the\_month} \_the\_year} % date format
33 \_quotationmarks cs {"","'"}
34 \_preplangmore cs {\_frenchspacing \_postexhyphenchar=`-
35 \_def\_langreset{\_postexhyphenchar=0}}
36
37 \_let \_sortingdatacs = \_sortingdatalatin
38 \_let \_ignoredcharscs = \_ignoredcharsgeneric
39 \_def \_compoundcharscs {ch:~T Ch:~U CH:~V} % see \_compoundchars in section 2.33
40
41
42 \_langdata de {German} % -----
43 \_langw de Kapitel Tabelle Abbildung Betreff

```



```

44 \_quotationmarks de {"",'}
45 %todo
46 \_let \_sortingdatade = \_sortingdatalatin
47 \_let \_ignoredcharsde = \_ignoredcharsgeneric
48 \_def \_compoundcharsde {ß:ss}
49 \_def \_xcompoundcharsde {} % ß is interpreted in second pass of sorting
...etc. (see lang-data.opm)

```

There are analogical declaration for more languages here. Unfortunately, this file is far for completeness. I welcome you send me a part of declaration for your language.

If your language is missing in this file then a warning is reported during language initialization. You can create your private declaration in your macros (analogical as in the `lang-data.opm` file but without the `\_langdata` prefix). Then you will want to remove the warning about missing data. This can be done by `\_nolanginput{<lang-tag>}` given before initialization of your language.

The whole file `lang-data.opm` is not preloaded in the format because I suppose a plenty languages here and I don't want to waste the T<sub>E</sub>X memory by these declarations. Each part of this file prefixed by `\_langdata <lang-tag> {<LongName>}` is read separately when `\_langinput{<lang-tag>}` is used. And it is used in the `\_langinit` macro (i.e. when the language is initialized), so the appropriate part of this file is read automatically on demand.

If the part of the `lang-data.opm` concerned by `<lang-tag>` is read already then `\_li:<lang-tag>` is set to R and we don't read this part of the file again.

lang-decl.opm

```

296 \_def\_langinput #1{%
297   \_unless \_ifcsize _li:#1\_endcsize
298     \_bgroup
299       \_edef\_tmp{\_noexpand\_langdata #1 }\_everyeof\_ea{\_tmp{}}%
300       \_long \_ea\_def \_ea\_tmp \_ea##\_ea1\_tmp{\_readlangdata{#1}}%
301       \_globaldefs=1
302       \_ea\_tmp \_input{lang-data.opm}%
303       \_ea\_glet \_csize _li:#1\_endcsize R%
304     \_egroup
305   \_fi
306 }
307 \_def\_readlangdata #1#2{%
308   \_ifx^#2\_opwarning{Missing data for language "#1" in lang-data.opm}%
309   \_else \_wlog{Reading data for the language #2 (#1)}%
310   \_fi
311 }
312 \_def\_langdata #1 #2{\_endinput}
313 \_def\_nolanginput #1{\_ea\_glet \_csize _li:#1\_endcsize N}
314 \_public \_nolanginput ;

```

Data of two preferred languages are preloaded in the format:

lang-decl.opm

```

320 \_langinput{en} \_langinput{cs}

```

## 2.38 Other macros

Miscellaneous macros are here.

others.opm

```

3 \_codedecl \_uv {Miscellaneous <2026-05-17>} % 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\_usesecond\_currpage}
18 \_def\_currpage {{0}{?}}
19 \_public \_lastpage \_totalpages ;

```

We need `\_uv`, `\_clqq`, `\_crqq`, `\_clq`, `\_crq`, `\_flqq`, `\_frqq`, `\_flq`, `\_frq`, `\_uslang`, `\_ehyph` `\_chyph`, `\_shyph`, for backward compatibility with  $\mathcal{S}$ plain. Codes are set according to Unicode because we are using Czech only in Unicode when LuaT<sub>E</sub>X is used.

```

29
30 % \uv, \promile, \clqq, \crqq for compatibility with csplain:
31
32 \_chardef\clqq=8222 \_chardef\crqq=8220 \_chardef\clq=8218 \_chardef\crq=8216
33 \_def\flqq{«\_kern.3em } \_def\frqq{\_kern.3em»} % including traditional nobreak space, see
34 \_def\flq{\"\_kern.15em } \_def\frq{\_kern.15em"} % Lexique des Règles typographiques, page 149
35 \_chardef\promile=8240
36 \_private \flqq \frqq \flq \frq ; % used in \frquotes
37
38 \_def\uv#1{„#1"}
39
40 \_let\uslang=\enlang \_let\ehyph=\enlang
41 \_let\chyph=\cslang \_let\shyph=\sklang
42 \_let\csUnicode=\csPatt \_let\czUnicode=\csPatt \_let\skUnicode=\skPatt

```

The `\letfont` was used in `ℳplain` instead of `\fontlet`.

```

48 \_let \letfont = \fontlet

```

Non-breaking space in Unicode.

```

54 \_let ^^a0=~

```

Old macro packages need these funny control sequences. We don't use them in new macros.

```

61 \_catcode`\@=11
62 \_let\z@=\_zo \_let\z@skip=\_zoskip
63 \_newdimen\p@ \p@=1pt
64 \_toksdef\toks@=0
65 \_let\voidb@x=\_voidbox
66 \_chardef\@ne=1 \_chardef\tw@=2 \_chardef\thr@=3 \_chardef\sixt@@n=16
67 \_mathchardef\@m=1000 \_mathchardef\@M=10000 \_mathchardef\@MMM=20000
68 \_countdef\@mne=22 \m@ne=-1
69 \_chardef\@cclv=255 \_mathchardef\@ccclvi=256
70 \_skipdef\skip@=0
71 \_dimendef\dimen@=0 \_dimendef\dimen@i=1
72 \_dimendef\dimen@ii=2
73 \_countdef\count@=255
74 \_def\m@th{\_mathsurround\z@}
75 \_def\o@lign{\_lineskiplimit\z@ \_oalign}
76 \_def\n@space{\_nulldelimiterspace\z@ \m@th}
77 \_newdimen\p@renwd \p@renwd=8.75pt
78 \_def\alloc@#1#2#3#4#5{\_allocator#5{\_csstring#2}#3}
79 \_catcode`\@=12

```

We don't want to read `opmac.tex` unless `\input opmac` is specified.

```

85 \_def\OPmacversion{0pTeX}

```

We allow empty lines in math formulae. It is more comfortable.

```

91 \_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  $\text{\LaTeX}$  file `lipsum.ltd.tex` by `\_lipsumload` and prints the selected paragraph(s). Next usages of `\lipsum` prints the selected paragraph(s) from memory. `\lipsum` is fully expandable.

`\lipsum` adds `\_par` after each printed paragraph. If you don't need such `\_par` here, use `\lipsumtext[⟨number⟩]` or `\lipsum[⟨number⟩.]` (i.e. dot after the parameter). The first case prints the paragraph `⟨number⟩` without the final `\_par` and the second case prints only first sentence from the paragraph `⟨number⟩` using `\_lipsumdot`.

```

110 \_newbox\_nonebox
111 \_def\_lipsumtext[#1]{\_lipsumload\_cs{\_lip:#1}}
112 \_def\_lipsum[#1]{\_lipsumA #1.}{#1}}
113 \_def\_lipsumA #1.#2}#3{\_ifx^#2\_lipsumB #1\_empty-\_empty\_fin \_else \_lipsumdot[#1].\_fi}
114 \_def\_lipsumB #1-#2\_empty#3\_fin{%
115 \_forum #1..\_ifx^#2^#1\_else#2\_fi \_do {\_lipsumtext[##1]\_par}}

```

```

116 \def\lipsumfile{lipsum.ltd.tex}
117 \def\lipsumload{\beglocalcontrol
118   {\setbox\nonebox=\vpack{\tmpnum=0 % vertical mode during \input lipsum.ltd.tex
119     \def\ProvidesFile##1[##2]{}%
120     \def\SetLipsumLanguage##1{}%
121     \def\NewLipsumPar{\incr\tmpnum \sxddef{lip:\the\tmpnum}}}%
122     \opinput\lipsumfile
123     \glet\lipsumload=\empty
124   }}%
125 \endlocalcontrol}
126 \def\lipsumdot[#1]{\lipsumload \ea\ea\ea \lipsumdotA \csname lip:#1\endcsname.\fin}
127 \def\lipsumdotA #1.#2\fin {#1}
128
129 \public \lipsum \lipsumtext ;
130 \let \lorem=\lipsum

```

Selected macros from `OpTeX tricks` are registered using `\regtrick⟨cs-name⟩`. The `⟨cs-name⟩` is defined as `\loadtrick ⟨cs-name⟩ ⟨cs-name⟩`. When a user runs such a registered `⟨cs-name⟩` then `\loadtrick ⟨cs-name⟩` reads the appropriate code from the file `optex-tricks.opm` and the `⟨cs-name⟩` is redefined. Finally, `⟨cs-name⟩` is run again.

The `optex-tricks.opm` file includes blocks started by `\_trick` followed by the declared `⟨cs-names⟩` followed by semicolon followed by the code with declarations of `⟨cs-names⟩` itself. The next `\_trick` does `\endpinput` of the file. The file is read inside temporary `\vbox` with `\globaldefs=1` because it can be read inside horizontal mode and/or inside a group. The `optextrick` name space is used during reading the code from the file. Only registered control sequences are re-defined directly in user name space.

You can load a code chunk by `\loadtrick ⟨cs-name⟩`. This command doesn't run the `⟨cs-name⟩`, only loads the appropriate code. It should be usable if you want to load the code before the first usage of the `⟨cs-name⟩`.

```

152 \def\regtrick#1{\ifx#1\undefined\def#1{\loadtrickD#1}\else\_badtrick\_fi}
153 \def\loadtrickD#1{\loadtrick#1#1}
154 \def\loadtrick#1{\beglocalcontrol
155   \resetnamespace{optextrick}\setctable\optexcatcodes
156   \savecatcodetable\tmpcatcodes \catcodetable\tmpcatcodes
157   \long\def\loadtrickA ##1\_trick##2#1##3;{##1}%
158   \wlog{Loading trick macros for \string#1}%
159   \setbox\nonebox=\vpack{\let\:=\_trickprefix \def\_trickseq{#1}%
160     \globaldefs=1 \ea\loadtrickA \input {optex-tricks.opm}}}%
161   \restoretable \endnamespace
162   \endlocalcontrol
163 }
164 \def\_trick #1;{\endinput}
165 \public \loadtrick ;
166
167 \xargs \regtrick \begfile \createfile \beglua \begLUA \logginglua
168   \sethours \setminutes \setseconds \setweekday \Othe \ordnum \showpglists \shownodes \runsystem
169   \directoutput \algol \algolkw \algolcap \algolnum \makeLOA \setaca \scaleto \scaletoof
170   \ttliref \lref \easylst \keepstyle \fread \shadedframe \roundframe \cancel \ignoreinspic
171   \keystroke \colortab \crx \crtop \crbot \crmid \longtable \vcent \vbot \tnote
172   \tabnodes \tablebefore \framedblocks \twoblocks \pstart \settabs \import \incrrp \ispageodd
173   \iniseccc \seccc \makeLOF \makeLOT \captionF \captionT \correctsize \pgforeground
174   \onlyifnew \thedimen \rebox \leftfill \rightfill \lrfill \directchar \ereplstring
175   \xreplstring \replmacro \tdnum \clipbox \ismatch \isinmacro \mathinexpr \xparshape
176   \csvread \csvdecl \dbcreate \inode \enode \excedges \showtree \jsonread \jsoncnv
177   \treedata \tracingtreedata \treenodes \ul \inspicto \inspictoh
178   \verbhyphens \verbhyphenpenalty ;
179 \sdef{item:m}{\loadtrick{\style m}\cs{item:m}}

```

The `\:` is used as a prefix before `\def` etc. in the file `optex-tricks.opm` if the declared control sequence is from user namespace. It checks if a user have defined it before the trick is loaded and prints warning in this case. The `\:` is set to `\_trickprefix` in the group before trick loading. If a trick includes only one autoloaded control sequence then it needs not to be prefixed by `\:` because if a user redefines it then the usage of it doesn't run autoloading.

```

191 \def\_trickprefix #1#2{\ifx#2\undefined \else \def\_tmp{\loadtrickD#2}\ifx\_tmp#2\_else
192   \opwarning{trick by \ea\string\_trickseq\_space loaded: \string#2 redefined}\fi\_fi #1#2}

```

LuaTeX version 1.14 and newer provides `\partokenname` which allows to specify something different than `\par` at empty lines. We set `\_par` (see below) in OpTeX version 1.04+ and newer. Some macros were rewritten due to this change. And we copy old versions of these changed macros here in order to allow to use older LuaTeX versions where `\partokenname` is not provided.

Note that your macros where a parameter is separated by the empty line must be changed too. Use `\def\macro #1\_par{...}` instead `\def\macro #1\par{...}`.

others.opm

```

206 \ifx\_partokenname\_undefined % LuaTeX 1.13 or older:
207
208 \def\_incaption {\_bgroup
209   \ifcsname\_\_tmpa num\_endcsname \_ea\_incr \_csname\_\_tmpa num\_endcsname
210   \else \_opwarning{Unknown caption /\_tmpa}\_fi
211   \edef\_thecapnum {\_csname\_the\_tmpa num\_endcsname}%
212   \edef\_thecapttitle{\_mtext{\_tmpa}}}%
213   \_ea\_the \_csname\_everycaption\_tmpa\_endcsname
214   \def\_par{\_nbpar\_egroup}\_let\par=\_par
215   \_cs{\_printcaption}\_tmpa}%
216 }
217 \def\_boxlines{%
218   \def\_boxlinesE{\_ifhmode\_egroup\_empty\_fi}%
219   \def\_nl{\_boxlinesE}%
220   \_bgroup \_lccode\~=\^M\_lowercase{\_egroup\_let~}\_boxlinesE
221   \_everypar{\_setbox0=\_lastbox\_endgraf
222     \_hbox\_bgroup \_catcode\^M=13 \_let\par=\_nl \_aftergroup\_boxlinesC}%
223 }
224 \def\_letter{
225   \def\_address{\_vtop\_bgroup\_boxlines \_parskip=0pt \_let\par=\_egroup}
226   \def\_subject{{\_bf\_mtext{subj}: }}
227   \_public \_address \_subject ;
228   \_typesize[11/14]
229   \_vsize=\_dimexpr \_topskip + 49\_baselineskip \_relax % added 2020-03-28
230   \_parindent=0pt
231   \_parskip=\_medskipamount
232   \_nopagenumbers
233 }
234 \def\_printverblinenum#1{\_puttptpenalty \_indent \_printverblinenum \_kern\_ttshift #1\par}
235 \_public \_boxlines \_letter ;
236
237 \_else % LuaTeX 1.14 or newer:

```

We set `\partokenname` to `\_par` in order to keep the name `\par` in the public namespace for end users. I.e. a user can say `\def\par{paragraph}` for example without crash of processing the document. See section 2.2.1 for more details about the name space concept.

Moreover, we set `\partokencontext` to one in order to the `\_par` token is inserted not only at empty lines, but also at the end of `\vbox`, `\vtop` and `\vcenter` if horizontal mode is opened here. This differs from default TeX behavior where horizontal mode is closed in these cases without inserting `par` token.

We set `\_partokenset` to defined value 1 in order to the macro programmer can easily check these settings in OpTeX format by `\ifx\_partokenset\_undefined ... \else ... \fi`.

others.opm

```

254 \_partokenname\_par
255 \_partokencontext=1
256 \_let\_partokenset=1
257 \_fi

```

## 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 L<sup>A</sup>T<sub>E</sub>X 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.

The allocations are declared in subsection 2.39.2, callbacks are implemented in subsection 2.39.3 and handling with colors can be found in the subsection 2.39.6.

Local imports of standard Lua or Lua<sub>TeX</sub> functions and constants are the first thing we do. They mostly serve one of several purposes:

- shorthands for commonly used functions,
- speed of access for commonly executed functions,
- common constants to be used consistently,
- use of local definition that can't be externally overwritten.

optex.lua

```

14
15 local fmt = string.format
16
17 local node_id = node.id
18 local node_subtype = node.subtype
19 local flush_node = node.flush_node
20 local flush_list = node.flush_list
21 local node_traverse = node.traverse
22 local glyph_id = node_id("glyph")
23 local rule_id = node_id("rule")
24 local glue_id = node_id("glue")
25 local hlist_id = node_id("hlist")
26 local vlist_id = node_id("vlist")
27 local disc_id = node_id("disc")
28 local kern_id = node_id("kern")
29 local whatsit_id = node_id("whatsit")
30 local pdfliteral_id = node_subtype("pdf_literal")
31 local pdfsave_id = node_subtype("pdf_save")
32 local pdfrestore_id = node_subtype("pdf_restore")
33 local token_getmacro = token.get_macro
34
35 local direct = node.direct
36 local todirect = direct.todirect
37 local tonode = direct.tonode
38 local getfield = direct.getfield
39 local setfield = direct.setfield
40 local getwhd = direct.getwhd
41 local getid = direct.getid
42 local getlist = direct.getlist
43 local setlist = direct.setlist
44 local getleader = direct.getleader
45 local getattribute = direct.get_attribute
46 local setattribute = direct.set_attribute
47 local insertbefore = direct.insert_before
48 local copy = direct.copy
49 local traverse = direct.traverse
50 local one_bp = tex.sp("1bp")
51
52 local tex_print = tex.print
53 local tex_setbox = tex.setbox
54 local tex_getcount = tex.getcount
55 local tex_setcount = tex.setcount
56 local token_scanint = token.scan_int
57 local token_scanlist = token.scan_list
58 local token_setmacro = token.set_macro

```

### 2.39.1 General

Define namespace where “public” Op<sub>T</sub>EX functions will be added.

optex.lua

```

63
64 local optex = _ENV.optex or {}
65 _ENV.optex = optex
66

```

Error function used by following functions for critical errors.

optex.lua

```

68 local function err(...)
69     local message = fmt(...)
70     error("\nerror: " .. message .. "\n")
71 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.

optex.lua

```
75 local function registernumber(name)
76     return token.create(name).index
77 end
78 _ENV.registernumber = registernumber
79 optex.registernumber = registernumber
```

MD5 hash of given file.

optex.lua

```
82 function optex.md5file(file)
83     local fh = io.open(file, "rb")
84     if fh then
85         local data = fh:read("*a")
86         fh:close()
87         tex_print(md5.sumhexa(data))
88     end
89 end
90
```

The `optex.raw_ht()` function measures the height plus depth of given `\vbox` saved by `\setbox`. It solves the “dimension too large” problem with big boxes. It is used in the `\_rawht` macro.

optex.lua

```
94
95 function optex.raw_ht()
96     local nod = tex.box[token_scanint()]    -- read head node of the given box
97     local ht = 0
98     if nod and nod.id == vlist_id then
99         for n in node_traverse(nod.head) do
100             if n.id == hlist_id or n.id == vlist_id or n.id == rule_id then
101                 ht = ht + n.height + n.depth
102             elseif n.id == glue_id then
103                 ht = ht + n.width
104             elseif n.id == kern_id then
105                 ht = ht + n.kern
106             end
107         end
108     end
109     tex_print(fmt("%.0f", ht/65536))
110 end
111
```

## 2.39.2 Allocators

optex.lua

```
114 local alloc = _ENV.alloc or {}
115 _ENV.alloc = alloc
```

An attribute allocator in Lua that cooperates with normal OpTeX allocator.

optex.lua

```
118 local attributes = {}
119 function alloc.new_attribute(name)
120     local cnt = tex.count["_attributealloc"] + 1
121     if cnt > 65534 then
122         tex.error("No room for a new attribute")
123     else
124         tex.setcount("global", "_attributealloc", cnt)
125         texio.write_nl("log", "'..'..name..'='\@attribute'..tostring(cnt))
126         attributes[name] = cnt
127         return cnt
128     end
129 end
```

Allocator for Lua functions (“pseudoprimitives”). It passes variadic arguments (“...”) like “global” to `token.set_lua`.

optex.lua

```
133 local function_table = lua.get_functions_table()
134 local function define_lua_command(csname, fn, ...)
135     local luafnalloc = #function_table + 1
136     token.set_lua(csname, luafnalloc, ...) -- WARNING: needs LuaTeX 1.08 (2019) or newer
```



```

137     function_table[luafnalloc] = fn
138 end
139 _ENV.define_lua_command = define_lua_command
140 optex.define_lua_command = define_lua_command

```

### 2.39.3 Callbacks

```

143 local callback = _ENV.callback or {}
144 _ENV.callback = callback

```

optex.lua

Save `callback.register` function for internal use.

optex.lua

```

147 local callback_register = callback.register
148 function callback.register(name, fn)
149     err("direct registering of callbacks is forbidden, use 'callback.add_to_callback'")
150 end

```

Table with lists of functions for different callbacks.

optex.lua

```

153 local callback_functions = {}

```

Table that maps callback name to a list of descriptions of its added functions. The order corresponds with `callback_functions`.

optex.lua

```

156 local callback_description = {}

```

Table used to differentiate user callbacks from standard callbacks. Contains user callbacks as keys.

optex.lua

```

160 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).

optex.lua

```

165 local default_functions = {}

```

Table that maps standard (and later user) callback names to their types.

optex.lua

```

168 local callback_types = {
169     -- file discovery
170     find_read_file    = "exclusive",
171     find_write_file   = "exclusive",
172     find_font_file    = "data",
173     find_output_file  = "data",
174     find_format_file  = "data",
175     find_vf_file      = "data",
176     find_map_file     = "data",
177     find_enc_file     = "data",
178     find_pk_file      = "data",
179     find_data_file    = "data",
180     find_opentype_file = "data",
181     find_truetype_file = "data",
182     find_type1_file   = "data",
183     find_image_file   = "data",
184
185     open_read_file    = "exclusive",
186     read_font_file    = "exclusive",
187     read_vf_file      = "exclusive",
188     read_map_file     = "exclusive",
189     read_enc_file     = "exclusive",
190     read_pk_file      = "exclusive",
191     read_data_file    = "exclusive",
192     read_truetype_file = "exclusive",
193     read_type1_file   = "exclusive",
194     read_opentype_file = "exclusive",
195
196     -- data processing
197     process_input_buffer = "data",
198     process_output_buffer = "data",
199     process_jobname     = "data",
200     input_level_string  = "data",

```

```

201
202 -- node list processing
203 contribute_filter      = "simple",
204 buildpage_filter      = "simple",
205 build_page_insert     = "exclusive",
206 pre_linebreak_filter  = "list",
207 linebreak_filter      = "exclusive",
208 append_to_vlist_filter = "exclusive",
209 post_linebreak_filter = "reverselist",
210 hpack_filter          = "list",
211 vpack_filter          = "list",
212 hpack_quality         = "list",
213 vpack_quality         = "list",
214 process_rule          = "exclusive",
215 pre_output_filter     = "list",
216 hyphenate             = "simple",
217 ligaturing            = "simple",
218 kerning               = "simple",
219 insert_local_par      = "simple",
220 mlist_to_hlist        = "exclusive",
221
222 -- information reporting
223 pre_dump              = "simple",
224 start_run              = "simple",
225 stop_run              = "simple",
226 start_page_number     = "simple",
227 stop_page_number      = "simple",
228 show_error_hook       = "simple",
229 show_error_message    = "simple",
230 show_lua_error_hook   = "simple",
231 start_file             = "simple",
232 stop_file              = "simple",
233 call_edit              = "simple",
234 finish_synctex         = "simple",
235 wrapup_run            = "simple",
236
237 -- pdf related
238 finish_pdffile         = "data",
239 finish_pdffpage        = "data",
240 page_order_index       = "data",
241 process_pdf_image_content = "data",
242
243 -- font related
244 define_font            = "exclusive",
245 glyph_not_found        = "exclusive",
246 glyph_info             = "exclusive",
247
248 -- undocumented
249 glyph_stream_provider = "exclusive",
250 provide_charproc_data = "exclusive",
251 }

```

Return a list containing descriptions of added callback functions for specific callback.

optex.lua

```

255 function callback.callback_descriptions(name)
256     return callback_description[name] or {}
257 end
258
259 local valid_callback_types = {
260     exclusive = true,
261     simple = true,
262     data = true,
263     list = true,
264     reverselist = true,
265 }

```

Create a user callback that can only be called manually using `call_callback`. A default function is only needed by "exclusive" callbacks.

optex.lua

```

269 function callback.create_callback(name, cbtype, default)

```

```

270 if callback_types[name] then
271     err("cannot create callback '%s' - it already exists", name)
272 elseif not valid_callback_types[cbtype] then
273     err("cannot create callback '%s' with invalid callback type '%s'", name, cbtype)
274 elseif cbtype == "exclusive" and not default then
275     err("unable to create exclusive callback '%s', default function is required", name)
276 end
277
278 callback_types[name] = cbtype
279 default_functions[name] = default or nil
280 user_callbacks[name] = true
281 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).

optex.lua

```

289 local call_callback
290 function callback.add_to_callback(name, fn, description)
291     if user_callbacks[name] or callback_functions[name] or default_functions[name] then
292         -- either:
293         -- a) user callback - no need to register anything
294         -- b) standard callback that has already been registered
295         -- c) standard callback with default function registered separately
296         -- (mlist_to_hlist)
297     elseif callback_types[name] then
298         -- This is a standard luatex callback with first function being added,
299         -- register a proxy function as a real callback. Assert, so we know
300         -- when things break, like when callbacks get redefined by future
301         -- luatex.
302         callback_register(name, function(...)
303             return call_callback(name, ...)
304         end)
305     else
306         err("cannot add '%s' to callback '%s' - no such callback exists", description, name)
307     end
308
309     if not description or description == "" then
310         err("missing description when adding a callback to '%s'", name)
311     end
312
313     local prev_descriptions = callback_description[name] or {}
314     for _, desc in ipairs(prev_descriptions) do
315         if desc == description then
316             err("for callback '%s' there already is '%s' added", name, description)
317         end
318     end
319
320     if callback_types[name] == "exclusive" and #prev_descriptions == 1 then
321         err("can't register '%s' as callback '%s' - exclusive callback occupied by '%s'",
322             description, name, prev_descriptions[1])
323     end
324
325     if type(fn) ~= "function" then
326         err("expected Lua function to be added as '%s' for callback '%s'", description, name)
327     end
328
329     -- add function to callback list for this callback
330     callback_functions[name] = callback_functions[name] or {}
331     table.insert(callback_functions[name], fn)
332
333     -- add description to description list
334     callback_description[name] = callback_description[name] or {}
335     table.insert(callback_description[name], description)
336 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.

```

340 function callback.remove_from_callback(name, description)
341     local descriptions = callback_description[name]
342     local index
343     for i, desc in ipairs(descriptions) do
344         if desc == description then
345             index = i
346             break
347         end
348     end
349
350     if not index then
351         err("can't remove '%s' from callback '%s': not found", description, name)
352     end
353
354     table.remove(descriptions, index)
355     local fn = table.remove(callback_functions[name], index)
356
357     if #descriptions == 0 then
358         -- Delete the list entirely to allow easy checking of "truthiness".
359         callback_functions[name] = nil
360
361         if not user_callbacks[name] and not default_functions[name] then
362             -- this is a standard callback with no added functions and no
363             -- default function (i.e. not mlist_to_hlist), restore standard
364             -- behaviour by unregistering.
365             callback_register(name, nil)
366         end
367     end
368
369     return fn, description
370 end

```

helper iterator generator for iterating over reverselist callback functions

```

373 local function reverse_ipairs(t)
374     local i, n = #t + 1, 1
375     return function()
376         i = i - 1
377         if i >= n then
378             return i, t[i]
379         end
380     end
381 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.

```

390 function callback.call_callback(name, ...)
391     local cbtype = callback_types[name]
392     -- either take added functions or the default function if there is one
393     local functions = callback_functions[name] or {default_functions[name]}
394
395     if cbtype == nil then
396         err("cannot call callback '%s' - no such callback exists", name)
397     elseif cbtype == "exclusive" then
398         -- only one function, at least default function is guaranteed by
399         -- create_callback
400         return functions[1](...)
401     elseif cbtype == "simple" then
402         -- call all functions one after another, no passing of data
403         for _, fn in ipairs(functions) do
404             fn(...)
405         end
406         return
407     elseif cbtype == "data" then
408         -- pass data (first argument) from one function to other, while keeping

```

```

409     -- other arguments
410     local data = (...)
411     for _, fn in ipairs(functions) do
412         data = fn(data, select(2, ...))
413     end
414     return data
415 end
416
417 -- list and reverselist are like data, but "true" keeps data (head node)
418 -- unchanged and "false" ends the chain immediately
419 local iter
420 if cbtype == "list" then
421     iter = ipairs
422 elseif cbtype == "reverselist" then
423     iter = reverse_ipairs
424 end
425
426 local head = (...)
427 local new_head
428 for _, fn in iter(functions) do
429     new_head = fn(head, select(2, ...))
430     if new_head == false then
431         return false
432     elseif new_head ~= true then
433         head = new_head
434     end
435 end
436 return head
437 end
438 call_callback = callback.call_callback

```

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`.

optex.lua

```

444 default_functions["mlist_to_hlist"] = node.mlist_to_hlist
445 callback.create_callback("pre_mlist_to_hlist_filter", "list")
446 callback.create_callback("post_mlist_to_hlist_filter", "reverselist")
447 callback_register("mlist_to_hlist", function(head, ...)
448     -- pre_mlist_to_hlist_filter
449     local new_head = call_callback("pre_mlist_to_hlist_filter", head, ...)
450     if new_head == false then
451         flush_list(head)
452         return nil
453     else
454         head = new_head
455     end
456     -- mlist_to_hlist means either added functions or standard luatex behavior
457     -- of node.mlist_to_hlist (handled by default function)
458     head = call_callback("mlist_to_hlist", head, ...)
459     -- post_mlist_to_hlist_filter
460     new_head = call_callback("post_mlist_to_hlist_filter", head, ...)
461     if new_head == false then
462         flush_list(head)
463         return nil
464     end
465     return new_head
466 end)

```

Create “virtual” callback `pre_append_to_vlist_filter` by setting `append_to_vlist_filter` callback. The default behaviour of `append_to_vlist_filter` is kept by using a default function, but it can still be overridden by using `add_to_callback`.

optex.lua

```

472 default_functions["append_to_vlist_filter"] = function(n, _, prevdepth)
473     return node.prepend_prevdepth(n, prevdepth)
474 end
475 callback.create_callback("pre_append_to_vlist_filter", "list")
476 callback_register("append_to_vlist_filter", function(box, locationcode, prevdepth, mirrored)
477     local current = call_callback("pre_append_to_vlist_filter",

```

```

478             box, locationcode, prevdepth, mirrored)
479     if not current then
480         flush_node(box)
481         return
482     end
483     -- append_to_vlist_filter means either added functions or standard luatex behavior
484     -- of node.prepend_prevdepth (handled by default function)
485     return call_callback("append_to_vlist_filter",
486                         current, locationcode, prevdepth, mirrored)
487 end
488 )

```

For preprocessing boxes just before shipout we define custom callback. This is used for coloring based on attributes. There is however a challenge - how to call this callback? We could redefine `\shipout` and `\pdfxform` (which both run `ship_out` procedure internally), but they would lose their primitive meaning - i.e. `\immediate` wouldn't work with `\pdfxform`. The compromise is to require anyone to run `\_preshipout`*<destination box number>**<box specification>* just before `\shipout` or `\pdfxform` if they want to call `pre_shipout_filter` (and achieve colors and possibly more).

optex.lua

```

499 callback.create_callback("pre_shipout_filter", "list")
500
501 define_lua_command("_preshipout", function()
502     local boxnum = token_scanint()
503     local head = token_scanlist()
504     head = call_callback("pre_shipout_filter", head)
505     tex_setbox(boxnum, head)
506 end)

```

Compatibility with L<sup>A</sup>T<sub>E</sub>X through luatexbase namespace. Needed for luaotfload.

optex.lua

```

510 _ENV.luatexbase = {
511     registernumber = registernumber,
512     attributes = attributes,
513     -- `provides_module` is needed by older version of luaotfload
514     provides_module = function() end,
515     new_attribute = alloc.new_attribute,
516     callback_descriptions = callback.callback_descriptions,
517     create_callback = callback.create_callback,
518     add_to_callback = callback.add_to_callback,
519     remove_from_callback = callback.remove_from_callback,
520     call_callback = callback.call_callback,
521     callbacktypes = {},
522 }

```

`\tracingmacros` callback registered. Use `\tracingmacros=3` or `\tracingmacros=4` if you want to see the result.

optex.lua

```

526 callback.add_to_callback("input_level_string", function(n)
527     if tex.tracingmacros > 3 then
528         return "[" .. n .. "]"
529     elseif tex.tracingmacros > 2 then
530         return "~" .. string.rep(".", n)
531     else
532         return ""
533     end
534 end, "_tracingmacros")

```

## 2.39.4 PDF object utilities

The PDF format defines various kinds of “objects”: numbers, names, arrays, dictionaries. A PDF document is mostly a tree of these objects (i.e. objects contain other objects), with either direct (“in-place”) or indirect references.

These objects are saved in the PDF file in a serialized form, often compressed. While LuaT<sub>E</sub>X takes care of the compression, and most of the structure of the PDF format, sometimes we want to insert our own PDF objects - and there are places where LuaT<sub>E</sub>X allows us to use our own objects to insert anything, but it already expects a serialized form of the objects. The serialized form for some kinds of objects has various formatting and escaping rules, and is inconvenient to produce in T<sub>E</sub>X especially when some of the e.g. names and strings are *user input*.



Here we define a couple of utilities for both Lua and T<sub>E</sub>X that aid with representation and serialization of objects. The functions are exported, but may still evolve in incompatible ways.

A reference for the serialization is the “Syntax” section of the PDF specification<sup>3</sup>.

`\pdfname{⟨name⟩}` serializes a “PDF name” to bytes.

optex.lua

```
560 local function name_escape(char)
561     return fmt("#%02X", char:byte())
562 end
```

optex.lua

```
564 local function pdf_name(name)
565     -- Delimiters (i.e. "(<>[]{}/%") and control characters (0x00-0x1f, and 0x7f),
566     -- space (" ", 0x20) as well as number sign ("#", 0x23) characters have to be escaped
567     return "/" .. name:gsub("[%(<>[%[]{}/%%#]", name_escape)
568 end
569 optex.pdf_name = pdf_name
570 define_lua_command("_pdfname", function()
571     tex_print(-2, pdf_name(token.scan_string()))
572 end)
```

`\pdfstring{⟨string⟩}` serializes a PDF string to bytes.

optex.lua

```
575 local function pdf_string(str)
576     local out = {"<FEFF"}
577     for _, c in utf8.codes(str) do
578         if c < 0x10000 then
579             out[#out+1] = fmt("%04X", c)
580         else
581             c = c - 0x10000
582             local high = bit32.rshift(c, 10) + 0xD800
583             local low = bit32.band(c, 0x3FF) + 0xDC00
584             out[#out+1] = fmt("%04X%04X", high, low)
585         end
586     end
587     out[#out+1] = ">"
588     return table.concat(out)
589 end
590 optex.pdf_string = pdf_string
591 define_lua_command("_pdfstring", function()
592     tex_print(-2, pdf_string(token.scan_string()))
593 end)
```

optex.lua

```
595 local function pdf_ref(id)
596     return fmt("%d 0 R", id)
597 end
598 optex.pdf_ref = pdf_ref
```

optex.lua

```
600 local pdfdict_mt = {
601     __tostring = function(dict)
602         local out = {"<<"}
603         for k, v in pairs(dict) do
604             out[#out+1] = fmt("%s %s", pdf_name(k), tostring(v))
605         end
606         out[#out+1] = ">>"
607         return table.concat(out, "\n")
608     end,
609 }
610 local function pdf_dict(t)
611     return setmetatable(t or {}, pdfdict_mt)
612 end
613 optex.pdf_dict = pdf_dict
```

optex.lua

```
615 tex.print ("\\_public \\pdfname \\pdfstring ;")
```

<sup>3</sup> [https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000\\_2008.pdf](https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000_2008.pdf)

### 2.39.5 Management of PDF page resources

Traditionally, pdfTeX allowed managing PDF page resources (graphics states, patterns, shadings, etc.) using a single toks register, `\pdfpageresources`. This is insufficient due to the expected PDF object structure and also because many “packages” want to add page resources and thus fight for the access to that register. We add a finer alternative, which allows adding different kinds of resources to a global page resources dictionary. Note that some resource types (fonts and XObjects) are already managed by LuaTeX and shouldn’t be added!

XObject forms can also use resources, but there are several ways to make LuaTeX reference resources from forms. It is hence left up to the user to insert page resources managed by us, if they need them. For that, use `pdf.get_page_resources()`, or the below TeX alternative for that.

optex.lua

```
634 local resource_dict_objects = {}
635 local page_resources = {}
636 function pdf.add_page_resource(type, name, value)
637     local resources = page_resources[type]
638     if not resources then
639         local obj = pdf.reserveobj()
640         pdf.setpageresources(fmt("%s %s %s", pdf.get_page_resources(), pdf_name(type), pdf_ref(obj)))
641         resource_dict_objects[type] = obj
642         resources = pdf_dict()
643         page_resources[type] = resources
644     end
645     page_resources[type][name] = value
646 end
647 function pdf.get_page_resources()
648     return pdf.getpageresources() or ""
649 end
```

New “pseudo” primitives are introduced. `\_addpageresource{<type>}{<PDF name>}{<PDF dict>}` adds more resources of given resource `<type>` to our data structure. `\_pageresources` expands to the saved `<type>`s and object numbers.

optex.lua

```
655 define_lua_command("_addpageresource", function()
656     pdf.add_page_resource(token.scan_string(), token.scan_string(), token.scan_string())
657 end)
658 define_lua_command("_pageresources", function()
659     tex_print(pdf.get_page_resources())
660 end)
```

We write the objects with resources to the PDF file in the `finish_pdffile` callback.

optex.lua

```
664 callback.add_to_callback("finish_pdffile", function()
665     for type, dict in pairs(page_resources) do
666         local obj = resource_dict_objects[type]
667         pdf.immediateobj(obj, tostring(dict))
668     end
669 end, "_pageresources")
```

### 2.39.6 Handling of colors and transparency using attributes

Because LuaTeX doesn’t do anything with attributes, we have to add meaning to them. We do this by intercepting TeX just before it ships out a page and inject PDF literals according to attributes.

The attribute for coloring is allocated in `colors.opm`

optex.lua

```
678 local color_attribute = registernumber("_colorattr")
679 local transp_attribute = registernumber("_transpattr")
```

Now we define function which creates whatsit nodes with PDF literals. We do this by creating a base literal, which we then copy and customize.

optex.lua

```
684 local pdf_base_literal = direct.new("whatsit", "pdf_literal")
685 setfield(pdf_base_literal, "mode", 2) -- direct mode
686 local function pdfliteral(str)
687     local literal = copy(pdf_base_literal)
688     setfield(literal, "data", str)
689     return literal
690 end
691 optex.directpdfliteral = pdfliteral
```

The function `colorize(head, current, current_stroke, current_tr)` goes through a node list and injects PDF literals according to attributes. Its arguments are the head of the list to be colored and the current color for fills and strokes and the current transparency attribute. It is a recursive function – nested horizontal and vertical lists are handled in the same way. Only the attributes of “content” nodes (glyphs, rules, etc.) matter. Users drawing with PDF literals have to set color themselves.

Whatsit node with color setting PDF literal is injected only when a different color or transparency is needed. Our injection does not care about boxing levels, but this isn’t a problem, since PDF literal whatsits just instruct the `\shipout` related procedures to emit the literal.

We also set the stroke and non-stroke colors separately. This is because stroke color is not always needed – LuaTeX itself only uses it for rules whose one dimension is less than or equal to 1 bp and for fonts whose `mode` is set to 1 (outline) or 2 (outline and fill). Catching these cases is a little bit involved. For example rules are problematic, because at this point their dimensions can still be running ( $-2^{30}$ ) – they may or may not be below the one big point limit. Also the text direction is involved. Because of the negative value for running dimensions the simplistic check, while not fully correct, should produce the right results. We currently don’t check for the font mode at all, and instead conservatively consider glyphs to always need stroke color. But users can set `\directlua{optex.glyphstroked=false}` if they are sure that their document doesn’t include colorized fonts in mode 1 or 2. Then the PDF output can be smaller in its size (when there is huge amount of color switching, like in this document).

Leaders (represented by glue nodes with leader field) are not handled fully. They are problematic, because their content is repeated more times and it would have to be ensured that the coloring would be right even for e.g. leaders that start and end on a different color. We came to conclusion that this is not worth, hence leaders are handled just opaquely and only the attribute of the glue node itself is checked. For setting different colors inside leaders, raw PDF literals have to be used.

We use the `node.direct` way of working with nodes. This is less safe, and certainly not idiomatic Lua, but faster and codewise more close to the way TeX works with nodes.

`optex.lua`

```

734 optex.glyphstroked = true -- default: insert stroke color for glyphs too.
735
736 local function is_color_needed(head, n, id, subtype) -- returns fill, stroke color needed
737     if id == glyph_id then
738         return true, optex.glyphstroked
739     elseif id == glue_id then
740         n = getleader(n)
741         if n then
742             return true, true
743         end
744     elseif id == rule_id then
745         local width, height, depth = getwhd(n)
746         if width <= one_bp or height + depth <= one_bp then
747             -- running ( $-2^{30}$ ) may need both
748             return true, true
749         end
750         return true, false
751     elseif id == whatsit_id and (subtype == pdfliteral_id
752         or subtype == pdfsave_id
753         or subtype == pdfrestore_id) then
754         return true, true
755     end
756     return false, false
757 end
758
759 local function colorize(head, current, current_stroke, current_tr)
760     for n, id, subtype in traverse(head) do
761         if id == hlist_id or id == vlist_id then
762             -- nested list, just recurse
763             local list = getlist(n)
764             list, current, current_stroke, current_tr =
765                 colorize(list, current, current_stroke, current_tr)
766             setlist(n, list)
767         elseif id == disc_id then
768             -- at this point only no-break (replace) list is of any interest
769             local replace = getfield(n, "replace")
770             if replace then
771                 replace, current, current_stroke, current_tr =
772                     colorize(replace, current, current_stroke, current_tr)

```

```

773         setfield(n, "replace", replace)
774     end
775 else
776     local fill_needed, stroke_needed = is_color_needed(head, n, id, subtype)
777     local new = getattribute(n, color_attribute) or 0
778     local newtr = getattribute(n, transp_attribute) or 0
779     local newliteral = nil
780     if current ~= new and fill_needed then
781         newliteral = token_getmacro("_color:".new)
782         current = new
783     end
784     if current_stroke ~= new and stroke_needed then
785         local stroke_color = token_getmacro("_color-s:".current)
786         if stroke_color then
787             if newliteral then
788                 newliteral = fmt("%s %s", newliteral, stroke_color)
789             else
790                 newliteral = stroke_color
791             end
792             current_stroke = new
793         end
794     end
795     if newtr ~= current_tr and fill_needed then -- (fill_ or stroke_needed) = fill_needed
796         if newliteral ~= nil then
797             newliteral = fmt("%s /tr%d gs", newliteral, newtr)
798         else
799             newliteral = fmt("/tr%d gs", newtr)
800         end
801         current_tr = newtr
802     end
803     if newliteral then
804         head = insertbefore(head, n, pdfliteral(newliteral))
805     end
806 end
807 end
808 return head, current, current_stroke, current_tr
809 end

```

Colorization should be run just before shipout. We use our custom callback for this. See the definition of `pre_shipout_filter` for details on limitations.

```

814 callback.add_to_callback("pre_shipout_filter", function(list)
815     -- By setting initial color to -1 we force initial setting of color on
816     -- every page. This is useful for transparently supporting other default
817     -- colors than black (although it has a price for each normal document).
818     local list = colorize(todirect(list), -1, -1, 0)
819     return tonode(list)
820 end, "_colors")

```

optex.lua

We also hook into luaotfload's handling of color and transparency. Instead of the default behavior (inserting colorstack whatsits) we set our own attribute. On top of that, we take care of transparency resources ourselves.

The hook has to be registered *after* luaotfload is loaded.

```

827 local color_count = registernumber("_colorcnt")
828 local function set_node_color(n, color) -- "1 0 0 rg" or "0 g", etc.
829     local attr = tonumber(token_getmacro("_color:".color))
830     if not attr then
831         attr = tex_getcount(color_count)
832         tex_setcount(color_count, attr + 1)
833         local strattr = tostring(attr)
834         token_setmacro("_color:".color, strattr, "global")
835         token_setmacro("_color:".strattr, color, "global")
836         token_setmacro("_color-s:".strattr, string.upper(color), "global")
837     end
838     setattribute(todirect(n), color_attribute, attr)
839 end
840 optex.set_node_color = set_node_color

```

optex.lua

```

842 function optex.hook_into_luaotfload()

```

```

843 -- color support for luaotfload v3.13+, otherwise broken
844 pcall(luaotfload.set_colorhandler, function(head, n, rgbcolor) -- rgbcolor = "1 0 0 rg"
845     set_node_color(n, rgbcolor)
846     return head, n
847 end)
848
849 -- transparency support for luaotfload v3.22+, otherwise broken
850 pcall(function()
851     luatexbase.add_to_callback("luaotfload.parse_transparent", function(input) -- from "00" to "FF"
852         -- in luaotfload: 0 = transparent, 255 = opaque
853         -- in optex:      0 = opaque,      255 = transparent
854         local alpha = tonumber(input, 16)
855         if not alpha then
856             tex.error("Invalid transparency specification passed to font")
857             return nil
858         elseif alpha == 255 then
859             return nil -- this allows luaotfload to skip calling us for opaque style
860         end
861         local transp = 255 - alpha
862         local transpv = fmt("%.3f", alpha / 255)
863         pdf.add_page_resource("ExtGState", fmt("tr%d", transp), pdf_dict{ca = transpv, CA = transpv})
864         pdf.add_page_resource("ExtGState", "tr0", pdf_dict{ca = 1, CA = 1})
865         return transp -- will be passed to the below function
866     end, "optex")
867
868     luaotfload.set_transparenthandler(function(head, n, transp)
869         setattribute(n, transp_attribute, transp)
870         return head, n
871     end)
872 end)
873 end
874

```

`\_beglocalcontrol`  $\langle tokens \rangle$  `\_endlocalcontrol` runs  $\langle tokens \rangle$  fully at expand processor level despite the fact that  $\langle tokens \rangle$  processes unexpandable commands.

optex.lua

```

877
878 define_lua_command("_beglocalcontrol", function()
879     return tex.runtoks(token.get_next)
880 end)
881
882 -- History:
883 -- 2025-11-25 pre_append_to_vlist_filter callback added
884 -- 2025-05-12 optex.glyphstoked introduced
885 -- 2025-05-11 if not \vbox then raw_ht returns zero instead error
886 -- 2024-12-18 \pdfstring etc. introduced
887 -- 2024-09-06 raw_ht() implemented
888 -- 2024-06-02 more checking in add_to_callback and remove_from_callback
889 -- 2024-02-18 \_beglocalcontrol added
890 -- 2022-08-25 expose some useful functions in `optex` namespace
891 -- 2022-08-24 luaotfload transparency with attributes added
892 -- 2022-03-07 transparency in the colorize() function, current_tr added
893 -- 2022-03-05 resources management added
894 -- 2021-07-16 support for colors via attributes added
895 -- 2020-11-11 optex.lua released

```

## 2.40 Printing documentation

The `\printdoc`  $\langle filename \rangle \langle space \rangle$  and `\printdoctail`  $\langle filename \rangle \langle space \rangle$  commands are defined after the file `doc.opm` is load by `\load` [doc].

The `\printdoc` starts reading of given  $\langle filename \rangle$  from the second line. The file is read in *the listing mode*. The `\printdoctail` starts reading given  $\langle filename \rangle$  from the first occurrence of the `\_endcode`. The file is read in normal mode (like `\input`  $\langle filename \rangle$ ).

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 or one tab character 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 main documentation point can document a control sequence (use `\`\sequence``) or a normal sequence without backslash (use `\`\text``). It is printed in red. Examples: `\`\foo`` (control sequence), `\`\foo`` (normal sequence). The user documentation point is the first occurrence of `\`\sequence`` or `\`\text``. There can be more such markups, all of them are hyperlinks to the relevant main documentation point. And main documentation point is a hyperlink to the relevant user documentation point if this point precedes. Finally, the `\`\sequence`` (for example `\`\foo``) are hyperlinks to the user documentation point.

By default, the hyperlink from main documentation point to the user documentation point is active only if it is backward link, i.e. the main documentation point is given later. The reason is that we don't know if such user documentation point will exist when creating main documentation point and we don't want broken links. If you are sure that user documentation point will follow then use prefix `\fw` before `\``, for example `\fw\`\foo`` is main documentation point where the user documentation point is given later and forward hyperlink is created here.

Documented sequences and their page positions of main documentation points and user documentation points are saved to the index.

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. Moreover, active links are control sequences of the type `\_foo` or `\.foo` although the documentation mentions only `\foo`. Another text is printed in black.

The listing mode is able to generate external links to another OpTeX-like documentation, if the macro `\el:<csname>` is defined. The macro should create a hyperlink using `\_tmpa` where the link name of the `<csname>` is saved and `\_tmpb` where the name of the `<csname>` to be printed is saved (`\_tmpb` is without backslash but it can include preceding `_` or `.` unlike `\_tmpa`). For example, suppose, that we have created `optex-doc.eref` file by:

```
TEXINPUTS='.;$TEXMF/{doc,tex}//' optex optex-doc
grep Xindex optex-doc.ref > optex-doc.eref
```

The `.eref` file includes only `\_Xindex{;<csname>}}{}` lines from `optex-doc.ref` file. Then we can use following macros:

```
\def\_Xindex#1#2{\slet{el:\ignoreit#1}{optexdoclink}}
\def\optexdoclink{%
  \edef\extlink{\optexdocurl\csstring\#cs:\_tmpa}%
  \xlink{url}{\extlink}{\Cyan}{\csstring\\_tmpb}}
\def\optexdocurl{http://petr.olsak.net/ftp/olsak/optex/optex-doc.pdf}
\isfile{optex-doc.eref}\iftrue \input{optex-doc.eref}\fi
```

All `\el:<csname>`, where `<csname>` is from `optex-doc.ref`, have the same meaning: `\optexdoclink` in this example. And `\optexdoclink` creates the external link in `\Cyan` color.

## 2.40.1 Implementation

```
3 \_codedecl \printdoc {Macros for documentation printing <2025-11-07>} % loaded on demand by \load[doc]
```

General declarations.



```

9 \_fontfam[lmfonts]
10
11 \_let \mlinkcolor=\Red % main doc. points
12 \_let \ulinkcolor=\Blue % user doc. points
13 \_let \fnamecolor=\Brown % file names in listing headers
14 \_def \bgverbcolor {\_setcmykcolor{0 0 .3 .03}} % background for listings
15 \_def \outlinkcolor {\_setcmykcolor{1 0 1 .2}} % green for outerlinks
16 \_def \inlinkcolor {\_setcmykcolor{0 1 0 .1}} % magenta for internal links
17 \_hyperlinks \inlinkcolor \outlinkcolor
18 \_enlang
19 \_enquotes

```

Maybe, somebody needs \seccc or \secccc?

```

25 \_eoldef\seccc#1{\_medskip \_noindent{\_bf#1}\_par\_nobreak\_firstnoindent}
26 \_def\secccc{\_medskip \_noindent $\_bullet$ }

```

\enddocument can be redefined.

```

32 \_let\enddocument=\_bye

```

A full page of listing causes underfull \vbox in output routine. We need to add a small tolerance.

```

39 \_pgbottomskip=0pt plus10pt minus2pt

```

The listing mode is implemented here. The \maxlines is maximal lines of code printed in the listing mode.

```

46 \_newcount \_maxlines \_maxlines=100000
47 \_public \_maxlines ;
48
49 \_eoldef\_cod#1{\_par \_wipeepar
50 \_vskip\_parskip \_medskip \_ttskip
51 \_begingroup
52 \_typo[8/10]
53 \_let\_printverblinenum=\_printcodeline
54 \_ttline=\_inputlineno
55 \_setverb
56 \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
57 \_adef{ }{\ }\_adef{^~I{t}}\_parindent=\_ttindent \_parskip=0pt
58 \_def{t}{\_hskip \_dimexpr\_tabspace em/2\_relax}%
59 \_relax \_ttfont
60 \_endlinechar=^~J
61 \_def\_tmpb{\_start}%
62 \_readverblinenum
63 }
64 \_def\_readverblinenum #1^~J{%
65 \_def\_tmpa{\_empty#1}%
66 \_let\_next=\_readverblinenum
67 \_ea\_isinlist\_ea\_tmpa\_ea{\_Doc}\_iftrue \_let\_next=\_processinput \_fi
68 \_ea\_isinlist\_ea\_tmpa\_ea{\_Doctab}\_iftrue \_let\_next=\_processinput \_fi
69 \_ea\_isinlist\_ea\_tmpa\_ea{\_Endcode}\_iftrue \_def\_next{\_processinput\_endinput}\_fi
70 \_ifx\_next\_readverblinenum \_addto\_tmpb{#1^~J}\_fi
71 \_next
72 }
73 {\_catcode\_ =13 \_gdef\_aspace{ }}\_def\_asp{\_ea\_noexpand\_aspace}
74 \_edef\_Doc{\_asp\_asp\_bslash\_doc}
75 \_bgroup \_lccode^~=\_^~I \_lowercase{\_egroup\_edef\_Doctab{\_noexpand\_bslash\_doc}}
76 \_edef\_Endcode{\_noexpand\_empty\_bslash\_endcode}

```

The scanner of the control sequences in the listing mode replaces all occurrences of \ by \\_makes. This macro reads next tokens and accumulates them to \\_tmpa as long as they have category 11. It means that \\_tmpa includes the name of the following control sequence when \\_makes is run. The printing form of the control sequence is set to \\_tmpb and the test of existence \,;\langle csname\rangle is performed. If it is true then active hyperlink is created. If not, then the first \_ or . is removed from \\_tmpa and the test is repeated.

```

89 \def\makecs{\def\tmpa{\futurelet\next\makecsD}
90 \def\makecsD{\if.\next \ea\makecsB \else \ea\makecsA \fi} % \foo is accepted
91 \def\makecsA{\ifcat a\noexpand\next \ea\makecsB \else \ea\makecsF \fi}
92 \def\makecsB#1{\_addto\tmpa{#1}\futurelet\next\makecsA}
93 \def\makecsF{\_let\tmpb=\tmpa
94   \_ifx\tmpa\_empty \_bslash\_returnfi \_fi
95   \_ifcsname el:\tmpa\_endcsname \csname el:\tmpa\_endcsname \_returnfi \_fi
96   \_ifcsname ,;\tmpa\_endcsname \_intlink \_returnfi \_fi
97   \_remfirstunderscoreordot\tmpa
98   \_ifcsname el:\tmpa\_endcsname \csname el:\tmpa\_endcsname \_returnfi \_fi
99   \_ifcsname ,;\tmpa\_endcsname \_intlink \_returnfi \_fi
100  \csstring\\\_tmpb \_relax
101 }
102 \def\_processinput{%
103   \_let\_start=\_relax
104   \_ea\_replstring\_ea\tmpb\_ea{\_aspace^^J}{^^J}
105   \_addto\tmpb{\_fin}%
106   \_isinlist\tmpb{\_start^^J}\_iftrue \_advance\_ttline by1\_fi
107   \_replstring\tmpb{\_start^^J}{\_start}%
108   \_replstring\tmpb{\_start}{}%
109   \_replstring\tmpb{^^J\_fin}{\_fin}%
110   \_replstring\tmpb{^^J\_fin}{}%
111   \_replstring\tmpb{\_fin}{}%
112   \_ea\_prepareverbdata\_ea\tmpb\_ea{\_tmpb^^J}%
113   \_replthis{\_csstring\\}{\_noexpand\_makecs}%
114   \_ea\_printverb \tmpb\_fin
115   \_par
116   \_endgroup \_ttskip
117   \_isnextchar\_par{}{\_noindent}%
118 }
119 \def\_remfirstunderscoreordot#1{\_ea\_remfirstuordotA#1\_relax#1}
120 \def\_remfirstuordotA#1#2\_relax#3{\_if\_#1\_def#3{#2}\_fi \_if\_string#1.\_def#3{#2}\_fi}

```

By default the internal link is created by `\_intlink` inside listing mode. But you can define `\el:<csname>` which has precedence and it can create an external link. The `\tmpa` includes the name used in the link and `\tmpb` is the name to be printed. See `\_makecsF` above and the example at the beginning of this section.

```

130 \def\_intlink{\_link[cs:\tmpa]{\_ulinkcolor}{\_csstring\\\_tmpb}}

```

The lines in the listing mode have a yellow background.

```

136 \def\_printcodeline#1{\_advance \_maxlines by-1
137   \_ifnum \_maxlines<0 \ea \_endverbprinting \_fi
138   \_ifx\_printfilename\_relax \_penalty \_tptenalty \_fi \_vskip-4pt
139   \_noindent\_rlap{\bgverbcolor \_vrule height8pt depth5pt width\_hsize}%
140   \_printfilename
141   \_indent \_printverblinenum #1\_par}
142
143 \def\_printfilename{\_hbox to0pt{%
144   \_hskip\_hsize\_vbox to0pt{\_vss\_llap{\fnamecolor\docfile}\_kern7.5pt}\_hss}%
145   \_let\_printfilename=\_relax
146 }
147 \_let\_normalprintfilename=\_printfilename
148 \_addto\_printcomments{\_let\_printfilename=\_normalprintfilename}
149 \_everytt={\_let\_printverblinenum=\_relax}
150
151 \_long\_def\_endverbprinting#1\_fin#2\_fin{\_fi\_fi \_global\_maxlines=100000
152   \_noindent\_typosize[8/]\_dots etc. (see {\_tt\fnamecolor\docfile})}

```

`\docfile` is currently documented file.

`\printdoc` and `\printdoctail` macros are defined here.

```

159 \def\docfile{}
160 \def\_printdoc #1 {\_par \_def\docfile{#1}%
161   \_everytt={\_ttshift=-15pt \_let\_printverblinenum=\_relax}%
162   \_ea\_cod \_input #1
163   \_everytt={\_let\_printverblinenum=\_relax}%
164   \_def\docfile{}%

```

```

165 }
166 \def\printdoctail #1 {\_bgroup
167   \everytt={}\_ttline=-1 \_ea\_printdoctailA \_input #1 \_egroup}
168 {\\_long\_gdef\_printdoctailA#1\_endcode{}}
169
170 \_public \printdoc \printdoctail ;

```

You can do `\verbininput \vitt{⟨filename⟩} (⟨from⟩-⟨to⟩) ⟨filename⟩` if you need analogical design like in listing mode.

```

177 \def\vitt#1{\_def\docfile{#1}\_ttline=-1 \_everytt={}%
178   \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
179   \_let\_printverblinenum=\_normalprintverblinenum
180   \_typosize[8/10]\_let\_printverblinenum=\_printcodeline \_medskip
181 }
182 \_let\_normalprintverblinenum=\_printverblinenum
183
184 \_public \vitt ;

```

doc.opm

The Index entries of control sequences are with semicolon instead backslash in `.ref` file. It is ignored when sorting but doesn't cause problems in the `.ref` file. The Index entries of normal sequences are finished by `|` because these entries should be sorted just after control sequences with the same name. When printing Index, we first try to refer to the main documentation point (the `\cs:name` or `\ns:name` exists). Index entries with only user documentation points refers to this point. Other index entries are printed as usual without backslash.

```

197 \addto \_ignoredchars {_} % \foo, \_foo is the same in the first pass of sorting
198 \_let\_optexprintii=\_printii
199 \_def\_printii #1#2&{%
200   \_if ;#1\_def\_tmp{#2}\_else \_printiiB #1#2\_end\_end\_relax{#1#2}\_fi % does \def\_tmp{#1#2}
201   \_ea\_definefirstii \_tmp&%
202   \_ifx\_firstii\_lastii\_else
203     \_ea\_newiiletter\_ea{\_firstii}{\_tmp}\_let\_lastii=\_firstii\_fi
204   \_let\_currii=\_tmp \_the\_everyii \_noindent
205   \_hskip-\_iindent \_ignorespaces
206   \_if ;#1\_printiiC c\_backslash{#1#2}\_else \_printiiC n\_empty{#1#2}\_fi
207 }
208 \_def\_printiiB#1|\_end#2\_relax#3{\_ifx^#2\_def\_tmp{#3}\_else\_def\_tmp{#1}\_fi}
209 \_def\_printiiC #1#2#3{% #1: c or n, #2: \_backslash or \_empty, \_tmp: name, #3: (raw name)
210   \_printiiD #1#2{#3}% maybe other index types
211   \_ifcsname #1s:\_tmp\_endcsname {\_tt\_link[#1s:\_tmp]\_ulinkcolor{#2\_tmp}}\_returnfi\_fi
212   \_ifcsname #1s:~\_tmp\_endcsname {\_tt\_link[#1s:~\_tmp]\_ulinkcolor{#2\_tmp}}\_returnfi\_fi
213   \_isnextchar<{\_ea\_printiiA\_ignoreit}%
214   {\_isnextchar>{\_ea\_printiiA\_ignoreit}{\_printiiA}}#3//% other index types
215   \_relax \_space
216 }
217 \_def\_printiiD #1#2#3{} % macro programmer can redefine it and declare another index types
218 \_def\_pgprintA #1{#1} % no hyperlinks from page numbers
219
220 \_def\_printiiPages#1&{\_let\_pgtype=\_undefined \_tmpnum=0
221   {\_rm\_printpages #1,.,\_par}}
222
223 \_sdef\_tocl:1{#1#2#3{\_nofirst\_bigskip
224   \_bf\_llaptoclink{#1}{#2}\_hfill \_pgn{#3}\_tocpar\_medskip}

```

doc.opm

If this macro is loaded by `\load` then we need to initialize catcodes using the `\_afterload` macro.

```

231 \_def\_afterload{\_catcode`<=13 \_catcode`^`=13
232   \_wlog {doc.opm: catcodes of < and ^ activated.}%
233 }

```

doc.opm

The `<something>` will be print as `⟨something⟩`.

doc.opm

```

239 \_let\_lt=<
240 \_catcode`<=13
241
242 \_def<#1>{\_langle\_hbox{\it#1/}\rangle$}
243 \_everyintt{\_catcode`<=13 \_catcode`^`=11 }

```

Main documentation point to a control sequence is created by `\`foo`` and its link name is `cs:foo`. Main documentation point to another sequence (without backslash) is created by `\`foo`` and its link name is `ns:foo`. User documentation point to a control sequence is created by `\`~`foo`` and its link name is `cs:~foo`. User documentation point to another sequence (without backslash) is created by `\`~`foo`` and its link name is `ns:~foo`. Only the first occurrence of the user documentation point has its link destination. User documentation point are linked to relevant main documentation points. When the link destination is created then the control sequence with the same name is defined as an empty macro. Users can create a link to the user documentation point by `\`~`foo`.

doc.opm

```

261 \_def\_docrefcodes{\_catcode`\.=11\_relax}
262
263 \_verbchar`
264
265 \_def\`{\_bgroup \_docrefcodes \_mainpoint}
266 \_def\_mainpoint #1`{\_egroup\_leavevmode\_edef\_tmp{\_csstring#1}%
267   \_ea \_mainpointA \_string#1\_relax % \defines \_cn as c or n
268   \_iindex{\_if\_cn c;\_tmp\_else\_tmp}\_fi}%
269   \_ifcsname \_cn s:\_tmp\_endcsname \_moremainpoints \_else \_dest[\_cn s:\_tmp]\_fi
270   \_sxdef{\_cn s:\_tmp}\_fi}%
271   \_hbox{\_ifcsname \_cn s:\_tmp\_endcsname
272     \_link[\_cn s:\_tmp]{\mlinkcolor}{\_tt\_if\_cn c\_bslash\_fi \_tmp}\_else
273     {\_tt \mlinkcolor \_if\_cn c\_bslash\_fi \_tmp}\_fi}%
274 }
275 \_def\_mainpointA #1#2\_relax {\_edef\_cn{\_if\_csstring\`#1c\_else n\_fi}}
276 \_def\_moremainpoints{\_opwarning{Second main documentation point \_if\_cn c\_bslash\_fi\_tmp}}
277
278 \_def\`~{\_bgroup \_docrefcodes \_docpoint}
279 \_def\_docpoint #1`{\_egroup\_leavevmode\_edef\_tmp{\_csstring#1}%
280   \_ea \_mainpointA \_string#1\_relax % \defines \_cn as c or n
281   \_if\_cn c\_iindex{\_tmp}%
282   \_hbox{\_ifcsname cs:\_tmp\_endcsname
283     \_else \_dest[cs:\_tmp]\_sxdef{cs:\_tmp}\_fi
284     \_link[cs:\_tmp]{\ulinkcolor}{\_tt\_string#1}}%
285   \_afterfi{\_futurelet\_next\_cslinkA}%
286   \_else \_afterfi {\_docpointN #1}\_fi
287 }
288 \_def\_cslinkA{\_ifx\_next\_ea\_ignoreit \_else \_ea\_ea\_ea\_ea\_string\_fi}
289 \_def\_docpointN #1`{\_iindex{#1}}%
290   \_hbox{\_ifcsname ns:~#1\_endcsname \_else \_dest[ns:~#1]\_sxdef{ns:~#1}\_fi
291   \_link[ns:~#1]{\ulinkcolor}{\_tt#1}}%
292 }
293 \_def\`~{\_bgroup \_docrefcodes \_doctpoint}
294 \_def\_doctpoint #1`{\_egroup\_leavevmode\_edef\_tmp{\_csstring#1}%\_iindex{\_tmp}%
295   \_ea \_mainpointA \_string#1\_relax % \defines \_cn as c or n
296   \_if\_cn c\_iindex{\_tmp}%
297   \_hbox{\_link[cs:\_tmp]{\ulinkcolor}{\_tt\_string#1}}%
298   \_afterfi{\_futurelet\_next\_cslinkA}%
299   \_else \_afterfi{\_docpointNA #1}\_fi
300 }
301 \_def\_docpointNA #1`{\_iindex{#1}}\_hbox{\_link[ns:~#1]{\ulinkcolor}{\_tt#1}}

```

The `\fw` macro for forward links to user documentation point (given later) is defined here.

doc.opm

```

309 \_def\_fw\`#1`{\_slet{cs:\_csstring#1}{\_fw}\_slet{ns:\_csstring#1}{\_fw}\`#1`}
310 \_public \fw ;

```

# Index

Control sequences declared by OpTeX have page list here and they are internal links to their main documentation point. TeX primitives used by OpTeX have no page list here and they are external links to [TeX in a Nutshell](#) to the place where the primitive is briefly described.

<code>\_AbbreviateFirstname</code> 176	<code>\_balancecolumns</code> 164	<code>\biboptions</code> 53, 178
<code>\abovedisplayshortskip</code> *	<code>\_banner</code> 41	<code>\_bibp</code> 165
<code>\abovedisplayskip</code> *	<code>\basefilename</code> 29, 37	<code>\bibpart</code> 21, 53, 165, 171
<code>\_aboveliskip</code> 141	<code>\baselineskip</code> *	<code>\_bibskip</code> 168
<code>\_abovetitle</code> 136, 138	<code>\bbchar</code> 88, 107	<code>\bibtexhook</code> 53, 169
<code>\accent</code> *	<code>\_bbdigits</code> 105	<code>\_bibwarning</code> 170, 176
<code>\active</code> 62	<code>\bbig</code> 94	<code>\big</code> 94
<code>\activequotes</code> 205	<code>\bBig</code> 94	<code>\Big</code> 94
<code>\_addcitelist</code> 166–167	<code>\bbigl</code> 94	<code>\bigbreak</code> 63
<code>\_addcolor</code> 124	<code>\bBigl</code> 94	<code>\bigg</code> 94
<code>\addextgstate</code> 122, 155	<code>\bbigm</code> 94	<code>\Bigg</code> 94
<code>\_additcorr</code> 115	<code>\bBigm</code> 94	<code>\biggl</code> 94
<code>\_addpageresource</code> 155, 223	<code>\bbigr</code> 94	<code>\Biggl</code> 94
<code>\address</code> 25, 198	<code>\bBigr</code> 94	<code>\biggm</code> 94
<code>\_addtabitemx</code> 160	<code>\_bbvariables</code> 105	<code>\Biggm</code> 94
<code>\addto</code> 29, 40, 116	<code>\_bcalvariables</code> 105	<code>\biggr</code> 94
<code>\_addtomodlist</code> 84	<code>\begblock</code> 14, 27–28, 142	<code>\Biggr</code> 94
<code>\addUmathfont</code> 101, 103	<code>\_begeqanno</code> 98	<code>\bigl</code> 94
<code>\adef</code> 17, 29, 40	<code>\begfile</code> 27	<code>\Bigl</code> 94
<code>\adjdemerits</code> *	<code>\begingroup</code> *	<code>\bigm</code> 94
<code>\adots</code> 95	<code>\begitems</code> 13–14, 28, 52,	<code>\Bigm</code> 94
<code>\advance</code> *	141–142	<code>\bigr</code> 94
<code>\advancepageno</code> 116–117	<code>\_beglocalcontrol</code> 226	<code>\Bigr</code> 94
<code>\_After</code> 176	<code>\beglua</code> 27	<code>\_bigreek</code> 105
<code>\afterassignment</code> *	<code>\begLUA</code> 27	<code>\_biGreek</code> 105
<code>\afterfi</code> 29, 44	<code>\begmulti</code> 19, 28, 53, 163–164	<code>\bigskip</code> 62
<code>\aftergroup</code> *	<code>\_begoutput</code> 116, 133	<code>\bigskipamount</code> 50
<code>\_afteritcorr</code> 115	<code>\begtt</code> 16–18, 28, 51–52, 116,	<code>\_bileters</code> 105
<code>\_afterload</code> 56	144, 146	<code>\binoppenalty</code> *
<code>\afterxfi</code> 29, 44	<code>\_begtti</code> 144	<code>\_bisansgreek</code> 105
<code>\aheadto</code> 29, 40, 103	<code>\belowdisplayshortskip</code> *	<code>\_bisansGreek</code> 105
<code>\algol</code> 27	<code>\belowdisplayskip</code> *	<code>\_bisansvariables</code> 105
<code>\algolcap</code> 27	<code>\_belowliskip</code> 141	<code>\_bivvariables</code> 105
<code>\algolkw</code> 27	<code>\_belowtitle</code> 136, 138	<code>\Black</code> 121
<code>\algolnum</code> 27	<code>\_betweencolumns</code> 163	<code>\Blue</code> 21, 121
<code>\_allocator</code> 42	<code>\bf</code> 8–9, 71, 73, 81, 88, 107	<code>\bmod</code> 96
<code>\allowbreak</code> 63	<code>\_bfdigits</code> 105	<code>\boldify</code> 75, 115
<code>\altquotes</code> 205	<code>\_bfgreek</code> 105, 108	<code>\boldmath</code> 9, 88, 90, 100,
<code>\_asciisortingtrue</code> 192	<code>\_bfGreek</code> 105	102–103, 114
<code>\_athe</code> 142	<code>\_bfletters</code> 105	<code>\_boldunimath</code> 102
<code>\atop</code> *	<code>\_bfrakvariables</code> 105	<code>\bordermatrix</code> 97
<code>\atopwithdelims</code> *	<code>\_bfvariables</code> 105	<code>\_bordermatrixwithdelims</code>
<code>\attributedef</code> *	<code>\bgroup</code> 39	97
<code>\_auscan</code> 173	<code>\bi</code> 8–9, 71, 73, 81, 88, 107	<code>\box</code> *
<code>\_auscanA</code> 173	<code>\bib</code> 20–21, 28, 168	<code>\boxlines</code> 198
<code>\_auscanB</code> 173	<code>\_bibA</code> 168	<code>\boxmaxdepth</code> *
<code>\_auscanD</code> 173	<code>\_bibB</code> 168	<code>\bp</code> 29, 59
<code>\_authlist</code> 185	<code>\_bibentry</code> 172	<code>\_bprinta</code> 170, 175
<code>\_authorname</code> 170, 176	<code>\_bibgl</code> 168	<code>\_bprintb</code> 170, 175–176
<code>\b</code> 65	<code>\bibmark</code> 165, 168, 170	<code>\_bprintc</code> 170, 175
<code>\_backgroundbox</code> 117	<code>\_bibnn</code> 166	<code>\_bprintv</code> 170, 175
<code>\backgroundpic</code> 153	<code>\bibnum</code> 129, 165	<code>\bracedparam</code> 57, 139

<code>\break</code> 63	<code>\closeout</code> *	<code>\cslang</code> 24, 206
<code>\breakafterdirmode</code> *	<code>\clq</code> 210	<code>\csname</code> *
<code>\brokenpenalty</code> *	<code>\clqq</code> 210	<code>\cspplain</code> 203
<code>\Brown</code> 121	<code>\clubpenalty</code> *	<code>\csquotes</code> 25, 205
<code>\_bsansdigits</code> 105	<code>\cmykcolordef</code> 124	<code>\csstring</code> *
<code>\_bsansgreek</code> 105	<code>\_cmyktorgb</code> 123–124	<code>\cstochar</code> 29, 58
<code>\_bsansGreek</code> 105	<code>\cnvinfo</code> 55	<code>\csvread</code> 27
<code>\_bsansvariables</code> 105	<code>\_coc</code> 154	<code>\_ctablelist</code> 56
<code>\bslash</code> 40	<code>\_cod</code> 29, 34–35, 60	<code>\_currfamily</code> 83
<code>\buildrel</code> 96	<code>\code</code> 16–17, 28, 52, 143	<code>\currfile</code> 29, 37
<code>\bye</code> 34, 41, 66	<code>\_codedecl</code> 29, 34–35, 37	<code>\_currpage</code> 59, 127, 131, 210
<code>\_byehook</code> 41, 126	<code>\colnum</code> 160	<code>\currstyle</code> 99
<code>\c</code> 65	<code>\_colorattr</code> 120, 122–123	<code>\_currV</code> 77, 83
<code>\cal</code> 88, 107	<code>\_colorcnt</code> 122	<code>\currvar</code> 8–9, 71–73, 75, 81
<code>callback_functions</code> 216	<code>\_colorcrop</code> 123	<code>\Cyan</code> 21, 121
<code>callback.register</code> 216	<code>\colordef</code> 21–22, 29, 121–123, 125	<code>\d</code> 65
<code>call_callback</code> 217	<code>\_colordefFin</code> 123	<code>\day</code> *
<code>\_calvariables</code> 105	<code>colorize</code> 224	<code>\dbcreate</code> 27
<code>\cancel</code> 27	<code>\_colorprefix</code> 122	<code>\_dbib</code> 168
<code>\caption</code> 10–12, 28, 139–140	<code>\colortab</code> 27	<code>\dbinsert</code> 27
<code>\captionF</code> 27	<code>\colsep</code> 53, 163	<code>\dbitem</code> 27
<code>\_captionformat</code> 139–140	<code>\commentchars</code> 18, 144, 146–147	<code>\dbloop</code> 27
<code>\_captionsep</code> 140	<code>\_commoncolordef</code> 124	<code>\dbshow</code> 27
<code>\captionT</code> 27	<code>\_completepage</code> 116–117	<code>\dbsort</code> 27
<code>\cases</code> 97	<code>\_compoundchars</code> 189–190	<code>\dbwhere</code> 27
<code>\casesof</code> 29, 47–48	<code>\cong</code> 96	<code>\_ddlinedata</code> 160
<code>\catalogexclude</code> 86	<code>\ConTeXt</code> 203	<code>\ddots</code> 95
<code>\catalogmathsample</code> 86	<code>\copy</code> *	<code>\_decddigits</code> 59
<code>\catalognextfam</code> 86	<code>\correctvsize</code> 27	<code>\decr</code> 29, 40
<code>\catalogonly</code> 86	<code>\countdef</code> *	<code>\def</code> *
<code>\catalogsample</code> 86	<code>\cr</code> *	<code>\_defaultfontfeatures</code> 86
<code>\catcode</code> 58	<code>\cramped</code> 99	<code>\defaultitem</code> 14, 52, 142
<code>\catcodetable</code> *	<code>\crampeddisplaystyle</code> *	<code>\defaultoptsize</code> 70
<code>\cdots</code> 95	<code>\crampedscriptscriptstyle</code> *	<code>\defaultskewchar</code> *
<code>\centering</code> 43	<code>\crampedscriptstyle</code> *	<code>\_definefirstii</code> 194
<code>\centerline</code> 63	<code>\crampedtextstyle</code> *	<code>\delang</code> 24, 206
<code>\chap</code> 10, 12, 17–18, 28, 57, 136–137	<code>\crbot</code> 27	<code>\delcode</code> *
<code>\_chapfont</code> 75, 135	<code>\crrc</code> *	<code>\delimiter</code> *
<code>\_chapx</code> 137	<code>\_createbibmark</code> 185	<code>\delimiterfactor</code> *
<code>\char</code> *	<code>\_createcolumns</code> 165	<code>\delimitershortfall</code> *
<code>\chardef</code> *	<code>\_CreateField</code> 176	<code>\dequotes</code> 25, 205
<code>\_checkexists</code> 36	<code>\createfile</code> 27	<code>\dest</code> 13, 129–130
<code>\chyph</code> 24, 210	<code>\crl</code> 15, 159, 162	<code>\_destactive</code> 129
<code>\_circle</code> 153–154	<code>\crli</code> 15, 157, 160, 162	<code>\_destbox</code> 129
<code>\circleparams</code> 55	<code>\crl1</code> 15, 162	<code>\_destboxes</code> 202
<code>\cite</code> 12, 20–21, 28, 166, 169	<code>\crl1i</code> 15, 157, 162	<code>\_destheight</code> 129
<code>\_citeA</code> 166	<code>\crlp</code> 15, 157, 162	<code>\detokenize</code> *
<code>\_citeborder</code> 12, 129	<code>\crm1d</code> 27	<code>\dimen</code> *
<code>\_citeI</code> 167, 174, 177	<code>\crq</code> 210	<code>\dimendef</code> *
<code>\_citelist</code> 171, 174, 177	<code>\crqq</code> 210	<code>\dimexpr</code> *
<code>\_classfam</code> 105	<code>\crtop</code> 27	<code>\directchar</code> 27
<code>\cleaders</code> *	<code>\crx</code> 27	<code>\directlua</code> *
<code>\clipbox</code> 27	<code>\cs</code> 29, 40	<code>\directoutput</code> 27
<code>\clipincircle</code> 24, 156	<code>\CS</code> 203	<code>\dirhbox</code> 100
<code>\clipinoval</code> 24, 156	<code>\cskip</code> 10, 139	<code>\dirvbox</code> 100
<code>\_clipinpath</code> 156		<code>\discretionary</code> *
<code>\closein</code> *		<code>\displaylines</code> 98



<code>\displaystyle *</code>	<code>\enlang</code> 24, 208	<code>\_ffletterspace</code> 86–87
<code>\displaywidowpenalty *</code>	<code>\enquotes</code> 25, 205	<code>\_ffonum</code> 83
<code>\displaywidth *</code>	<code>\enskip</code> 62	<code>\_ffwordspace</code> 86–87
<code>\divide *</code>	<code>\enspace</code> 62	<code>\fi *</code>
<code>\_do</code> 45	<code>\_entrydata</code> 172	<code>\_fieldalias</code> 169, 173
<code>\do</code> 44–45	<code>\eoldef</code> 29, 57, 144	<code>\_fielddata</code> 172
<code>\_dobibsorting</code> 174	<code>\eqalign</code> 54, 97	<code>\_fieldname</code> 172
<code>\dobystyle</code> 99	<code>\eqalignno</code> 10, 98	<code>\filbreak</code> 63
<code>\_doc</code> 29, 34–35, 60	<code>\eqbox</code> 29, 158, 163	<code>\_fin</code> 37
<code>\docfile</code> 229	<code>\eqboxsize</code> 158, 163	<code>\finalhyphendemerits *</code>
<code>\docgen</code> 34, 41, 60	<code>\eqlines</code> 54, 97–98	<code>\_firstii</code> 194
<code>\_doccompound</code> 191	<code>\eqmark</code> 10, 12, 28, 98, 140	<code>\firstmark *</code>
<code>\doloadmath</code> 100–101	<code>\eqno *</code>	<code>\_Firstname</code> 170, 176
<code>\_doshadow</code> 156	<code>\_eqnoleft</code> 98, 100	<code>\_firstnoindent</code> 10, 136, 139
<code>\_dosorting</code> 190, 192	<code>\_eqnoright</code> 98, 100	<code>\fixmnotes</code> 7, 197
<code>\dospecials</code> 62	<code>\eqspace</code> 54, 98	<code>\fL</code> 15, 161
<code>\dosupereject</code> 63, 116	<code>\eqstyle</code> 54, 97–98	<code>\floatingpenalty *</code>
<code>\doteq</code> 96	<code>\ereplstring</code> 27	<code>\flq</code> 210
<code>\dotfill</code> 66	<code>\errmessage *</code>	<code>\flqq</code> 210
<code>\dots</code> 65	<code>\errorcontextlines *</code>	<code>\_fmodbf</code> 81
<code>\doublehyphendemerits *</code>	<code>\escapechar *</code>	<code>\_fmodbi</code> 81
<code>\_douseK</code> 123	<code>\everycapitonf</code> 53	<code>\_fmodit</code> 81
<code>\_doverbininput</code> 145	<code>\everycapitont</code> 53	<code>\_fmodrm</code> 81
<code>\downbracefill</code> 66	<code>\everycr *</code>	<code>\_fmodtt</code> 81–82
<code>\draft</code> 7, 118	<code>\everydisplay *</code>	<code>\fmtname</code> 31
<code>\_dsp</code> 147	<code>\everyeof *</code>	<code>\_fnfborder</code> 13, 130
<code>\dump *</code>	<code>\everyii</code> 53, 193	<code>\fnote</code> 7, 17, 28, 116, 196
<code>\_ea</code> 36	<code>\everyintt</code> 17, 52	<code>\fnotelinks</code> 13, 196
<code>\ea</code> 36, 58	<code>\everyitem</code> 52	<code>\fnotemark</code> 7, 196
<code>\easylist</code> 27	<code>\everyjob *</code>	<code>\fnotenum</code> 196
<code>\ecite</code> 20, 166	<code>\everylist</code> 14, 52	<code>\fnotenumchapters</code> 7, 137,
<code>\edef *</code>	<code>\everymath *</code>	196
<code>\_editorname</code> 170, 176	<code>\everymnote</code> 53	<code>\fnotenumglobal</code> 7, 196
<code>\egroup</code> 39	<code>\everypar *</code>	<code>\fnotenumpages</code> 7, 196, 202
<code>\ehyph</code> 24, 210	<code>\everytable</code> 53, 157	<code>\fnotetext</code> 7, 196
<code>\eject</code> 63	<code>\everytocline</code> 53, 131	<code>\_fnset</code> 142, 197
<code>\else *</code>	<code>\everytt</code> 17–18, 51–52, 144	<code>\_fntborder</code> 13, 130
<code>\em</code> 8, 115	<code>\ewref</code> 126	<code>\folio</code> 26, 117
<code>\emergencystretch *</code>	<code>\exhyphenpenalty *</code>	<code>\font *</code>
<code>\empty</code> 39	<code>\expandafter *</code>	<code>\fontdef</code> 29, 69, 71, 73, 84
<code>\end *</code>	<code>\expanded *</code>	<code>\fontdimen *</code>
<code>\endblock</code> 14, 27–28, 142	<code>\expr</code> 27, 29, 59	<code>\fontfam</code> 5, 7–9, 28, 30, 66,
<code>\_endcode</code> 29, 34–35, 37	<code>\_famalias</code> 81, 86	68–69, 72, 74–76, 80, 82,
<code>\endcsname *</code>	<code>\_famdecl</code> 73, 77–78, 83, 85	85–86, 88, 101
<code>\endfile</code> 27	<code>\_famdepend</code> 84	<code>\fontfamsub</code> 72, 85
<code>\endgraf</code> 62	<code>\_famfrom</code> 81, 86	<code>\_fontfeatures</code> 78, 86
<code>\endgroup *</code>	<code>\_faminfo</code> 81, 86	<code>\fontlet</code> 29, 69–71, 73
<code>\endinput *</code>	<code>\_famsrc</code> 86	<code>\_fontloaded</code> 82
<code>\endinsert</code> 11, 118	<code>\_famsubstitute</code> 85	<code>\fontname *</code>
<code>\enditems</code> 13, 28, 52, 141	<code>\_famtext</code> 81, 86	<code>\_fontnamegen</code> 77, 82–83
<code>\endline</code> 62	<code>\_famv</code> 81	<code>\fontsel</code> 81–82, 84
<code>\endlinechar *</code>	<code>\famvardef</code> 71–75, 77–78,	<code>\_fontselA</code> 82, 87
<code>\endmulti</code> 19, 28, 53, 163–164	81–84	<code>\fontspreload</code> 67
<code>\_endnamespace</code> 29, 35–36	<code>\fC</code> 15, 161	<code>\footins</code> 116, 118, 196
<code>\_endoutput</code> 116	<code>\fcolor</code> 154	<code>\footline</code> 6, 54, 116–117
<code>\_endslides</code> 200	<code>\fcread</code> 27	<code>\footlinedist</code> 6, 54
<code>\endtt</code> 16–18, 28, 51–52, 144,	<code>\_ffadded</code> 86–87	<code>\footnote</code> 7, 116, 118
146	<code>\_ffcolor</code> 86–87	<code>\_footnoterule</code> 116–117

<code>\footstrut</code> 118	<code>\hphantom</code> 96	<code>\inboxdirection</code> 55, 100, 153
<code>\_foreach</code> 44	<code>\hrule</code> *	<code>\_inchap</code> 137
<code>\foreach</code> 29, 44, 104	<code>\hrulefill</code> 66	<code>\incircle</code> 23, 55, 154
<code>\foreachdef</code> 29, 46	<code>\hsize</code> *	<code>\incr</code> 29, 40
<code>\foreachx</code> 45	<code>\hskip</code> *	<code>\incrpp</code> 27
<code>\_forlevel</code> 45	<code>\hss</code> *	<code>\indent</code> *
<code>\fornum</code> 29, 45	<code>\hyperlinks</code> 12–13, 20, 28,	<code>\iniseccc</code> 27
<code>\_fornumB</code> 45	129–130, 136	<code>\_initfontfamily</code> 78, 83
<code>\fornumstep</code> 45	<code>\hyphenation</code> *	<code>\initunifonts</code> 67–70, 82–83
<code>\fR</code> 15, 161	<code>\hyphenchar</code> *	<code>\_inkdefs</code> 151
<code>\frak</code> 88, 107	<code>\hyphenpenalty</code> *	<code>\inkinspic</code> 22, 151
<code>\_frakvariables</code> 105	<code>\ialign</code> 64	<code>\_inmath</code> 107
<code>\frame</code> 16, 23, 163	<code>\if</code> *	<code>\inoval</code> 23, 55, 154
<code>\framedblocks</code> 27	<code>\_ifAleB</code> 191	<code>\input</code> *
<code>\frenchspacing</code> 50	<code>\ifcase</code> *	<code>\inputlineno</code> *
<code>\frq</code> 210	<code>\ifcat</code> *	<code>\_inputref</code> 126
<code>\frqq</code> 210	<code>\ifcsname</code> *	<code>\_insec</code> 137
<code>\frquotes</code> 205	<code>\ifdim</code> *	<code>\_insecc</code> 137
<code>\fS</code> 15, 162	<code>\ifeof</code> *	<code>\_insertmark</code> 139
<code>\_fsetV</code> 78, 83	<code>\_ifexistfam</code> 47, 77	<code>\insertoutline</code> 13, 133
<code>\fullcite</code> 27	<code>\iffalse</code> *	<code>\inspic</code> 22, 27–28, 51, 150–151
<code>\_fullrectangle</code> 142	<code>\ifhmode</code> *	<code>\_inspicA</code> 151
<code>\futurelet</code> *	<code>\ifinner</code> *	<code>\_inspicB</code> 151
<code>\_fvars</code> 77, 83	<code>\_ifmathloading</code> 101	<code>\inspicto</code> 27
<code>\fw</code> 227, 231	<code>\_ifmathsb</code> 91	<code>\interdisplaylinepenalty</code>
<code>\fX</code> 15, 161	<code>\ifmmode</code> *	50
<code>\gdef</code> *	<code>\_ifnonum</code> 137	<code>\interfootnotelinepenalty</code>
<code>\_getfield</code> 174	<code>\_ifnotoc</code> 137	50
<code>\_getforstack</code> 45	<code>\ifnum</code> *	<code>\interlinepenalty</code> *
<code>\_getmathcode</code> 104	<code>\ifodd</code> *	<code>\_intlinc</code> 229
<code>\_gfnotenum</code> 129, 196	<code>\_ifpgfnote</code> 196	<code>\_isAleB</code> 191
<code>\glet</code> *	<code>\iftrue</code> *	<code>\_isansvariables</code> 105
<code>\global</code> *	<code>\ifvmode</code> *	<code>\isdefined</code> 29, 46
<code>\globaldefs</code> *	<code>\ifvoid</code> *	<code>\isempty</code> 29, 46
<code>\goodbreak</code> 63	<code>\ifx</code> *	<code>\isequal</code> 29, 46, 48
<code>\gpageno</code> 116, 129	<code>\_ignoredchars</code> 189–190	<code>\isfile</code> 29, 46–47
<code>\Green</code> 121	<code>\_ignoredcharsgeneric</code> 189	<code>\isfont</code> 29, 46–47
<code>\Grey</code> 121	<code>\ignoreinspic</code> 27	<code>\isinlist</code> 29, 46–47
<code>\halign</code> *	<code>\ignoreit</code> 29, 39, 161	<code>\isinmacro</code> 27
<code>\hang</code> 64	<code>\ignorept</code> 29, 58	<code>\iskv</code> 29, 51, 61
<code>\hangafter</code> *	<code>\ignoresecond</code> 29, 39	<code>\ismacro</code> 29, 46
<code>\hangindent</code> *	<code>\ignoreslash</code> 202–203	<code>\ismatch</code> 27
<code>\hbadness</code> *	<code>\ignorespaces</code> *	<code>\isnextchar</code> 29, 47
<code>\hbox</code> *	<code>\ignslash</code> 203	<code>\ispageodd</code> 27
<code>\headline</code> 6, 54, 116–117	<code>\ii</code> 18–19, 28, 195	<code>\istokseempty</code> 29, 46
<code>\headlinedist</code> 6, 54, 117	<code>\iid</code> 18–19, 195	<code>\it</code> 8, 71, 73, 81, 88, 102, 107
<code>\hfil</code> *	<code>\iindent</code> 14, 52	<code>\item</code> 64
<code>\hfill</code> *	<code>\iindex</code> 195	<code>\itemitem</code> 64
<code>\hfuzz</code> *	<code>\iis</code> 19, 195	<code>\itemnum</code> 141
<code>\hglue</code> 62	<code>\iitype</code> 19, 195	<code>\itemskipamount</code> 52, 142
<code>\hhkern</code> 54	<code>\_iitypesaved</code> 195	<code>\_itgreek</code> 105, 108
<code>\hicolor</code> 52, 148, 150	<code>\ilevel</code> 14, 52	<code>\_itGreek</code> 105
<code>\hicolors</code> 52, 148	<code>\ilink</code> 13, 130	<code>\_itletters</code> 102, 105
<code>\_hicomments</code> 147	<code>\ilistskipamount</code> 52, 141	<code>\_itvariables</code> 105
<code>\hideskip</code> 43	<code>\immediate</code> *	<code>\jobname</code> *
<code>\hidewidth</code> 64	<code>\immediateassigned</code> *	<code>\jointrel</code> 94
<code>\hisyntax</code> 18, 144, 147, 150	<code>\immediateassignment</code> *	<code>\jot</code> 50
<code>\hoffset</code> *	<code>\import</code> 27	<code>\jsonread</code> 27

<code>\_Junior</code> 170, 176	<code>\_linkdimens</code> 129	<code>\mathaccent</code> *
<code>\keepstyle</code> 27	<code>\lipsum</code> 26, 211	<code>\mathadef</code> 29, 40
<code>\kern</code> *	<code>\_lipsumdot</code> 211	<code>\mathbin</code> *
<code>\keystroke</code> 27	<code>\_lipsumload</code> 211	<code>\mathbox</code> 10, 55, 100, 103
<code>\kv</code> 29, 51, 60–61	<code>\lipsumtext</code> 211	<code>\mathchar</code> *
<code>\kvdict</code> 51, 61	<code>\_listfamnames</code> 85	<code>\mathchardef</code> *
<code>\_kvscan</code> 61	<code>\_listskipab</code> 141–142	<code>\mathchars</code> 101, 104
<code>\_kvunknown</code> 61	<code>\llap</code> 63	<code>\mathchoice</code> *
<code>\kvx</code> 29, 51, 61	<code>\_llaptoclink</code> 132	<code>\mathclose</code> *
<code>\label</code> 12, 28, 126, 128, 136, 202	<code>\lmfil</code> 54	<code>\mathcode</code> *
<code>\_labelcolor</code> 128	<code>\load</code> 25–26, 28, 34–35, 56, 226, 230	<code>\mathcodes</code> 101, 104
<code>\_langb</code> 204	<code>\loadboldmath</code> 100, 102–103	<code>\matheqdirmode</code> *
<code>\_langdata</code> 210	<code>\loadmath</code> 9, 73, 100–101, 104	<code>\_mathfaminfo</code> 83
<code>\_langdefault</code> 208–209	<code>\_loadmathfamily</code> 89	<code>\mathhexbox</code> 64
<code>\_langinit</code> 206, 208, 210	<code>\loadtrick</code> 212	<code>\mathinexpr</code> 27
<code>\_langinput</code> 208–210	<code>\_loadumathfamily</code> 103	<code>\_mathloadingfalse</code> 100–101
<code>\langlist</code> 24, 206	<code>\localcolor</code> 122	<code>\_mathloadingtrue</code> 101
<code>\_langreset</code> 208	<code>\loggingall</code> 41	<code>\mathop</code> *
<code>\language</code> *	<code>\logginglua</code> 27	<code>\mathopen</code> *
<code>\_langw</code> 24, 204	<code>\long</code> *	<code>\mathord</code> *
<code>\lastbox</code> *	<code>\longtable</code> 27	<code>\mathpalette</code> 96
<code>\_lastlabel</code> 128	<code>\_loop</code> 44	<code>\mathpunct</code> *
<code>\_Lastname</code> 170, 176	<code>\loop</code> 29, 44	<code>\mathrel</code> *
<code>\lastnamedcs</code> *	<code>\_loopauthors</code> 176	<code>\mathsboff</code> 34, 91
<code>\lastpage</code> 26, 210	<code>\lorem</code> 26, 211	<code>\mathsbon</code> 34, 91, 146
<code>\lastpenalty</code> *	<code>\lower</code> *	<code>\mathstrut</code> 96
<code>\_lastreflabel</code> 128	<code>\lowercase</code> *	<code>\mathstyles</code> 29, 99
<code>\lastskip</code> *	<code>\_lrmnote</code> 197	<code>\mathsurround</code> *
<code>\LaTeX</code> 203	<code>\luaescapestring</code> *	<code>\matrix</code> 97
<code>\layernum</code> 199–200	<code>\LuaTeX</code> 202	<code>\_matt</code> 107
<code>\layers</code> 200, 202	<code>\lwidth</code> 154	<code>\maxdepth</code> *
<code>\_layertext</code> 200, 202	<code>\_mabf</code> 107	<code>\maxdimen</code> 43
<code>\lccode</code> *	<code>\_mabi</code> 107	<code>\maxlines</code> 227–228
<code>\lcolor</code> 154	<code>\_mafam</code> 104	<code>\_maybetod</code> 184
<code>\ldots</code> 95	<code>\mag</code> *	<code>\mbisans</code> 107
<code>\leaders</code> *	<code>\Magenta</code> 121	<code>\_mdfive</code> 126
<code>\leavevmode</code> 64	<code>\magnification</code> 66	<code>\meaning</code> *
<code>\left</code> *	<code>\magscale</code> 6, 28, 119	<code>\medbreak</code> 63
<code>\leftarrowfill</code> 66	<code>\magstep</code> 62	<code>\medmuskip</code> *
<code>\lefthyphenmin</code> *	<code>\magstephalf</code> 62	<code>\_medsk</code> 93
<code>\leftline</code> 63	<code>\mainbaselineskip</code> 8, 114	<code>\medskip</code> 62
<code>\_leftofcolumns</code> 163	<code>\mainfosize</code> 8, 114	<code>\medskipamount</code> 50
<code>\leftskip</code> *	<code>\_mait</code> 107	<code>\_mergesort</code> 192
<code>\legalignno</code> 98	<code>\_makecs</code> 228	<code>\message</code> *
<code>\legno</code> *	<code>\_makecsF</code> 228–229	<code>\_mfactor</code> 90, 101, 103
<code>\let</code> *	<code>\_makefootline</code> 117	<code>\_mfam</code> 89
<code>\letfont</code> 211	<code>\_makeheadline</code> 116–117	<code>\_mfontfeatures</code> 103
<code>\letter</code> 25, 28, 198	<code>\makeindex</code> 18–19, 24, 28, 188, 193	<code>\mfontsrule</code> 114
<code>\_lfnotenum</code> 196	<code>\makeLOA</code> 27	<code>\middle</code> *
<code>\LightGrey</code> 121	<code>\makeLOF</code> 27	<code>\midinsert</code> 11, 118
<code>\limits</code> *	<code>\makeLOT</code> 27	<code>\misans</code> 107
<code>\line</code> 63	<code>\maketoc</code> 18, 27–28, 133, 136	<code>\mit</code> 88
<code>\linepenalty</code> *	<code>\margins</code> 5–6, 28, 30, 116, 119	<code>\mkern</code> *
<code>\lineskip</code> *	<code>\mark</code> *	<code>\_mninvert</code> 197
<code>\lineskiplimit</code> *	<code>\_marm</code> 107	<code>\mnote</code> 7, 28, 53, 197
<code>\link</code> 130	<code>\_math</code> 95	<code>\_mnoteA</code> 197
<code>\_linkactions</code> 129		<code>\_mnoteD</code> 197
		<code>\mnoteindent</code> 53

<code>\_mnotesfixed</code> 197	<code>\_noattr</code> 43	<code>\onlyrgb</code> 21–22, 121–122
<code>\mnotesize</code> 7, 53	<code>\nobibwarning</code> 169, 176	<code>\ooalign</code> 65
<code>\mnoteskip</code> 197	<code>\_nobibwarnlist</code> 176	<code>\openin</code> *
<code>\moddef</code> 72–74, 77–78, 83	<code>\nobreak</code> 63	<code>\openout</code> *
<code>\_modlist</code> 84	<code>\nocite</code> 20, 166, 169	<code>\openref</code> 116, 126
<code>\month</code> *	<code>\noexpand</code> *	<code>\openup</code> 98
<code>\_monthw</code> 204	<code>\_nofileext</code> 37	<code>\_opfootnote</code> 117, 196–197
<code>\morecolors</code> 21, 125	<code>\_nofilepath</code> 37	<code>\opinput</code> 29, 55
<code>\moveleft</code> *	<code>\_nofirst</code> 132	<code>\OPmac</code> 203
<code>\moveright</code> *	<code>\_nograd</code> 144	<code>\opt</code> 57, 60
<code>\_mparams</code> 103	<code>\noindent</code> *	<code>\optdef</code> 29, 57, 60, 137
<code>\mskip</code> *	<code>\nointerlineskip</code> 62	<code>\OpTeX</code> 202
<code>\mspan</code> 15–16, 162	<code>\nokvx</code> 29, 61	<code>\optexcatcodes</code> 56
<code>\_mtext</code> 139, 204	<code>\nolanginput</code> 210	<code>optex.glyphstroked</code> 224
<code>\_Mtext</code> 183, 186	<code>\_nold</code> 190	<code>\_optexoutput</code> 115–116
<code>\multiply</code> *	<code>\nolimits</code> *	<code>optex.raw_ht</code> 215
<code>\_multiskip</code> 163	<code>\noloadmath</code> 9, 73, 101	<code>\_optexshipout</code> 116
<code>\multispan</code> 15, 64	<code>\nonfrenchspacing</code> 50, 208	<code>\optexversion</code> 31
<code>\muskip</code> *	<code>\nonscript</code> *	<code>\_optfn</code> 71
<code>\muskipdef</code> *	<code>\nonum</code> 10, 136–137	<code>\_optfontalias</code> 79, 87
<code>\_mv</code> 153	<code>\nonumcitations</code> 20, 28, 166, 185	<code>\_optname</code> 79, 87
<code>\_namecount</code> 170, 176	<code>\nopagenumbers</code> 6, 117	<code>\_optnameA</code> 87
<code>\_NameCount</code> 170, 176	<code>\_noprefix</code> 57	<code>\_optsize</code> 70
<code>\_namespace</code> 29, 34–36	<code>\normalbaselines</code> 50	<code>\opwarning</code> 29, 40
<code>\narrower</code> 64	<code>\normalbaselineskip</code> 50	<code>\_othe</code> 137
<code>\_narrowlastlinecentered</code> 140	<code>\normalbottom</code> 117	<code>\outer</code> *
<code>\nbb</code> 40	<code>\normalcatcodes</code> 56	<code>\outlines</code> 13, 28, 133
<code>\nbpar</code> 136, 138	<code>\normallineskip</code> 50	<code>\_outlinesA</code> 133
<code>\_negationof</code> 111	<code>\normallineskiplimit</code> 50	<code>\_outlinesB</code> 133
<code>\negthinspace</code> 62	<code>\normalmath</code> 9, 88, 90, 100, 102–103, 114	<code>\output</code> *
<code>\newattribute</code> 43	<code>\_normalunimath</code> 102	<code>\outputmode</code> *
<code>\newbox</code> 42	<code>\nospaceafter</code> 29, 47, 57	<code>\outputpenalty</code> *
<code>\newcatcodetable</code> 43	<code>\nospacefuturelet</code> 29, 57, 85	<code>\_oval</code> 153–154
<code>\newcount</code> 29, 42	<code>\nospec</code> 24, 153	<code>\ovalparams</code> 23, 55
<code>\newcurrfontsize</code> 70, 114	<code>\not</code> 99, 111	<code>\over</code> *
<code>\newdimen</code> 29, 42	<code>\notin</code> 96	<code>\overbrace</code> 95
<code>\newfam</code> 42, 103	<code>\notoc</code> 10, 137	<code>\overfullrule</code> *
<code>\_newfontloaded</code> 82	<code>\novspaces</code> 14, 142	<code>\overlapmargin</code> 154
<code>\newif</code> 29, 35, 43	<code>\_nsprivate</code> 35–36	<code>\overleftarrow</code> 95
<code>\_newifi</code> 29, 35, 43	<code>\_nspublic</code> 35–36	<code>\overrightarrow</code> 95
<code>\_newiiletter</code> 193–194	<code>\null</code> 39	<code>\_pagecontents</code> 116–117
<code>\newinsert</code> 42	<code>\nulldelimiterspace</code> *	<code>\_pagedest</code> 116–117, 202
<code>\newlanguage</code> 42, 206	<code>\nullfont</code> *	<code>\pagedirection</code> *
<code>\newlinechar</code> *	<code>\number</code> *	<code>\pageinsert</code> 118
<code>\newmarks</code> 43	<code>\numberedpar</code> 11, 141	<code>\pageno</code> 26, 116–117
<code>\newmuskip</code> 42	<code>\numexpr</code> *	<code>\_pageresources</code> 155, 223
<code>\newpublic</code> 35–36	<code>\_numprint</code> 120	<code>\par</code> *
<code>\newread</code> 42	<code>\obeylines</code> 62	<code>\_paramtabdeclarep</code> 161
<code>\newskip</code> 42	<code>\obeyspaces</code> 62	<code>\parfillskip</code> *
<code>\newtoks</code> 42	<code>\offinterlineskip</code> 62	<code>\parindent</code> *
<code>\newwrite</code> 42	<code>\oldaccents</code> 30, 65, 169	<code>\parskip</code> *
<code>\_nextat</code> 172	<code>\olistskipamount</code> 52, 141	<code>\_partokenset</code> 213
<code>\_nextfield</code> 172	<code>\omit</code> *	<code>\patterns</code> *
<code>\nextpages</code> 55	<code>\onlycmyk</code> 21, 121–122	<code>\pcent</code> 40
<code>\nl</code> 10, 138	<code>\_onlyif</code> 78, 83	<code>\_pdfborder</code> 129
<code>\nnum</code> 59	<code>\onlyifnew</code> 27	<code>\pdfcatalog</code> *
<code>\noalign</code> *		<code>\pdfdest</code> *
		<code>\pdfendlink</code> *

<code>\pdfextension *</code>	<code>\_prepareverbdata</code> 144–145, 150	<code>\quad</code> 62
<code>\pdffeedback *</code>	<code>\_prepcommalist</code> 83	<code>\_quotationmarks</code> 205
<code>\pdfhorigin *</code>	<code>\_prepinverb</code> 135	<code>\quotes</code> 205
<code>\pdfinfo *</code>	<code>\_preplang</code> 204, 206, 208	<code>\quoteschars</code> 205
<code>\pdflastximage *</code>	<code>\_preplangmore</code> 208–209	<code>\radical *</code>
<code>\pdflastxpos *</code>	<code>\_prepoffsets</code> 116	<code>\raggedbottom</code> 117
<code>\pdflastypos *</code>	<code>\_prepsecondpass</code> 191, 193	<code>\raggedright</code> 64
<code>\pdfliteral *</code>	<code>\_preshipout</code> 116, 120, 221	<code>\raise *</code>
<code>\pdfname</code> 222	<code>\_pretolerance *</code>	<code>\ratio</code> 23, 154
<code>\pdfoutline *</code>	<code>\_prevdepth *</code>	<code>\_rawht</code> 165, 215
<code>\pdfpageheight *</code>	<code>\_prevgraph *</code>	<code>\rcite</code> 20, 28, 166
<code>\pdfpagewidth *</code>	<code>\_prevrefhash</code> 126	<code>\read *</code>
<code>\pdfrefximage *</code>	<code>\_prime</code> 93	<code>\readbibs</code> 170–172, 177
<code>\pdfrestore *</code>	<code>\_printbib</code> 168, 170, 175	<code>\readkv</code> 29, 51, 60–61
<code>\pdfrotate</code> 23, 152	<code>\_printcaptionf</code> 140	<code>\_readverb</code> 143
<code>\pdfsave *</code>	<code>\_printcaptiont</code> 140	<code>\rebox</code> 27
<code>\pdfsavepos *</code>	<code>\_printchap</code> 10, 136	<code>\Red</code> 21, 121
<code>\pdfscale</code> 23, 152	<code>\_printcolumns</code> 165	<code>\ref</code> 12, 28, 128, 202
<code>\pdfsetmatrix *</code>	<code>\_printcomments</code> 147	<code>\_refborder</code> 12, 129
<code>\pdfstartlink *</code>	<code>\_printdoc</code> 34, 226, 229	<code>\refdecl</code> 127
<code>\pdfstring</code> 134–135, 222	<code>\_printdoctail</code> 226, 229	<code>\_refdecldata</code> 127
<code>\pdfunidef</code> 133, 135, 205	<code>\_printentry</code> 174–175	<code>\_reftext</code> 128
<code>\pdfvariable *</code>	<code>\_printfnotemark</code> 196	<code>\regmacro</code> 13, 18, 116, 133, 135
<code>\pdfvorigin *</code>	<code>\_printfnotemarkA</code> 196	<code>\_regmark</code> 116, 133
<code>\pdfximage *</code>	<code>\_printfnotemarkB</code> 196	<code>\_regoptsizes</code> 69, 71, 77, 79, 87
<code>\penalty *</code>	<code>\_printii</code> 74, 193	<code>\_regoul</code> 133, 143
<code>\pend</code> 27	<code>\_printiipages</code> 193–194	<code>\_regquotes</code> 205
<code>\pg</code> 200	<code>\_printindexitem</code> 193	<code>\_regtfm</code> 69, 71
<code>\pgbackground</code> 7, 55, 116–117	<code>\_printinverbatim</code> 143	<code>\_regtoc</code> 133
<code>\_pgborder</code> 12–13, 130	<code>\_printitem</code> 142	<code>\_regtrick</code> 27, 212
<code>\pgbottomskip</code> 54, 116–117	<code>\_printlabel</code> 128	<code>\relax *</code>
<code>\pgforeground</code> 27	<code>\_printlayers</code> 200	<code>\relpenalty *</code>
<code>\_pgn</code> 132	<code>\_prinnumberedpar</code> 141	<code>\removelastskip</code> 63
<code>\_pgprint</code> 195	<code>\_printrefnum</code> 136, 138	<code>\_removeoutbraces</code> 135
<code>\pgr</code> 128	<code>\_printsavedcites</code> 167	<code>\_removeoutmath</code> 135
<code>\pgref</code> 12, 28, 128, 202	<code>\_printsec</code> 10, 136, 200	<code>\removespaces</code> 58
<code>\_pgrefB</code> 128	<code>\_printsecc</code> 10, 136	<code>\_repeat</code> 44
<code>\phantom</code> 96	<code>\_printtit</code> 136	<code>\repeat</code> 29, 44
<code>\picdir</code> 22, 51	<code>\_printverb</code> 144–145, 147	<code>\replfromto</code> 149
<code>\pich</code> 22, 51	<code>\_printverbline</code> 144	<code>\replmacro</code> 27
<code>\picheight</code> 22, 51	<code>\_printverblineum</code> 144	<code>\replstring</code> 27, 29, 58, 124, 159
<code>\picparams</code> 22, 51	<code>\private</code> 29, 34, 36	<code>\replthis</code> 149
<code>\picw</code> 22, 51	<code>\protected *</code>	<code>\_repltoks</code> 58
<code>\picwidth</code> 22, 51	<code>\protrudechars *</code>	<code>\report</code> 25, 28, 198
<code>\_pkglabel</code> 36	<code>\pshow</code> 199	<code>\_resetattrs</code> 117, 123
<code>\plaintexcatcodes</code> 56	<code>\pstart</code> 27	<code>\_resetfam</code> 84
<code>\plaintexsetting</code> 50	<code>\ptmunit</code> 90	<code>\resetmod</code> 72, 77, 79, 84
<code>\pllang</code> 24, 206	<code>\ptunit</code> 8, 90	<code>\_resetnamespace</code> 35–36
<code>\pmatrix</code> 97	<code>\public</code> 29, 33–34, 36	<code>\_resetnonumnotoc</code> 137
<code>\pmod</code> 96	<code>\_putforstack</code> 45	<code>\resizethefont</code> 68–70, 82
<code>\pospg</code> 29, 59	<code>\putpic</code> 24, 153	<code>\restorectable</code> 29, 56
<code>\postdisplaypenalty *</code>	<code>\puttext</code> 24, 55, 153	<code>\_restoremathsb</code> 146
<code>\posx</code> 29, 59	<code>\_putttpenalty</code> 144	<code>\_restoretline</code> 144
<code>\posy</code> 29, 59	<code>\qcasesof</code> 29, 48	<code>\_RetrieveField</code> 176
<code>\predisplaypenalty *</code>	<code>\_qqA</code> 205	<code>\_RetrieveFieldIn</code> 176
<code>\_preparebibsoring</code>	<code>\_qqB</code> 205	<code>\_reversetfm</code> 71
170–171, 174	<code>\qqquad</code> 62	
<code>\_preparesorting</code> 191		

<code>\_reversewords</code> 190, 193	<code>\seccc</code> 27	<code>\_shadowmoveto</code> 155
<code>\rgbcmykmap</code> 121, 123–124	<code>\seccfont</code> 75, 135	<code>\shipout</code> *
<code>\rgbcolordef</code> 22, 121–122, 124	<code>\seccx</code> 137	<code>\shortcitations</code> 20, 28, 167
<code>\_rgbtocmyk</code> 123–124	<code>\secfont</code> 75, 135	<code>\show</code> *
<code>\right</code> *	<code>\secl</code> 139	<code>\showbox</code> *
<code>\rightarrowfill</code> 66	<code>\seclp</code> 139	<code>\showboxbreadth</code> *
<code>\righthyphenmin</code> *	<code>\secondpass</code> 190	<code>\showboxdepth</code> *
<code>\rightleftharpoons</code> 96	<code>\sectionlevel</code> 136	<code>\_showcolor</code> 125
<code>\rightline</code> 63	<code>\secx</code> 137	<code>\showhyphens</code> 66
<code>\_rightofcolumns</code> 163	<code>\setbaselineskip</code> 113	<code>\showlabels</code> 12, 128
<code>\rightskip</code> *	<code>\setbibmark</code> 185	<code>\showlists</code> *
<code>\rlap</code> 63	<code>\setbox</code> *	<code>\shownodes</code> 27
<code>\rm</code> 8, 71, 73, 81, 88, 102, 107	<code>\setcmykcolor</code> 21, 120–122, 124	<code>\showpglists</code> 27
<code>\_rmdigits</code> 105	<code>\setcolor</code> 120, 122, 154	<code>\showthe</code> *
<code>\_rmfixed</code> 115	<code>\setcolsize</code> 165	<code>\showtree</code> 27
<code>\_rmgreek</code> , 105	<code>\setctable</code> 29, 56	<code>\shyph</code> 24, 210
<code>\_rmGreek</code> 105	<code>\setff</code> 72–73, 75–76, 78, 86–87	<code>\_sizemscript</code> 89, 113
<code>\_rmletters</code> 102, 105	<code>\setflcolors</code> 154	<code>\_sizemsscript</code> 89, 113
<code>\_rmvariables</code> 105	<code>\setfontcolor</code> 78, 86–87	<code>\_sizemtext</code> 89, 113
<code>\romannumeral</code> *	<code>\setfontsize</code> 9, 68–70, 72–74, 76, 113	<code>\_sizespec</code> 70, 82
<code>\root</code> *	<code>\setgreycolor</code> 21, 120, 122	<code>\skew</code> 95
<code>\rotbox</code> 23, 55, 152	<code>\sethours</code> 27	<code>\skewchar</code> *
<code>\roundframe</code> 27	<code>\setletterspace</code> 73, 76, 78, 86–87	<code>\skip</code> *
<code>\rulewidth</code> 16, 163	<code>\setlistskip</code> 141–142	<code>\skipdef</code> *
<code>\_runboldmath</code> 115	<code>\setmainvalues</code> 113–114	<code>\skiptoel</code> 57
<code>\_runoptfn</code> 71	<code>\setmainvaluesL</code> 114–115	<code>\sklang</code> 24, 206
<code>\runsystem</code> 27	<code>\setmathdimens</code> 90, 102	<code>\skquotes</code> 205
<code>\_sansdigits</code> 105	<code>\setmathfamily</code> 89	<code>\_slantcorr</code> 203
<code>\_sansvariables</code> 105	<code>\setmathfonts</code> 114	<code>\slash</code> 62
<code>\savecatcodetable</code> *	<code>\setmathparam</code> 90	<code>\slet</code> 29, 40
<code>\_savedcites</code> 166–167	<code>\setmathsizes</code> 88, 90	<code>\_slidelayer</code> 200
<code>\_savedtttchar</code> 143	<code>\setmathstyle</code> 100	<code>\_slidelinks</code> 201
<code>\_savedtttcharc</code> 143	<code>\setminutes</code> 27	<code>\slideopen</code> 201
<code>\_savemathsb</code> 146	<code>\setnewmeaning</code> 84	<code>\_slidepage</code> 200
<code>\sb</code> 93	<code>\setpagerightoffset</code> 120	<code>\_slidepageB</code> 200
<code>\_scalebig</code> 94	<code>\setpos</code> 29, 59	<code>\slides</code> 25, 28, 153, 168, 199
<code>\_scalebigcoef</code> 94	<code>\setprimarysorting</code> 190–191	<code>\_slideshowook</code> 202
<code>\scalemain</code> 9, 114	<code>\setrgbcolor</code> 21, 120, 122, 124	<code>\slideshow</code> 200–202
<code>\scaleto</code> 27	<code>\setsecondarysorting</code> 191	<code>\smallbreak</code> 63
<code>\scaletoof</code> 27	<code>\setseconds</code> 27	<code>\smallskip</code> 62
<code>\_scantabdata</code> 160, 162	<code>\setsplitsize</code> 165	<code>\smallskipamount</code> 50
<code>\scantextokens</code> *	<code>\settabs</code> 27	<code>\smash</code> 96
<code>\scantoel</code> 57, 131, 136–137	<code>\settinglayer</code> 202	<code>\sortcitations</code> 20, 28, 167
<code>\scantokens</code> *	<code>\setunimathdimens</code> 102	<code>\_sortingdata</code> 189–190
<code>\_scantwodimens</code> 153	<code>\setverb</code> 143–144	<code>\_sortingdatalatin</code> 189–190
<code>\script</code> 88, 107	<code>\setweekday</code> 27	<code>\_sortinglang</code> 190, 192
<code>\scriptfont</code> *	<code>\setwordspace</code> 73, 76, 86–87	<code>\_SortingOrder</code> 176
<code>\_scriptmff</code> 100, 103	<code>\setwsp</code> 87	<code>\_sortrevers</code> 193
<code>\scriptscriptfont</code> *	<code>\setxhsize</code> 116	<code>\sp</code> 93
<code>\scriptscriptstyle</code> *	<code>\sfcode</code> *	<code>\space</code> 39
<code>\_scriptspacefactor</code> 90	<code>\sfont</code> 68–70	<code>\spacefactor</code> *
<code>\scriptstyle</code> *	<code>\shadedframe</code> 27	<code>\spaceskip</code> *
<code>\sdef</code> 6, 14, 24, 29, 40	<code>\shadow</code> 154	<code>\span</code> *
<code>\_sdestbox</code> 201	<code>\_shadowb</code> 155	<code>\_SpecialSort</code> 176
<code>\sec</code> 10, 12, 17–18, 28, 57, 136–137	<code>\shadowlevels</code> 155	<code>\splitmaxdepth</code> *
<code>\secc</code> 10, 12, 18, 28, 57, 136–137		<code>\splittopskip</code> *
		<code>\sqrt</code> *
		<code>\_sscriptmff</code> 103



<code>\_startitem</code> 141–142	<code>\textdirection</code> *	<code>\tracingfonts</code> *
<code>\_startverb</code> 144–145	<code>\textfont</code> *	<code>\tracinggroups</code> *
<code>\_startxlink</code> 130	<code>\textindent</code> 64	<code>\tracingifs</code> *
<code>\_stopxlink</code> 130	<code>\_textmff</code> 100, 103	<code>\tracinglostchars</code> *
<code>\string</code> *	<code>\textstyle</code> *	<code>\tracingmacros</code> *
<code>\_stringat</code> 70	<code>\the</code> *	<code>\tracingnone</code> 41
<code>\_stripzeros</code> 124	<code>\_thecapnum</code> 139–140	<code>\tracingonline</code> *
<code>\strutbox</code> 63, 113	<code>\_thecaptitle</code> 139–140	<code>\tracingoutput</code> *
<code>\style</code> 13, 142	<code>\_thechapnum</code> 136–137	<code>\tracingpages</code> *
<code>\stylenum</code> 99	<code>\thedelcodechar</code> 104	<code>\tracingparagraphs</code> *
<code>\subject</code> 25, 198	<code>\thedelcodefam</code> 104	<code>\tracingrestores</code> *
<code>\subtit</code> 199	<code>\thedimen</code> 27	<code>\tracingscantokens</code> *
<code>\supereject</code> 63	<code>\_thednum</code> 137	<code>\tracingstats</code> *
<code>\suppressfontnotfounderror</code>	<code>\_thefnum</code> 137	<code>\tracingtreedata</code> 27
*	<code>\thefontscale</code> 9, 28, 114	<code>\transformbox</code> 23, 55, 151–152
<code>\swapeqno</code> 10, 100	<code>\thefontsize</code> 9, 28, 114	<code>\_translatecolor</code> 122
<code>\sxdef</code> 29, 40	<code>\themathcodechar</code> 104	<code>\transparency</code> 22, 87, 122
<code>\_tabdata</code> 160	<code>\themathcodeclass</code> 104	<code>\_transpattr</code> 122–123
<code>\_tabdeclarec</code> 16, 161	<code>\themathcodefam</code> 104	<code>\treedata</code> 27
<code>\_tabdeclarel</code> 161	<code>\_theoutline</code> 138	<code>\treenodes</code> 27
<code>\_tabdeclarer</code> 161	<code>\thepp</code> 27	<code>\_trick</code> 212
<code>\tabiteml</code> 15, 53, 157	<code>\thepplast</code> 27	<code>\_trickprefix</code> 212
<code>\tabitemr</code> 15, 53, 157	<code>\_theseccnum</code> 136–137	<code>\_truedimen</code> 119
<code>\table</code> 14–16, 28, 53, 157–158	<code>\_theseccnum</code> 136–137	<code>\truepage</code> 27
<code>\_tableA</code> 158	<code>\_thetnum</code> 137	<code>\trycs</code> 29, 40
<code>\_tableB</code> 158–159, 161	<code>\thickmuskip</code> *	<code>\trykv</code> 29, 61
<code>\_tablebox</code> 158	<code>\_thicksk</code> 93	<code>\_tryloadfamslocal</code> 85
<code>\_tablepar</code> 161–162	<code>\thinmuskip</code> *	<code>\tsize</code> 54, 157, 159
<code>\_tableparA</code> 162	<code>\_thinneg</code> 93	<code>\_tsizelast</code> 161
<code>\_tableparB</code> 161	<code>\_thinsk</code> 93	<code>\_tsizesum</code> 159, 161
<code>\_tableparbox</code> 161–162	<code>\thinspace</code> 62	<code>\tskip</code> 15, 162
<code>\_tableparC</code> 161	<code>\_thisentry</code> 175	<code>\tt</code> 13, 72–75, 81–82, 88
<code>\_tableparD</code> 162	<code>\thisoutline</code> 13, 138	<code>\_ttdigits</code> 105
<code>\_tablew</code> 158–159	<code>\thistable</code> 53, 157	<code>\_ttfamv</code> 84
<code>\_tableW</code> 158	<code>\time</code> *	<code>\_ttfont</code> 74, 84, 143
<code>\tablinespace</code> 53, 158, 162	<code>\tit</code> 10, 28, 53, 136	<code>\ttindent</code> 17–18, 52
<code>\tabnodes</code> 27	<code>\_titfont</code> 75, 135	<code>\ttline</code> 17, 52
<code>\_tabreplstrings</code> 159	<code>\titskip</code> 53	<code>\ttlineref</code> 27
<code>\tabs</code> 27	<code>\_tmpcatcodes</code> 56	<code>\_ttpenalty</code> 143–144
<code>\tabskip</code> *	<code>\tnote</code> 27	<code>\ttraggedright</code> 64, 74
<code>\tabskipl</code> 54, 157	<code>\_tocborder</code> 12, 130	<code>\ttshift</code> 52
<code>\_tabskipmid</code> 159	<code>\_tocdotfill</code> 132	<code>\_ttskip</code> 143
<code>\tabskipr</code> 54, 157	<code>\_tocline</code> 131–132	<code>\_ttunifont</code> 74–75, 82
<code>\tabspaces</code> 52	<code>\_toclist</code> 131, 133	<code>\_ttvariables</code> 105
<code>\tabstrut</code> 53, 158	<code>\_tocpar</code> 132	<code>\twoblocks</code> 27
<code>\_tbreset</code> 153	<code>\tocrefnum</code> 129, 132	<code>\typoscale</code> 8–9, 28, 72,
<code>\tdnum</code> 27	<code>\today</code> 204	113–115
<code>\tenbf</code> 67	<code>\toks</code> *	<code>\typosize</code> 8–9, 28, 72, 75,
<code>\tenbi</code> 67	<code>\toksdef</code> *	113–115
<code>\tenit</code> 67	<code>\tolerance</code> *	<code>\uccode</code> *
<code>\tenrm</code> 67	<code>\topglue</code> 62	<code>\Uchar</code> *
<code>\tentt</code> 67	<code>\topins</code> 116, 118	<code>\uchyph</code> *
<code>\_testAleB</code> 191	<code>\topinsert</code> 11, 116, 118	<code>\Udelcode</code> *
<code>\_testAleBsecondary</code> 191,	<code>\topskip</code> *	<code>\Udelimiter</code> *
193	<code>\totalpages</code> 26, 210	<code>\ufont</code> 68–70
<code>\_testAleBsecondaryX</code> 191	<code>\tracingall</code> 41	<code>\ulink</code> 12–13, 130
<code>\_testcommentchars</code> 144, 147	<code>\tracingassigns</code> *	<code>\Umathaccent</code> *
<code>\TeX</code> 202	<code>\tracingcommands</code> *	<code>\Umathchar</code> *

<code>\Umathchardef *</code>	<code>\usemathstyle</code> 100	<code>\vvkern</code> 54
<code>\_umathcharholes</code> 105	<code>\useoptex</code> 26, 210	<code>\_wbib</code> 168
<code>\Umathcode *</code>	<code>\useOpTeX</code> 26, 210	<code>\wd *</code>
<code>\_umathrange</code> 105, 107	<code>\useseccond</code> 29, 39	<code>\White</code> 121
<code>\_umathrangegreek</code> 107	<code>\uslang</code> 210	<code>\widowpenalty *</code>
<code>\_umathrangeGREEK</code> 107	<code>\uv</code> 210	<code>\_wipeepar</code> 139
<code>\_umathrangespec</code> 105, 107	<code>\vadjust *</code>	<code>\wlabel</code> 12, 128, 202
<code>\underbar</code> 63	<code>\valign *</code>	<code>\wlog</code> 29, 37
<code>\underbrace</code> 95	<code>\vbadness *</code>	<code>\_wref</code> 126
<code>\unexpanded *</code>	<code>\vbot</code> 27	<code>\write *</code>
<code>\unhbox *</code>	<code>\vbox *</code>	<code>\wterm</code> 30, 37
<code>\unhcopy *</code>	<code>\vcent</code> 27	<code>\xargs</code> 30, 36
<code>\_uniaccents</code> 65	<code>\vcenter *</code>	<code>\_Xbib</code> 166, 168
<code>\_unichars</code> 64	<code>\_vcomments</code> 147	<code>\xcasesof</code> 29, 48
<code>\_unifmodtt</code> 82	<code>\vdots</code> 95	<code>\_Xcite</code> 167
<code>\_unimathfont</code> 101–102	<code>\_verbatimcatcodes</code> 143	<code>\_xcompoundchars</code> 190
<code>\unless *</code>	<code>\verbchar</code> 16–17, 28, 52, 143	<code>\xdef *</code>
<code>\unpenalty *</code>	<code>\verbhyphens</code> 27	<code>\_Xeqlbox</code> 163
<code>\_unresolvedrefs</code> 127	<code>\verbinput</code> 17–18, 28, 51–52, 145–146, 230	<code>\XeTeX</code> 202
<code>\unskip *</code>	<code>\vfil *</code>	<code>\_xfamv</code> 84
<code>\_unsskip</code> 161	<code>\vfill *</code>	<code>\xfipoint</code> 44
<code>\unvbox *</code>	<code>\vfilneg *</code>	<code>\_Xfnote</code> 196
<code>\unvcopy *</code>	<code>\vfootnote</code> 117, 196–197	<code>\xfontname</code> 70
<code>\upbracefill</code> 66	<code>\vfuzz *</code>	<code>\_xhsize</code> 116
<code>\uppercase *</code>	<code>\vglue</code> 62	<code>\_Xindex</code> 195
<code>\Uradical *</code>	<code>\_vidolines</code> 145	<code>\_Xlabel</code> 126, 128
<code>\Urange</code> 101, 104	<code>\_vifile</code> 143	<code>\_xlink</code> 129–130
<code>\url</code> 12–13, 130	<code>\_viline</code> 143	<code>\_xlinkactive</code> 129
<code>\_urlA</code> 130	<code>\_vinolines</code> 145	<code>\_xloop</code> 44
<code>\_urlaction</code> 129	<code>\_viscanminus</code> 145	<code>\xloop</code> 29, 44
<code>\_urlB</code> 130	<code>\_viscanparameter</code> 145	<code>\_Xmnote</code> 197
<code>\_urlborder</code> 12, 130	<code>\visiblesp</code> 147	<code>\_Xpage</code> 126–127, 131, 196, 210
<code>\_urlbskip</code> 130	<code>\vitt</code> 230	<code>\xparshape</code> 27
<code>\_urlC</code> 130	<code>\voffset *</code>	<code>\_Xpos</code> 59
<code>\_urlfont</code> 74, 84, 131	<code>\voidbox</code> 43	<code>\Xrefversion</code> 127
<code>\_urlgskip</code> 130	<code>\_Von</code> 170, 176	<code>\xreplstring</code> 27
<code>\_urlskip</code> 130	<code>\vphantom</code> 96	<code>\_xscan</code> 150
<code>\_urlxskip</code> 130	<code>\vrule *</code>	<code>\_xscanR</code> 150
<code>\usebib</code> 20, 28, 167, 169–171, 174, 177	<code>\vsize *</code>	<code>\xspaceskip *</code>
<code>\_usedirectly</code> 64	<code>\vskip *</code>	<code>\_Xtoc</code> 57, 126, 131, 135
<code>\useit</code> 29, 39	<code>\vspan</code> 16, 162	<code>\year *</code>
<code>\useK</code> 121, 123, 125	<code>\vsplit *</code>	<code>\Yellow</code> 21, 121
<code>\_uselang</code> 209	<code>\vss *</code>	<code>\_zerotabrule</code> 162
<code>\uselanguage</code> 24, 206, 209	<code>\vtop *</code>	<code>\_zo</code> 43
		<code>\_zoskip</code> 43