OpTEX

Format Based on Plain TEX and OPmac[1]

Version 0.10

Petr Olšák, 2020

http://petr.olsak.net/optex

OpTEX is LuaTEX format with Plain TEX and OPmac. Only LuaTEX engine is supported.

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

The main goal of OpTEX is:

- OpTEX keeps the simplicity (like in Plain TEX and OPmac macros).
- There is no old obscurities concerning with various 8-bit encodings and various engines.
- OpTEX provides a powerful Fonts Selection System (for Unicode font families, of course).
- OpTEX supports hyphenations of all languages installed in your TEX system.
- All features from OPmac macros are copied. For example sorting words in the Index[2], reading `.bib` files directly[2], syntax highlighting[2], colors, graphics, hyperlinks, references).
- Macros are documented in the same place where code is.
- User name space of control sequences is separated from internal name space of OpTEX and primitives (`\foo` versus `\_foo`). The name spaces for macro writers are designed too.

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

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

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

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

[1] The OPmac package is a set of simple additional macros to Plain TEX. It enables users to take advantage of LATEX functionality but keeps Plain TEX simplicity. See http://petr.olsak.net/opmac-e.html for more information about it.

[2] All these features are implemented by TEX macros, no external program is needed.

Contents

Chapter 1
User documentation

1.1 Starting with OpTEX

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

```
optex document
```

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

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

A minimal document should be

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

The first line `\fontfam[LMfonts]` tells that Unicode Latin Modern fonts (derived from Computer Modern) are used. If you omit this line then preloaded Latin Modern fonts are used but preloaded fonts cannot be in Unicode[1]. So the sentence `Hello World` will be OK without the first line, but you cannot print such sentence in another languages (for example `Ahoj světe!`) where Unicode fonts are needed because of the characters like `ě` are not mapped correctly in preloaded fonts.

A somewhat larger example with common settings should be:

```
\fontfam[Termes]  % selecting Unicode font family Termes (section 1.3.1)
\typosize[11/13]  % setting default font size and baselineskip (sec. 1.3.2)
\margins/1 a4 (1,1,1,1)in % setting A4 paper, 1 in margins (section 1.2.1)
\cslang           % Czech hyphenation patterns (section 1.7.1)

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

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

1.2 Page layout

1.2.1 Setting the margins

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

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

Parameters are:

- ⟨*pg*⟩ ... `1` or `2` specifies one-page or two-pages design.
- ⟨*fmt*⟩ ... paper format (a4, a4l, a5, letter, etc. or user defined).
- ⟨*left*⟩, ⟨*right*⟩, ⟨*top*⟩, ⟨*bot*⟩ ... gives the amount of left, right, top and bottom margins.
- ⟨*unit*⟩ ... unit used for values ⟨*left*⟩, ⟨*right*⟩, ⟨*top*⟩, ⟨*bot*⟩.

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

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

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

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

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

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

The ⟨*fmt*⟩ can be also in the form `(`⟨*width*⟩`,`⟨*height*⟩`)`⟨*unit*⟩ where ⟨*unit*⟩ is optional. If it is missing then ⟨*unit*⟩ after margins specification is used. For example:

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

declares the paper 100×200 mm with all four margins 7 mm. The spaces before and after ⟨*fmt*⟩ parameter are necessary.

The command `\magscale[`⟨*factor*⟩`]` scales the whole typesetting area. The fixed point of such scaling is the upper left corner of the paper sheet. Typesetting (breakpoints etc.) is unchanged. All units are relative after such scaling. Only paper formats dimensions stays unscaled. Example:

```
\margins/2 a5 (22,17,19,21)mm
\magscale[1414] \margins/1 a4 (,,,)mm
```

The first line sets the `\hsize` and `\vsize` and margins for final printing at a5 format. The setting on the second line centers the scaled typesetting area to the true a4 paper while breaking points for paragraphs and pages are unchanged. It may be usable for review printing. After review is done, the second line can be commented out.

1.2.2 Concept of default page

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

The margins declared by `\margins` macro (documented in the previous section 1.2.1) is concerned to the page body, i.e. the `\headline` and `\footline` are placed to the top and bottom margins.

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

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

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

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

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

1.2.3 Footnotes and marginal notes

The Plain TEX's macro \footnote can be used as usual. But a new macro \fnote{⟨*text*⟩} is defined. The footnote mark is added automatically and it is numbered on each chapter from one[2]. The ⟨*text*⟩ is scaled to 80 %. User can redefine footnote mark or scaling, as shown in the file `fnotes.opm`.

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

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

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

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

1.3 Fonts

1.3.1 Font families

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

If you write \fontfam[?] then all font families registered in OpTEX are listed on the terminal and in the log file. If you write \fontfam[catalog] then a catalog of all fonts registered in

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

OpTEX and available in your TEX system is printed. The instructions how to register your own font family is appended in such catalog.

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

The font modifiers (`\caps`, `\cond` for example) can be used before a variant selector and they can be (independently) combined: `\caps\it` or `\cond\caps\bf`. The modifiers keeps their internal setting until group ends or until another modifier which negates the previous feature is used. So `\caps \rm First text \it Second text` gives FIRST TEXT *SECOND TEXT*.

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

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

1.3.2 Font sizes

The command `\typosize[`⟨*fontsize*⟩`/`⟨*baselineskip*⟩`]` sets the font size of text and math fonts and baselineskip. If one of these two parameters is empty, the corresponding feature stays unchanged. Don't write the unit of these parameters. The unit is internally set to `\ptunit` which is 1pt by default. You can change the unit by the command `\ptunit=`⟨*something-else*⟩, for instance `\ptunit=1mm` enlarges all font sizes declared by `\typosize`. Examples:

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

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

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

```
\typoscale[800/800]    % fonts and baselineskip re-size to 80 %
\typoscale[\magstep2/] % fonts bigger 1,44times (\magstep2 expands to 1440)
```

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

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

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

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

There is "low level" `\setfontsize{`⟨*size-spec*⟩`}` command which behaves like a font modifier and sets given font size used by next variant selectors. It doesn't change the font size immediately, but following variant slelector does it. For example `\setfontsize{at15pt}\currvar` sets current variant to 15pt.

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

More information about resizing of fonts is documented in the section 2.11.

1.3.3 Typesetting math

OpTEX preloads a collection of 7bit Computer Modern math fonts and AMS fonts in its format for math typesetting. You can use them in any size and in the `\boldmath` variant. Most declared text font families (see `\fontfam` in the section 1.3.1) are configured with recommended Unicode math font. This font is automatically loaded unless you specify `\noloadmath` before first `\fontfam` command. See log file for more information about loading text font family and Unicode math fonts. If you prefer another Unicode math font, specify it by `\loadmath{[`⟨*font-file*⟩`]}` or `\loadmath{font-name}` before first `\fontfam` command.

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

The following math alphabets are available:

`\mit`	`% mathematical variables`	$abc{-}xyz, ABC{-}XYZ$
`\it`	`% text italics`	$abc{-}xyz, ABC{-}XYZ$
`\rm`	`% text roman`	abc–xyz, ABC–XYZ
`\cal`	`% normal calligraphics`	$\mathcal{ABC}{-}\mathcal{XYZ}$
`\script`	`% script`	$\mathscr{ABC}{-}\mathscr{XYZ}$
`\frak`	`% fracture`	$\mathfrak{abc}{-}\mathfrak{xyz}, \mathfrak{ABC}{-}\mathfrak{XYZ}$
`\bbchar`	`% double stroked letters`	$\mathbb{ABC}{-}\mathbb{XYZ}$
`\bf`	`% sans serif bold`	$\mathbf{abc{-}xyz, ABC{-}XYZ}$
`\bi`	`% sans serif bold slanted`	$\boldsymbol{abc{-}xyz, ABC{-}XYZ}$

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

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

1.4.1 Chapters and sections

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

\tit Document title ⟨*end of line*⟩
\chap Chapter title ⟨*end of line*⟩
\sec Section title ⟨*end of line*⟩
\secc Subsection title ⟨*end of line*⟩

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

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

The implicit design of the titles of chapter etc. are implemented in the macros _printchap, _printsec and _printsecc. A designer can simply change these macros if he/she needs another behavior.

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

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

The chapter, section or subsection isn't numbered if the \nonum precedes. And the chapter, section or subsection isn't delivered to the table of contents if \notoc precedes. You can combine both prefixes.

1.4.2 Another numbered objects

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

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

If the \eqalignno is used, then user can put \eqmark to the last column before \cr. For example:

```
\eqalignno{
    a^2+b^2 &= c^2 \cr
          c &= \sqrt{a^2+b^2} & \eqmark \cr}
```

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

```
\caption/t The dependency of the computer-dependency on the age.
\cskip
\noindent\hfil\table{rl}{
    age   & value \crl\noalign{\smallskip}
    0--1  & unmeasured \cr
```

```
 1--6  & observable \cr
 6--12 & significant \cr
12--20 & extremal \cr
20--40 & normal \cr
40--60 & various \cr
60--$\infty$ & moderate}
```

This example produces:

Table 1.4.1 The dependency of the computer-dependency on the age.

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

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

The macro `\caption/f` behaves like `\caption/t` but it is intended for figure captions with independent numbering. The word (Table, Figure) depends on the actual selected language (see section 1.7.1 about languages).

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

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

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

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

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

1.4.3 References

Each automatically numbered object documented in sections 1.4.1 and 1.4.2 can be referenced if optional parameter `[`⟨*label*⟩`]` is appended to `\chap`, `\sec`, `\secc`, `\caption/t`, `\caption/f` or `\eqmark`. The alternative syntax is to use `\label[`⟨*label*⟩`]` before mentioned commands (not necessarily directly before). The reference is realized by `\ref[`⟨*label*⟩`]` or `\pgref[`⟨*label*⟩`]`. Example:

[3] This feature can be changed, see the section 2.25 in the technical documentation.

```
\sec[beatle] About Beatles

\noindent\hfil\table{rl}{...} % the table
\cskip
\caption/t [comp-depend] The dependency of the comp-dependency on the age.

\label[pythagoras]
$$ a^2 + b^2 = c^2 \eqmark $$

Now we can point to the section~\ref[beatle] on the page~\pgref[beatle]
or write something about the equation~\ref[pythagoras]. Finally there
is an interesting Table~\ref[comp-depend].
```

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

You can use the `\label[`⟨*label*⟩`]` before the `\theorem`, `\definition` etc. (macros defined by `\numberedpar`) if you want to reference these numbered objects. You can't use `\theorem[`⟨*label*⟩`]` because the optional parameter is reserved to another purpose here.

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

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

1.4.4 Hyperlinks, outlines

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

- numbers generated by `\ref` or `\pgref`,
- numbers of chapters, sections and subsections in the table of contents,
- numbers or marks generated by `\cite` command (bibliography references),
- texts printed by `\url` or `\ulink` commands.

The last object is an external link and it is colored by ⟨*color-out*⟩. Others links are internal and they are colored by ⟨*color-in*⟩. Example:

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

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

```
\def\tocborder {1 0 0}  % links in table of contents: red frame
\def\pgborder {0 1 0}   % links to pages: green frame
\def\citeborder {0 0 1} % links to references: blue frame
```

By default these macros are not defined. It means that no frames are created.

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

There are "low level" commands to create the links. You can specify the destination of the internal link by `\dest[`⟨*type*⟩`:`⟨*label*⟩`]`. The active text linked to the `\dest` can be created

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

The \url macro prints its parameter in \tt font and creates a potential breakpoints in it (after slash or dot, for example). If \hyperlinks declaration is used then the parameter of \url is treated as an external URL link. An example: \url{http://www.olsak.net} creates http://www.olsak.net. The characters %, \, #, { and } have to be protected by backslash in the \url argument, the other special characters ~, ^, & can be written as single character[4]. You can insert the \| command in the \url argument as a potential breakpoint.

If the linked text have to be different than the URL, you can use \ulink[⟨url⟩]{text} macro. For example: \ulink[http://petr.olsak.net/optex]{\OpTeX/ page} outputs to the text OpTEX page.

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

If you are using a special unprotected macro in section titles then \outlines macro may crash. You must declare variant of the macro for outlines case which is expandable. Use \regmacro in such case. See the section 1.5.1 for more information about \regmacro.

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

1.4.5 Lists

The list of items is surrounded by \begitems and \enditems commands. The asterisk (*) is active within this environment and it starts one item. The item style can be chosen by the \style parameter written after \begitems:

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

For example:

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

[4] More exactly, there are the same rules as for \code command, see section 1.4.7.

```
* Finito
\enditems
```

produces:

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

Another style can be defined by the command `\sdef{_item:`⟨*style*⟩`}{`⟨*text*⟩`}`. Default item can be set by `\defaultitem={`⟨*text*⟩`}`. The list environments can be nested. Each new level of item is indented by next multiple of `\iindent` value which is set to `\parindent` by default. The `\ilevel` register says what level of items is currently processed. Each `\begitems` starts `\everylist` tokens register. You can set, for example:

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

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

1.4.6 Tables

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

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

generates the following result:

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

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

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

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

The command `\cr` used in the ⟨*data*⟩ part of the table is generally known. It marks end row of the table. Moreover OpTeX defines following similar commands:

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

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

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

```
\everytable={}       % code used 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
```

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

If there is an item which spans to more than one column in the table then 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 `|` only after `c`, `l` or `r` letter in `\mspan` ⟨*declaration*⟩. The exception is only in the case when `\mspan` includes first column and the table have rules on the left side. The example of `\mspan` usage is below.

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

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

creates the following result:

Title		
first	second	third
seven	eight	nine

The `c`, `l`, `r` and `p` are default "declaration letters" but you can define more such letters by `\def\_tabdeclare`⟨*letter*⟩`{`⟨*left*⟩`##`⟨*right*⟩`}`. More about it is in technical documentation in the section 2.29.

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

Many tips about tables can be seen on 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 \activettchar⟨*char*⟩. For example \activettchar` declares the character ` for in-line verbatim markup. And you can use `\relax` for verbatim \relax (for example). Another alternative of printing in-line verbatim text is \code{⟨*text*⟩} (see below).

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

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

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

```
\everytt={\adef!{?}\adef?{!}}
\begtt
Each occurrence of the exclamation mark will be changed to
the question mark and vice versa. Really? You can try it!
\endtt
```

The \adef command sets its parameter as active *after* the parameter of \everytt is read. So you don't have to worry about active categories in this parameter.

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

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

```
\everytt={\typosize[9/11]}  % setting font size for verbatim
\everytt={\ttline=0}        % each listing will be numbered from one
\everytt={\visiblesp}       % visualization␣of␣spaces
```

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

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

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

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

You can print verbatim listing from external files by the \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 the \verbinput before 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 default). If \ttline<-1 then no line numbers are printed.

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

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

1.5 Autogenerated lists

1.5.1 Table of contents

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

The name of the section with table of contents is not printed. The direct usage of \chap or \sec isn't recommended here because the table of contents is typically not referenced to itself. You can print the unnumbered and unreferenced title of the section by the code:

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

If you need a customization of the design of the TOC, read the section 2.23.

If you are using a special macro in section or chapter titles and you need different behavior of such macro in other cases then use \regmacro{⟨*case-toc*⟩}{⟨*case-mark*⟩}{⟨*case-outline*⟩}. The parameters are applied locally in given cases. The \regmacro can be used repeatedly: then the parameters are accumulated (for more macros). If a parameter is empty then original definition is used in given case. For example:

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

1.5.2 Making the index

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

The \ii command (insert to index) declares the word separated by the space as the index item. This declaration is represented as invisible atom on the page connected to the next visible

word. The page number of the page where this atom occurs is listed in the index entry. So you can type:

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

You cannot double the word if you use the \iid instead \ii:

```
The \iid resistor is a passive electrical component ...
or:
Now we'll deal with the \iid resistor .
```

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

The multiple-words entries are commonly organized in the index as follows:

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

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

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

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

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

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

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

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

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

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

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

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

Only “pure words” can be propagated to the index by the \ii command. It means that there cannot be any macro, TEX primitive, math selector etc. But there is another possibility

to create such complex index entry. Use "pure equivalent" in the \ii parameter and map this equivalent to the real word which is printed in the index by the \iis command. Example:

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

The \iis ⟨*equivalent*⟩ {⟨*text*⟩} creates one entry in the "dictionary of the exceptions". The sorting is done by the ⟨*equivalent*⟩ but the ⟨*text*⟩ is printed in the index entry list.

The sorting rules when \makeindex runs depends on the current language. See section 1.7.1 about languages selection.

1.5.3 BibTEXing

The command \cite[⟨*label*⟩] (or \cite[⟨*label-1*⟩,⟨*label-2*⟩,...,⟨*label-n*⟩]) creates the citation in the form [42] (or [15, 19, 26]). If \shortcitations is declared at the beginning of the document then continuous sequences of numbers are re-printed like this: [3–5, 7, 9–11]. If \sortcitations is declared then numbers generated by one \cite command are sorted upward.

If \nonumcitations is declared then the marks instead numbers are generated depending on the used bib-style. For example the citations look like [Now08] or [Nowak, 2008].

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

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

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

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

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

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

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

References created manually using \bib[⟨*label*⟩] command.

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

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

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

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

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

The ⟨*bib-base*⟩ is one or more `*.bib` database source files (separated by spaces and without extension) and the ⟨*style*⟩ is the part of the filename `bib-`⟨*style*⟩`.opm` where the formatting of the references list is defined. OpTEX supports `simple` or `iso690` styles. The features of the `iso690` style is documented in the section 2.31.4 in detail.

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

1.6 Graphics

1.6.1 Colors

OpTEX provides a small number of color selectors: `\Blue`, `\Red`, `\Brown`, `\Green`, `\Yellow`, `\Cyan`, `\Magenta`, `\White`, `\Grey`, `\LightGrey` and `\Black`. User can define more such selectors by setting four CMYK components or three RGB components. For example

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

The command `\morecolors` reads more definitions of color selectors. There is about 300 color names like `\DeepPink`, `\Chocolate` etc. If there are numbered variants of the same name, then the letters B, C, etc. are appended to the name in OpTEX. For example `\Chocolate` is Chocolate1, `\ChocolateB` is Chocolate2 etc.

The color selectors work locally in groups by default but with limitations. See the technical documentation, section 2.19 for more information.

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

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

```
\colordef \myCyan {.3\Green + .5\Blue}  % 30 % green, 50 % blue, 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 substractive color space by default (RGB colors used in `\colordef` argument are converted first). If the resulting component is greater than 1 then it is truncated to 1. If a convex linear combination (as in the last example above) is used then it emulates color behavior on a painter's palette. You can use `\rgbcolordef` instead `\colordef` if you want to mix colors in the additive RGB color space.

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

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

The `\coloron` can be defined as follows:

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

1.6.2 Images

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

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

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

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

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

1.6.3 PDF transformations

All typesetting elements are transformed by linear transformation given by the current transformation matrix. The `\pdfsetmatrix` `{`⟨a⟩ ⟨b⟩ ⟨c⟩ ⟨d⟩`}` command makes the internal multiplication with the current matrix so linear transformations can be composed. One linear transformation given by the `\pdfsetmatrix` above transforms the vector $[0,1]$ to $[\langle a\rangle, \langle b\rangle]$ and $[1,0]$ to $[\langle c\rangle, \langle d\rangle]$. The stack-oriented commands `\pdfsave` and `\pdfrestore` gives a possibility of storing and restoring the current transformation matrix and the current point. The position of the current point have to be the same from TEX's point of view as from transformation point of view when `\pdfrestore` is processed. Due to this fact the `\pdfsave\rlap{`⟨*transformed text*⟩`}\pdfrestore` or something similar is recommended.

OpTEX provides two special transformation macros `\pdfscale` and `\pdfrotate`:

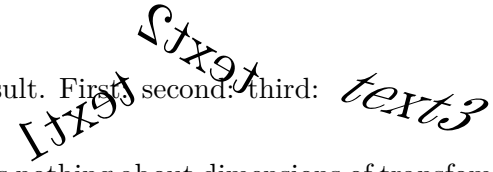
```
\pdfscale{<horizontal-factor>}{<vertical-factor>}
\pdfrotate{<angle-in-degrees>}
```

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

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

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

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

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

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

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

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
             \lwidth=0.5bp            % line width in the border
             \shadow=N                % use a shadow effect
             \overlapmargins=N        % ignore margins by surrounding text
             \hhkern=0pt \vvkern=0pt} % left-righ margin, top-bottom margin
```

The total distance from text to oval boundary is `\hhkern+\roundness` at the left and right sides and `\vvkern+\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 \lwidth=0.5bp
               \shadow=N \ignoremargins=N \hhkern=2pt \vvkern=2pt}
```

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

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

1.6.5 Putting images and texts wherever

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

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

1.7 Others

1.7.1 Using more languages

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

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

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

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

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

OpTEX generates three phrases used for captions and titles in technical articles or books: "Chapter", "Table" and "Figure". These phrases need to be known in used language and it depends on the previously used language selectors \⟨*iso-code*⟩lang. OpTEX declares these words only for few languages: Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English, If you need to use these words in another languages or you want to auto-generate more words in your macros, then you can declare it by \sdef or _langw commands as shown in the section 2.36.3.

The \makeindex command needs to know the sorting rules used in your language. OpTEX defines only few language rules for sorting: Czech, Slovak and English. How to declare sorting rules for more languages are described in the section 2.32.

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

1.7.2 Pre-defined styles

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

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

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

```
\address
  <first line of address>
  <second line of address>
  <etc.>
  <empty line>
```

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

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

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

1.7.3 Lorem ipsum dolor sit

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

1.7.4 Logos

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

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

1.7.5 The last page

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

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

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

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

1.7.6 Use OpTEX

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

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

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

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

\cite [labnames]  % refers to the item in the lits of references
\rcite [labnames] % similar to \cite but [] are not printed.
\sortcitations \shortcitations \nonumcitations % cite format
\bib [labname]  % an item in the list of references
\usebib/? (style) bib-base % direct using of .bib file, ? in {s,c}

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

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

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

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

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

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

1.9 Compatibility with Plain TEX

All macros of Plain TEX are re-written in OpTEX. Common macros should work in the same sense as in original Plain TEX. Internal control sequences like `\p@` or `\f@@t` are removed and mostly replaced by control sequences prefixed by `_` (like `\_this`). All primitives and common macros have two control sequences with the same meaning: in prefixed and unprefixed form. For example `\hbox` is equal to `\_hbox`. Internal macros of OpTEX have and use only prefixed form. User should use unprefixed forms, but prefixed forms are accessible too, because the `_` is set as a letter category code globally (in macro files and in users document too). User should re-define unprefixed forms of control sequences with no worries that something internal will be broken (only the sequence `\par` cannot be re-defined without internal change of TEX behavior because it is hard-coded in TEXs tokenization processor).

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

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

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

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

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

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 TEX is expected from the reader. You can see a short document TEX in a Nutshell or more detail TEX by topic.

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

2.1 The main initialization file

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

optex.ini

```
%% This is part of OpTeX project, see http://petr.olsak.net/optex

%% OpTeX ini file
%% Petr Olsak <project started from: Jan. 2020>
```

Category codes are set first. Note that the `_` is set to category code "letter", it can be used as a part of control sequence names. Other category codes are set as in the plain TEX.

optex.ini

```
% Catcodes:

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

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

optex.ini

```
% OpTeX version

\def\optexversion{Beta 0.10 Apr 2020}
\def\fmtname{OpTeX}
```

We check if LuaTEX engine is used at `-ini` state. And the `^^J` character is set as `\newlinechar`.

optex.ini

```
% Engine testing:

\newlinechar=`\^^J
\ifx\directlua\undefined
   \message{This format is based only on LuaTeX, use luatex -ini optex.ini^^J}
   \endinput \fi
```

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

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

optex.ini

```
% Basic .opm syntax:

\let\_endcode =\endinput
\def \_codedecl #1#2{\message{#2^^J}}%  information about .opm file
\long\def\_doc#1\_cod#2 {} % skip documentation
```

Individual `*.opm` macro files are read.

optex.ini

```
% Initialization:

\message{OpTeX (Olsak's Plain TeX) initialization <\optexversion>^^J}

\input prefixed.opm        % prefixed primitives and code syntax
\input luatex-ini.opm      % luaTeX initialization
\input basic-macros.opm    % basic macros
\input alloc.opm           % allocators for registers
\input if-macros.opm       % special \if-macros, \is-macros and loops
\input parameters.opm      % parameters setting
\input more-macros.opm     % OpTeX useful macros  (todo: doc)
\input plain-macros.opm    % plainTeX macros   (todo:doc)
\input fonts-preload.opm   % preloaded Latin Modern fonts
\input fonts-resize.opm    % font resizing (low-level macros) (todo: texdoc)
\input fonts-select.opm    % font selection system  (todo: texdoc)
\input math-preload.opm    % math fams CM + AMS preloaded  (todo: doc)
\input math-macros.opm     % basic macros for math plus mathchardefs  (todo: x)
\input math-unicode.opm    % macros for loading UnicodeMath fonts  (todo: x)
\input fonts-opmac.opm     % font managing macros from OPmac  (todo: doc)
\input output.opm          % output routine
\input margins.opm         % macros for margins setting  (todo: texdoc)
\input colors.opm          % colors
\input ref-file.opm        % ref file
\input references.opm      % references
\input hyperlinks.opm      % hyperlinks
\input maketoc.opm         % maketoc
\input outlines.opm        % PDF outlines  (todo: x)
\input pdfuni-string.opm   % PDFunicode strings for outlines  (todo: x)
\input sections.opm        % titles, chapters, sections
\input lists.opm           % lists, \begitems, \enditems
\input verbatim.opm        % verbatim
\input hi-syntax.opm       % syntax highlighting of verbatim listimgs
\input graphics.opm        % graphics
\input table.opm           % table macro
\input multicolumns.opm    % more columns by \begmulti ...\endmulti
\input cite-bib.opm        % Bibliography, \cite
\input makeindex.opm       % Make index and sorting
\input fnotes.opm          % \fnotes, \mnotes
\input styles.opm          % styles \report, \letter
\input logos.opm           % standard logos
\input uni-lcuc.opm        % Setting lccodes and uccodes for Unicode characters
\input hyphen-lan.opm      % initialization of hyphenation patterns  (todo: doc)
\input languages.opm       % languages
\input others.opm          % miscenaleous
```

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

optex.ini

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

```

96 _dump

2.2 Concept of name spaces of control sequences

2.2.1 Prefixing internal control sequences

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

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

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

We never change catcode of _, so internal macros can be redefined by user without problems if it is desired. We need not something like \makealetter from LATEX.

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

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

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

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

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

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

2.2.2 Name space of control sequences for users

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

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

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

  ```
  \int_a^b    ... is interpreted as \int _a^b
  \max_M      ... is interpreted as \max _M
  \alpha_{ij} ... is interpreted as \alpha _{ij}
  ```

  This feature is implemented using lua code at input processor level, see the section 2.14 for more details. You can deactivate this feature by \mathsboff. After this, you can still write $\int_a^b$ (Unicode) or $\int _a^b$ without problems but \int_a^b yields to undefined control sequence \int_a. You can activate this feature again by \mathsbon. The effect will take shape from next line read from input file.
- Control sequences in the form _⟨*pkg*⟩_⟨*word*⟩ is intended for package writers as internal macros for a package with ⟨*pkg*⟩ identifier, see section 2.2.3.

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

2.2.3 Name spaces for package writers

Package writer should use internal names in the form `\_`⟨*pgk*⟩`_`⟨*sequence*⟩, where ⟨*pkg*⟩ is a package label. For example: `\_qr_newbase` from `qrcode-x.tex` package.

The package writer needs not write repeatedly `\_pkg_foo \_pkg_bar` etc. again and again in the macro file.[1] When `\_namespace {`⟨*pkg*⟩`}` is declared at the beginning of the macro file then all occurrences of `\.foo` will be replaced by `\_`⟨*pkg*⟩`_foo` at the input processor level. The macro writer can write (and backward can read his 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 `\_encode` command or when another `\_namespace` is used. This command checks if the package label is not declared by the `\_namespace` twice.

The `\_public` macro does `\let\foo = \_`⟨*pkg*⟩`_foo` when `\_namespace {`⟨*pkg*⟩`}` is declared. And the `\_private` macro does reverse operation to it.

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 Macro files syntax

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

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

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

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

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

2.2.5 The implementation of the name spaces

prefixed.opm

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

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

prefixed.opm

```
\let\_directlua = \directlua
\_directlua {
    % enable all TeX primitives with _ prefix
    tex.enableprimitives('_', tex.extraprimitives('tex'))
    % enable all primitives without prefixing
    tex.enableprimitives('', tex.extraprimitives())
    % enable all primitives with _ prefix
    tex.enableprimitives('_', tex.extraprimitives())
}
```

`\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.
`\xargs` ⟨*what*⟩ ⟨*sequence*⟩ ⟨*sequence*⟩ ... ; does ⟨*what*⟩⟨*sequence*⟩ for each sequences.

prefixed.opm

```
\let\_ea =\_expandafter  % usefull shortcut

\_long\_def \_xargs #1#2{\_ifx #2;\_else \_ea#1\_ea#2\_ea\_xargs \_ea #1\_fi}

\_def \_pkglabel{}
\_def \_public {\_xargs \_publicA}
\_def \_publicA #1{\_ea\_let \_ea#1\_csname \_pkglabel _\_csstring #1\_endcsname}

```

[1] We have not adatped the idea from expl3 language:)

```
\_def \_private {\_xargs \_privateA}
\_def \_privateA #1{\_ea\_let \_csname \_pkglabel _\_csstring #1\_endcsname =#1}

\_public \public \private \xargs \ea ;
```

Each macro file should begin with _codedecl \macro {⟨*info*⟩}. If the \macro is defined already then the \endpinput protects to read such file more than one times. Else the ⟨*info*⟩ is printed to the terminal and the file is read.
The _endcode is defined as \endinput in the optex.ini file. \wterm {⟨*text*⟩} prints ⟨*text*⟩ to the terminal and to the .log file (as in plain TEX).

prefixed.opm

```
\_def \_codedecl #1#2{%
   \_ifx #1\_undefined \_wterm{#2}%
   \_else \_expandafter \_endinput \_fi
}
\_def \_wterm {\_immediate \_write16 }

\_public \wterm ;
```

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

prefixed.opm

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

The _mathsbon and _mathsboff are defined in math-macros.opm file. Now, we define the macro _namespace {⟨*pkg label*⟩} for package writers, see section 2.2.3.

prefixed.opm

```
\_def\_namespace #1{%
   \_ifcsname namesp:#1\_endcsname \_errmessage
      {The name space "#1" is used already, it cannot be used twice}%
      \_endinput
   \_else
      \_ea \_gdef \_csname namesp:#1\_endcsname {}%
      \_gdef \_pkglabel{_#1}%
      \_directlua{
      callback.register("process_input_buffer",
        function (str)
           return string.gsub(str, "\_nbb[.]([a-zA-Z])", "\_nbb _#1_\_pcent 1")
        end )
      }%
      \_gdef \_endcode {%
          \_ifmathsb \_mathsbon \_else \_mthsboff \_fi
          \_gdef \_pkglabel{_#1}%
          \_global \_let \_endcode=\_endinput
          \_endinput }%
   \_fi
}
```

2.3 pdfTEX initialization

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

luatex-ini.opm

```
\_codedecl \pdfprimitive {LuaTeX initialization code <2020-02-21>} % preloaded in format

\_let\_pdfpagewidth        \pagewidth
\_let\_pdfpageheight       \pageheight
\_let\_pdfadjustspacing    \adjustspacing
\_let\_pdfprotrudechars    \protrudechars
\_let\_pdfnoligatures      \ignoreligaturesinfont
\_let\_pdffontexpand       \expandglyphsinfont
\_let\_pdfcopyfont         \copyfont
\_let\_pdfxform            \saveboxresource
\_let\_pdflastxform        \lastsavedboxresourceindex
\_let\_pdfrefxform         \useboxresource
\_let\_pdfximage           \saveimageresource
\_let\_pdflastximage       \lastsavedimageresourceindex
```

```
\_let\_pdflastximagepages  \lastsavedimageresourcepages
\_let\_pdfrefximage        \useimageresource
\_let\_pdfsavepos          \savepos
\_let\_pdflastxpos         \lastxpos
\_let\_pdflastypos         \lastypos
\_let\_pdfoutput           \outputmode
\_let\_pdfdraftmode        \draftmode
\_let\_pdfpxdimen          \pxdimen
\_let\_pdfinsertht         \insertht
\_let\_pdfnormaldeviate    \normaldeviate
\_let\_pdfuniformdeviate   \uniformdeviate
\_let\_pdfsetrandomseed    \setrandomseed
\_let\_pdfrandomseed       \randomseed
\_let\_pdfprimitive        \primitive
\_let\_ifpdfprimitive      \ifprimitive
\_let\_ifpdfabsnum         \ifabsnum
\_let\_ifpdfabsdim         \ifabsdim

\_public
   \pdfpagewidth \pdfpageheight \pdfadjustspacing \pdfprotrudechars
   \pdfnoligatures \pdffontexpand \pdfcopyfont \pdfxform \pdflastxform
   \pdfrefxform \pdfximage \pdflastximage \pdflastximagepages \pdfrefximage
   \pdfsavepos \pdflastxpos \pdflastypos \pdfoutput \pdfdraftmode \pdfpxdimen
   \pdfinsertht \pdfnormaldeviate \pdfuniformdeviate \pdfsetrandomseed
   \pdfrandomseed \pdfprimitive \ifpdfprimitive \ifpdfabsnum \ifpdfabsdim ;

\_directlua {tex.enableprimitives('pdf',{'tracingfonts'})}

\_protected\_def \_pdftexversion     {\_numexpr 140\_relax}
           \_def \_pdftexrevision    {7}
\_protected\_def \_pdflastlink       {\_numexpr\_pdffeedback lastlink\_relax}
\_protected\_def \_pdfretval         {\_numexpr\_pdffeedback retval\_relax}
\_protected\_def \_pdflastobj        {\_numexpr\_pdffeedback lastobj\_relax}
\_protected\_def \_pdflastannot      {\_numexpr\_pdffeedback lastannot\_relax}
           \_def \_pdfxformname      {\_pdffeedback xformname}
{\_outputmode=1
           \_xdef\_pdfcreationdate   {\_pdffeedback creationdate}
}
           \_def \_pdffontname       {\_pdffeedback fontname}
           \_def \_pdffontobjnum     {\_pdffeedback fontobjnum}
           \_def \_pdffontsize       {\_pdffeedback fontsize}
           \_def \_pdfpageref        {\_pdffeedback pageref}
           \_def \_pdfcolorstackinit {\_pdffeedback colorstackinit}
\_protected\_def \_pdfliteral        {\_pdfextension literal}
\_protected\_def \_pdfcolorstack     {\_pdfextension colorstack}
\_protected\_def \_pdfsetmatrix      {\_pdfextension setmatrix}
\_protected\_def \_pdfsave           {\_pdfextension save\relax}
\_protected\_def \_pdfrestore        {\_pdfextension restore\relax}
\_protected\_def \_pdfobj            {\_pdfextension obj }
\_protected\_def \_pdfrefobj         {\_pdfextension refobj }
\_protected\_def \_pdfannot          {\_pdfextension annot }
\_protected\_def \_pdfstartlink      {\_pdfextension startlink }
\_protected\_def \_pdfendlink        {\_pdfextension endlink\relax}
\_protected\_def \_pdfoutline        {\_pdfextension outline }
\_protected\_def \_pdfdest           {\_pdfextension dest }
\_protected\_def \_pdfthread         {\_pdfextension thread }
\_protected\_def \_pdfstartthread    {\_pdfextension startthread }
\_protected\_def \_pdfendthread      {\_pdfextension endthread\relax}
\_protected\_def \_pdfinfo           {\_pdfextension info }
\_protected\_def \_pdfcatalog        {\_pdfextension catalog }
\_protected\_def \_pdfnames          {\_pdfextension names }
\_protected\_def \_pdfincludechars   {\_pdfextension includechars }
\_protected\_def \_pdffontattr       {\_pdfextension fontattr }
\_protected\_def \_pdfmapfile        {\_pdfextension mapfile }
\_protected\_def \_pdfmapline        {\_pdfextension mapline }
\_protected\_def \_pdftrailer        {\_pdfextension trailer }
\_protected\_def \_pdfglyphtounicode {\_pdfextension glyphtounicode }

\_protected\_edef\_pdfcompresslevel       {\pdfvariable compresslevel}
```

```
\_protected\_edef\_pdfobjcompresslevel    {\pdfvariable objcompresslevel}
\_protected\_edef\_pdfdecimaldigits       {\pdfvariable decimaldigits}
\_protected\_edef\_pdfgamma               {\pdfvariable gamma}
\_protected\_edef\_pdfimageresolution     {\pdfvariable imageresolution}
\_protected\_edef\_pdfimageapplygamma     {\pdfvariable imageapplygamma}
\_protected\_edef\_pdfimagegamma          {\pdfvariable imagegamma}
\_protected\_edef\_pdfimagehicolor        {\pdfvariable imagehicolor}
\_protected\_edef\_pdfimageaddfilename    {\pdfvariable imageaddfilename}
\_protected\_edef\_pdfpkresolution        {\pdfvariable pkresolution}
\_protected\_edef\_pdfinclusioncopyfonts  {\pdfvariable inclusioncopyfonts}
\_protected\_edef\_pdfinclusionerrorlevel {\pdfvariable inclusionerrorlevel}
\_protected\_edef\_pdfgentounicode        {\pdfvariable gentounicode}
\_protected\_edef\_pdfpagebox             {\pdfvariable pagebox}
\_protected\_edef\_pdfminorversion        {\pdfvariable minorversion}
\_protected\_edef\_pdfuniqueresname       {\pdfvariable uniqueresname}
\_protected\_edef\_pdfhorigin             {\pdfvariable horigin}
\_protected\_edef\_pdfvorigin             {\pdfvariable vorigin}
\_protected\_edef\_pdflinkmargin          {\pdfvariable linkmargin}
\_protected\_edef\_pdfdestmargin          {\pdfvariable destmargin}
\_protected\_edef\_pdfthreadmargin        {\pdfvariable threadmargin}
\_protected\_edef\_pdfpagesattr           {\pdfvariable pagesattr}
\_protected\_edef\_pdfpageattr            {\pdfvariable pageattr}
\_protected\_edef\_pdfpageresources       {\pdfvariable pageresources}
\_protected\_edef\_pdfxformattr           {\pdfvariable xformattr}
\_protected\_edef\_pdfxformresources      {\pdfvariable xformresources}
\_protected\_edef\_pdfpkmode              {\pdfvariable pkmode}

\_public
   \pdftexversion \pdftexrevision \pdflastlink \pdfretval \pdflastobj
   \pdflastannot \pdfxformname \pdfcreationdate \pdffontname \pdffontobjnum
   \pdffontsize \pdfpageref \pdfcolorstackinit \pdfliteral \pdfcolorstack
   \pdfsetmatrix \pdfsave \pdfrestore \pdfobj \pdfrefobj \pdfannot
   \pdfstartlink \pdfendlink \pdfoutline \pdfdest \pdfthread \pdfstartthread
   \pdfendthread \pdfinfo \pdfcatalog \pdfnames \pdfincludechars \pdffontattr
   \pdfmapfile \pdfmapline \pdftrailer \pdfglyphtounicode \pdfcompresslevel
   \pdfobjcompresslevel \pdfdecimaldigits \pdfgamma \pdfimageresolution
   \pdfimageapplygamma \pdfimagegamma \pdfimagehicolor \pdfimageaddfilename
   \pdfpkresolution \pdfinclusioncopyfonts \pdfinclusionerrorlevel
   \pdfgentounicode \pdfpagebox \pdfminorversion \pdfuniqueresname \pdfhorigin
   \pdfvorigin \pdflinkmargin \pdfdestmargin \pdfthreadmargin \pdfpagesattr
   \pdfpageattr \pdfpageresources \pdfxformattr \pdfxformresources \pdfpkmode ;

\_pdfminorversion     = 5
\_pdfobjcompresslevel = 2
\_pdfcompresslevel    = 9
\_pdfdecimaldigits    = 3
\_pdfpkresolution     = 600
```

2.4 Basic macros

We define first bundle of basic macros.

basic-macros.opm

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

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

basic-macros.opm

```
\_let\_bgroup={  \_let\_egroup=}
\_def \_empty {}
\_def \_space { }
\_def \_null {\_hbox{}}
\_def \_wlog {\_immediate\_write-1 } % write on log file (only)

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

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

basic-macros.opm

```
\_edef \_bslash {\_csstring\\}
```

```
\_edef \_nbb {\_bslash\_bslash}
\_edef \_pcent{\_csstring\%}

\_public \bslash \nbb \pcent ;
```

\sdef {⟨*text*⟩} is equivalent to \def\⟨*text*⟩, where \⟨*text*⟩ is a control sequence. You can use arbitrary parameter mask after \sdef{⟨*text*⟩}, don't put the (unwanted) space immediately after closing brace }.
\sxdef {⟨*text*⟩} is equivalent to \xdef\⟨*text*⟩.
\slet {⟨*textA*⟩}{⟨*textB*⟩} is equivalent to \let \⟨*textA*⟩ = \⟨*textB*⟩.

basic-macros.opm

```
\_def \_sdef #1{\_ea\_def \_csname#1\_endcsname}
\_def \_sxdef #1{\_ea\_xdef \_csname#1\_endcsname}
\_def \_slet #1#2{\_ea\_let \_csname#1\_ea\_endcsname \_csname#2\_endcsname}

\_public \sdef \sxdef \slet ;
```

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

basic-macros.opm

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

\cs {⟨*text*⟩} is only a shortcut to \csname ⟨*text*⟩\endcsname, but you need one more _ea if you need to get the real control sequence \⟨*text*⟩.
\trycs {⟨*csname*⟩}{⟨*text*⟩} expands to \⟨*csname*⟩ if it is defined else to the ⟨*text*⟩.

basic-macros.opm

```
\_def \_cs #1{\_csname#1\_endcsname}
\_def \_trycs#1#2{\_ifcsname #1\_endcsname \_csname #1\_endcsname \_else #2\_fi}
\_public \cs \trycs ;
```

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

basic-macros.opm

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

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

basic-macros.opm

```
\_def \_opwarning #1{\_wterm{WARNING: #1.}}
\_public \opwarning ;
```

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

basic-macros.opm

```
\_def\_loggingall{\_tracingcommands=3 \_tracingstats=2 \_tracingpages=1
  \_tracingoutput=1 \_tracinglostchars=1 \_tracingmacros=2
  \_tracingparagraphs=1 \_tracingrestores=1 \_tracingscantokens=1
  \_tracingifs=1 \_tracinggroups=1 \_tracingassigns=1 }
\_def\_tracingall{\_tracingonline=1 \_loggingall}

\_public \loggingall \tracingall ;
```

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 like in old plainTEX, but the range of directly used registers is wider.

Inserts are allocated form 254 to 201 using \newinsert.

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

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

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

alloc.opm

```
\_codedecl \newdimen {Allocators for registers <2020-01-23>} % loaded in format
```

The limits are set first.

alloc.opm

```
\_chardef\_maicount = 65535    % Max Allocation Index for counts registers in LuaTeX
\_let\_maidimen  = \_maicount
\_let\_maiskip   = \_maicount
\_let\_maimuskip = \_maicount
\_let\_maibox    = \_maicount
\_let\_maitoks   = \_maicount
\_chardef\_mairead  = 15
\_chardef\_maiwrite = 15
\_chardef\_maifam   = 255
```

Each allocation macro needs its own counter.

alloc.opm

```
\_countdef\_countalloc=10  \_countalloc=255
\_countdef\_dimenalloc=11  \_dimenalloc=255
\_countdef\_skipalloc=12   \_skipalloc=255
\_countdef\_muskipalloc=13 \_muskipalloc=255
\_countdef\_boxalloc=14    \_boxalloc=255
\_countdef\_toksalloc=15   \_toksalloc=255
\_countdef\_readalloc=16   \_readalloc=-1
\_countdef\_writealloc=17  \_writealloc=-1
\_countdef\_mathalloc=18   \_mathalloc=3
```

The common allocation macro _allocator \⟨*sequence*⟩ {⟨*type*⟩} \⟨*primitive declarator*⟩ is defined. This idea was used in classical plain TEX by Donald Knuth too but the macro from plain TEX seems to be more complicated:).

alloc.opm

```
\_def\_allocator #1#2#3{%
   \_global\_advance\_cs{_#2alloc}by1
   \_ifnum\_cs{_#2alloc}>\_cs{_mai#2}%
      \errmessage{No room for a new \_ea\_string\_csname #2\_endcsname}%
   \_else
      \_global#3#1=\_cs{_#2alloc}%
      \_wlog{\_string#1=\_ea\_string\_csname #2\_endcsname\_the\_cs{_#2alloc}}%
   \fi
}
```

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

alloc.opm

```
\_def\_newcount #1{\_allocator #1{count}\_countdef}
\_def\_newdimen #1{\_allocator #1{dimen}\_dimendef}
\_def\_newskip #1{\_allocator #1{skip}\_skipdef}
\_def\_newmuskip #1{\_allocator #1{muskip}\_muskipdef}
\_def\_newbox #1{\_allocator #1{box}\_chardef}
\_def\_newtoks #1{\_allocator #1{toks}\_toksdef}
\_def\_newread #1{\_allocator #1{read}\_chardef}
\_def\_newwrite #1{\_allocator #1{write}\_chardef}
\_def\_newmath #1{\_allocator #1{fam}\_chardef}

\_public \newcount \newdimen \newskip \newmuskip \newbox \newtoks \newread \newwrite \newmath ;
```

The \newinsert macro is defined differently than others.

alloc.opm

```
\_newcount\_insertalloc   \_insertalloc=255
\_chardef\_insertmin = 201

\_def\_newinsert #1{%
   \_advance\_insertalloc by-1
   \_ifnum\_insertalloc <\_insertmin
      \_errmessage {No room for a new \_string\insert}%
   \_else
      \_global\_chardef#1=\_insertalloc
      \_wlog {\_string#1=\_string\_insert\_the\_insertalloc}%
   \_fi
}
\_public \newinsert ;
```

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

alloc.opm

```
\_newcount \_attributealloc  \_attributealloc=0
\_chardef\_maiattribute=\_maicount
\_def\_newattribute #1{\_allocator #1{attribute}\_attributedef}

\_newcount \_catcodetablealloc  \_catcodetablealloc=10
\_chardef\_maicatcodetable=32767
\_def\_newcatcodetable #1{\_allocator #1{catcodetable}\_chardef}

\_public \newattribute \newcatcodetable ;
```

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

alloc.opm

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

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

alloc.opm

```
\_newdimen\_maxdimen \_maxdimen=16383.99999pt % the largest legal <dimen>
\_newskip\_hideskip \_hideskip=-1000pt plus 1fill % negative but can grow
\_newskip\_centering \_centering=0pt plus 1000pt minus 1000pt
\_newskip\_zoskip \_zoskip=0pt plus0pt minus0pt
\_newbox\_voidbox % permanently void box register

\_public \maxdimen \hideskip \centering \zoskip \voidbox ;
```

2.6 If-macros, loops, is-macros

if-macros.opm

```
\_codedecl \newif {Special if-macros, is-macros and loops <2020-04-15>} % preloaded in format
```

2.6.1 Classical \newif

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

The macro _newifi enables to declare _ifxxx and to use _xxxtrue and _xxxfalse. This means that it is usable for internal name space (_prefixed macros).

if-macros.opm

```
\_def\_newif #1{\_ea\_newifA \_string #1\_relax#1}
\_ea\_def \_ea\_newifA \_string\if #1\_relax#2{%
   \_sdef{#1true}{\_let#2=\_iftrue}%
   \_sdef{#1false}{\_let#2=\_iffalse}%
   \_let#2=\_iffalse
}
\_def\_newifi #1{\_ea\_newifiA \string#1\_relax#1}
\_ea\_def \_ea\_newifiA \string\_if #1\_relax#2{%
   \_sdef{_#1true}{\_let#2=\_iftrue}%
```

```
    \_sdef{_#1false}{\_let#2=\_iffalse}%
    \_let#2=\_iffalse
}
\_public \newif ;
```

2.6.2 Loops

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

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

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

if-macros.opm

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

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

\fornum

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

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

if-macros.opm

```
\_newcount\_frnum        % the numeric variable used in \fornum
\_def\_do{\_doundefined} % we need to ask \_ifx#1\_do ...

\_long\_def\_foreach #1\_do#2{\_putforstack
   \_immediateassignment\_def\_fbody##1{#2}%
   \_foreachA #1\_do}
\_long\_def\_foreachA #1{\_ifx\_do#1\_getforstack\_else\_fbody{#1}\_ea\_foreachA\_fi}

\_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
\_long\_def\_fornumstep#1:#2..#3\_do#4{\_putforstack
   \_immediateassigned{%
      \_def\_fbody##1{#4}%
      \_def\_fornumsgn{}%
      \_def\_fornumrel{<}%
      \_frnum=\_numexpr#2\_relax
      \_ifnum\_numexpr#3<\_frnum \_def\_fornumrel{>}\_fi %decreasing sequence
      \_ifnum\_numexpr#1\_fornumrel0 \_def\_fornumsgn{-}\_fi % correction
   }%
   \_fornumB{#3}{#1}%
}
\_def\_fornumB #1#2{\_ifnum\_numexpr#1\_fornumrel\_frnum \_getforstack \_else
   \_ea\_fbody\_ea{\_the\_frnum}%
   \_immediateassignment\_advance\_frnum by\_numexpr\_fornumsgn#2\_relax
   \_afterfi{\_fornumB{#1}{#2}}\_fi
}
\_def\_afterfi#1#2\_fi{\_fi#1}

\_def\foreach #1\do{\_foreach #1\_do}
\_def\fornum#1..#2\do{\_fornumstep 1:#1..#2\_do}
\_def\fornumstep#1:#2..#3\do{\_fornumstep #1:#2..#3\_do}
```

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

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

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

if-macros.opm

```
\_newcount\_forlevel
\_def\_putforstack{\_immediateassigned{%
   \_ifnum\_forlevel>0
      \_sdef{_frnum:\_the\_forlevel\_ea}\_ea{\_the\_frnum}%
      \_slet{_fbody:\_the\_forlevel}{_fbody}%
   \_fi
   \_advance\_forlevel by1
}}
\def\_getforstack{\_immediateassigned{%
   \_advance\_forlevel by-1
   \_ifnum\_forlevel>0
      \_slet{_fbody}{_fbody:\_the\_forlevel}%
      \_frnum=\_cs{_frnum:\_the\_forlevel}\_space
   \_fi
}}
```

2.6.3 Is-macros

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

> `\issomething` ⟨*params*⟩ `\iftrue` ⟨*codeA*⟩ `\else` ⟨*codeB*⟩ `\fi`
> or
> `\issomething` ⟨*params*⟩ `\iffalse` ⟨*codeB*⟩ `\else` ⟨*codeA*⟩ `\fi`

The `\else` part is optional. The ⟨*codeA*⟩ is processed if `\issomething`⟨*params*⟩ generates true condition. The ⟨*codeB*⟩ is processed if `\issomething`⟨*params*⟩ generates false condition.

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

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

`\isempty` `{`⟨*text*⟩`}\iftrue` is true if the ⟨*text*⟩ is empty. This macro is expandable.
`\istoksempty` ⟨*tokens variable*⟩`\iftrue` is true if the ⟨*tokens variable*⟩ is empty. It is expandable.

if-macros.opm

```
\_def \_isempty #1#2{\_if\_relax\_detokenize{#1}\_relax \_else \_ea\_unless \_fi#2}
\_def \_istoksempty #1#2{\_ea\_isempty\_ea{\_the#1}#2}
\_public \isempty \istoksempty ;
```

`\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.

if-macros.opm

```
\_def\_isequal#1#2#3{\_directlua{%
   if "\_luaescapestring{\_detokenize{#1}}"=="\_luaescapestring{\_detokenize{#2}}"
   then else tex.print("\_nbb unless") end}#3}
\_public \isequal ;
```

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

if-macros.opm

```
\_def\_ismacro#1{\_ea\_isequal\_ea{#1}}
\_public \ismacro ;
```

`\isdefined` `{`⟨*csname*⟩`}\iftrue` is true if `\`⟨*csname*⟩ is defined. The macro is expandable.

if-macros.opm

```
\_def\_isdefined #1#2{\_ifcsname #1\_endcsname \_else \_ea\_unless \_fi #2}
\_public \isdefined ;
```

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

if-macros.opm

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

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

if-macros.opm

```
\_newread \_testin
\_def\_isfile #1{%
   \_openin\_testin ={#1}\_relax
   \_ifeof\_testin \_ea\_unless
   \_else \_closein\_testin
   \_fi
}
\_public \isfile ;
```

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

if-macros.opm

```
\_newifi \_ifexistfam
\_def\_isfont#1#2{%
   \_begingroup
       \_suppressfontnotfounderror=1
       \_font\_testfont={#1}\_relax
       \_ifx\_testfont\_nullfont \_def\_tmp{\_existfamfalse \_unless}
       \else \_def\_tmp{\_existfamtrue}\_fi
   \_ea \_endgroup \_tmp #2%
}
\_public \isfont ;
```

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

if-macros.opm

```
\_long\_def\_isnextchar#1#2#3{\_begingroup\_toks0={\_endgroup#2}\_toks1={\_endgroup#3}%
   \_let\_tmp=#1\_futurelet\_next\_isnextcharA
}
\_def\_isnextcharA{\_the\_toks\_ifx\_tmp\_next0\_else1\_fi\_space}

\_public \isnextchar ;
```

2.7 Setting parameters

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

- primitive registers used in build-in algorithms of TEX,
- registers declared and used by OpTEX macros.

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

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

parameters.opm

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

2.7.1 Primitive registers

The primitive registers with the same default value as in plain TEX follow:

parameters.opm

```
\_parindent=20pt     % indentation of paragraphs
\_pretolerance=100   % parameters used in paragraph breaking algorithm
\_tolerance=200
\_hbadness=1000
\_vbadness=1000
\_doublehyphendemerits=10000
\_finalhyphendemerits=5000
\_adjdemerits=10000
\_uchyph=1
\_defaulthyphenchar=`\-
\_defaultskewchar=-1
\_hfuzz=0.1pt
\_vfuzz=0.1pt
\_overfullrule=5pt
\_linepenalty=10     % penalty between lines inside the paragraph
\_hyphenpenalty=50   % when a word is bro-ken
\_exhyphenpenalty=50 % when the hyphenmark is used explicitly
\_binoppenalty=700   % between binary operators in math
\_relpenalty=500     % between relations in math
\_brokenpenalty=100  % after lines if they end by a broken word.
\_displaywidowpenalty=50   % before last line of paragraph if display math follows
\_predisplaypenalty=10000  % above display math
\_postdisplaypenalty=0     % below display math
\_delimiterfactor=901 % parameter for scaling delimiters
\_delimitershortfall=5pt
\_nulldelimiterspace=1.2pt
\_scriptspace=0.5pt
\_maxdepth=4pt
\_splitmaxdepth=\_maxdimen
\_boxmaxdepth=\_maxdimen
\_parskip=0pt plus 1pt
\_abovedisplayskip=12pt plus 3pt minus 9pt
\_abovedisplayshortskip=0pt plus 3pt
\_belowdisplayskip=12pt plus 3pt minus 9pt
\_belowdisplayshortskip=7pt plus 3pt minus 4pt
\_parfillskip=0pt plus 1fil
\_thinmuskip=3mu
\_medmuskip=4mu plus 2mu minus 4mu
\_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

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

2.7.2 Plain TEX registers

Declared registers used in plain TEX

parameters.opm

```
% We also define special registers that function like parameters:
\_newskip\_smallskipamount \_smallskipamount=3pt plus 1pt minus 1pt
\_newskip\_medskipamount \_medskipamount=6pt plus 2pt minus 2pt
\_newskip\_bigskipamount \_bigskipamount=12pt plus 4pt minus 4pt
\_newskip\_normalbaselineskip \_normalbaselineskip=12pt
\_newskip\_normallineskip \_normallineskip=1pt
\_newdimen\_normallineskiplimit \_normallineskiplimit=0pt
\_newdimen\_jot \_jot=3pt
\_newcount\_interdisplaylinepenalty \_interdisplaylinepenalty=100
\_newcount\_interfootnotelinepenalty \_interfootnotelinepenalty=100

\_def\_normalbaselines{\_lineskip=\_normallineskip
  \_baselineskip=\_normalbaselineskip \_lineskiplimit=\_normallineskiplimit}

\_def\_frenchspacing{\_sfcode`\.=1000 \sfcode`\?=1000 \sfcode`\!=1000
```

```
  \sfcode`\:=1000 \sfcode`\;=1000 \sfcode`\,=1000 }
\_def\_nonfrenchspacing{\_sfcode`\.=3000 \sfcode`\?=3000 \sfcode`\!=3000
  \sfcode`\:=2000 \sfcode`\;=1500 \sfcode`\,=1250 }

\_public \normalbaselines \frenchspacing \nonfrenchspacing
   \smallskipamount \medskipamount \bigskipamount
   \normalbaselineskip \normallineskip \normallineskiplimit
   \jot \interdisplaylinepenalty \interfootnotelinepenalty ;
```

2.7.3 Different settings than in plain TEX

Default "baseline setting" is for 10 pt fonts (like in plain TEX). But `\typosize` and `\typoscale` macros re-declare it if another font size is used.

The `\nonfrenchspacing` is not set by default because the author of OpTEX is living in the Europe. If you set `\enlang` hyphenation patterns then `\nonfrenchspacing` is set.

parameters.opm

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

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

parameters.opm

```
\_emergencystretch=20pt % we want to use third pass of aparagraph building algoritmh
                        % we need not to keep the compatibility with old documents

\_clubpenalty=10000    % after first line of paragraph
\_widowpenalty=10000   % before last line of paragraph

\_showboxbreadth=150   % for tracing boxes
\_showboxdepth=7
\_errorcontextlines=15
\_tracinglostchars=2   % missing chracter warnings on terminal too

\_outputmode=1   % PDF ouput
\_pdfvorigin=0pt % orgin is exatly at left upper corner
\_pdfhorigin=0pt
\_hoffset=25mm   % margins are 2.5cm, no 1in
\_voffset=25mm
\_hsize=160mm    % 210mm (from A4 size) - 2*25mm (default margins)
\_vsize=244mm    % 297mm (from A4 size) - 2*25mm (default margins) -3mm baseline correction
\_pagewidth=210 true mm
\_pageheight=297 true mm
```

If you insist on plain TEX values of these parameters then you can call the `\plaintexsetting` macro.

parameters.opm

```
\_def\_plaintexsetting{%
   \_emergencystretch=0pt
   \_clubpenalty=150
   \_widowpenalty=150
   \_pdfvorigin=1in
   \_pdfhorigin=1in
   \_hoffset=0pt
   \_voffset=0pt
   \_hsize=6.5in
   \_vsize=8.9in
   \_pagewidth=8.5 true in
   \_pageheight=11 true in
   \_nonfrenchspacing
}
\_public \plaintexsetting ;
```

2.7.4 OpTEX parameters

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

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

parameters.opm

```
\_newtoks\_picdir
\_public \picdir ;
```

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

parameters.opm

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

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

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

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

parameters.opm

```
\_newtoks\_everytt
\_newtoks\_everyintt
\_public \everytt \everyintt ;
```

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

parameters.opm

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

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 right. Despite to the \ttindent, it does not shift the line numbers, only the text.
The \iindent gives default indentations used in table of contents, captions, lists, bib references,
It is strongly recommended to re-set this value if you set \parindent to another value than plain TEX default 20pt. A well typeset document should have the same dimension for all indentations, so you should say \ttindent=\parindent and \iindent=\parindent.

parameters.opm

```
\_newdimen\_ttindent \_ttindent=\_parindent % indentation in verbatim
\_newdimen\_ttshift
```

```
\_newdimen\_iindent  \_iindent=\parindent
\_public \ttindent \ttshift \iindent ;
```

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

parameters.opm

```
\_newcount \_tabspaces   \_tabspaces=3
\_public \tabspaces ;
```

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

parameters.opm

```
\_newtoks\_hicolors
\_public \hicolors ;
```

The default item mark used between `\begitems` and `\enditems` is bullet. The `\defaultitem` tokens list declare this default item mark.
The `\everyitem` tokens list is applied in vertical mode at the start of each item.
The `\everylist` tokens list is applied after group is opened by
The `\ilevel` keeps the value of current nesting level of the items list.
The `\listskipamount` gives vertical skip above and below the items list if `\ilevel=1`.

parameters.opm

```
\_newtoks\_defaultitem  \_defaultitem={$\_bullet$\_enspace}
\_newtoks\_everyitem
\_newtoks\_everylist
\_newskip \_listskipamount  \_listskipamount=\_medskipamount
\_newcount \_ilevel
\_public \defaultitem \everyitem \everylist \listskipamount \ilevel ;
```

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

parameters.opm

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

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

parameters.opm

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

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

parameters.opm

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

parameters.opm

```
\_newtoks\_bibtexhook
\_public \bibtexhook ;

\_newtoks\_everycaptiont  \_newtoks\_everycaptionf
\_public \everycaptiont \everycaptionf ;
```

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

parameters.opm

```
\_newtoks\_everyii
\_public \everyii ;
```

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

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

parameters.opm

```
\_newtoks\_everymnote
\_newdimen\_mnotesize   \_mnotesize=20mm   % the width of the mnote paragraph
\_newdimen\_mnoteindent \_mnoteindent=10pt % ditance between mnote and text
\_newdimen\_mnoteskip
\_public \everymnote \mnotesize \mnoteindent \mnoteskip ;
```

The \table parameters follows. The \thistable tokens list register should be used for giving an exception for only one \table which follows. It should change locally other parameters of the \table. It is reset to empty list after the table is printed.
The \everytable tokens list register is applied in every table. There is another difference between these two registers. The \thistable is used first, then strut and baselineskip settings are done, then \everytable is applied and then the table is printed.
\tabstrut configures the height and depth of lines in the table. You can declare \tabstrut={}, then normal baselineskip is used in the table. This can be used when you don't use horizontal nor vertical lines in tables.
\tabiteml is applied before each item, \tabitemr is applied after each item of the table.
\tablinespace is additional vertical space between horizontal rules and the lines of the table.
\hhkern gives the space between horizontal lines if they are doubled and \vvkern gives the space between such vertical lines.

parameters.opm

```
\_newtoks\_everytable \_newtoks\_thistable
\_newtoks\_tabiteml \_newtoks\_tabitemr \_newtoks\_tabstrut
\_newdimen\_tablinespace \_newdimen\_vvkern \_newdimen\_hhkern
\_everytable={}        % code used after settings in \vbox before table processing
\_thistable={}         % code used when \vbox starts, is is removed after using it
\_tabstrut={\_strut}
\_tabiteml={\_enspace} % left material in each column
\_tabitemr={\_enspace} % right material in each column
\_tablinespace=2pt     % additional vertical space before/after horizontal rules
\_vvkern=1pt           % space between double vertical line and used in \frame
\_hhkern=1pt           % space between double horizontal line and used in \frame
\_public \everytable \thistable \tabiteml \tabitemr \tabstrut \tablinespace \vvkern \hhkern ;
```

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

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

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

parameters.opm

```
\_newtoks\_headline   \_headline={}
\_newtoks\_footline   \_footline={\_hss\_rmfixed \_folio \_hss}
\_public \headline \footline ;
```

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

parameters.opm

```
\_newdimen \_headlinedist  \_headlinedist=14pt
\_newdimen \_footlinedist  \_footlinedist=24pt
\_public \headlinedist \footlinedist ;
```

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

parameters.opm
```
\_newskip \_pgbottomskip  \_pgbottomskip=0pt \_relax
\_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={} \nexptages={\headline={\fixedrm \firstmark \hfil}}
```

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

parameters.opm
```
\_newtoks \_nextpages
\_public \nextpages ;
```

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

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

parameters.opm
```
\_newtoks \_pgbackground   \_pgbackground={} % for page background
\_public \pgbackground  ;
```

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

parameters.opm
```
\_newtoks \_ovalparams
\_newtoks \_circleparams
%\_ovalparams={\_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
%              \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt }
%\_circleparams={\_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
%                \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt}

\_newdimen \_roundness    \_roundness=5mm % used in \clippingoval macro

\_public \ovalparams \circleparams \roundness ;
```

2.8 More OpTEX macros

The second bundle of OpTEX macros is here.

more-macros.opm
```
\_codedecl \eoldef {OpTeX useful macos <2020-03-15>} % preloaded in format
```

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

In order to achieve this, we declare \optexcatcodes catcode table and \plaintexcatocdes. They save the commonly used catcode tables. Note that \catcodetable is a part of LuaTEX extension. The catcodetable stack is implemented by OpTEX macros. The \setctable ⟨*catcode table*⟩ pushes current catcode table to the stack and activates catcodes from the ⟨*catcode table*⟩. The \restorectable returns to the saved catocdes from the catcode table stack. So, the \opinput macro can be implemented simply:

more-macros.opm
```
\_def\_opinput #1{\_setctable\_optexcatcodes \_input "#1" \_restorectable}

\_newcatcodetable \_optexcatcodes
\_newcatcodetable \_plaintexcatcodes

\_public \optexcatcodes \plaintexcatcodes \opinput ;

\_savecatcodetable\_optexcatcodes
{\_catcode`_=8 \savecatcodetable\plaintexcatcodes}
```

The implementation of the catcodetable stack follows.

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

more-macros.opm

```
\_newcount\_currctable \_currctable=0
\_catcodetable0

\_def\_setctable#1{\_edef\_ctablelist{{\_the\_currctable}\_ctablelist}%
   \_catcodetable#1\_relax \_currctable=#1\_relax
}
\_def\_restorectable{\_ea\_restorectableA\_ctablelist\_relax}
\_def\_restorectableA#1#2\_relax{%
   \_ifx^#2^\_opwarning
      {You can't use \_noindent\restorectable without previous \_string\setctable}%
   \_else \_def\_ctablelist{#2}\_catcodetable#1\_relax \_currctable=#1\_relax \_fi
}
\_def\_ctablelist{.}

\_public \setctable \restorectable ;
```

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

more-macros.opm

```
\_def\_normalcatcodes {\_catcodetable\_optexcatcodes \_savecatcodetable0 \_catcodetable0 }
\_public \normalcatodes ;
```

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

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

more-macros.opm

```
\_def\_optdef#1[#2]{%
   \_def#1{\_opt={#2}\_isnextchar[{\_cs{_oA:\_string#1}}{\_cs{_oB:\_string#1}}}%
   \_sdef{_oA:\_string#1}[##1]{\_opt={##1}\_cs{_oB:\string#1\_nospaceafter}}%
   \_sdef{_oB:\_string#1\_nospaceafter}%
}
\_def\_nospaceafter#1{\_ea#1\_romannumeral-`\.}
\_newtoks\_opt

\_public \opt \optdef ;
```

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

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

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

more-macros.opm

```
\_def\_eoldef #1{\_def #1{\_begingroup \_catcode`\^^M=12 \_eoldefA #1}%
   \_ea\_def\_csname _\_csstring #1:M\_endcsname}
\_catcode`\^^M=12 %
\_def\_eoldefA #1#2^^M{\_endgroup\_csname _\_csstring #1:M\_endcsname{#2}}%
\_normalcatcodes %

\_eoldef\_skiptoeol#1{}
\_def\_bracedparam#1{\_ifcsname _\_csstring #1:M\_endcsname
    \_csname _\_csstring #1:M\_ea \_endcsname
    \_else \_csname __in\_csstring #1:M\_ea \_endcsname \_fi
}
```

```
\_public \eoldef \skiptoeol \bracedparam ;
```

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

more-macros.opm

```
\_catcode`!=3 \_catcode`?=3
\_def\_replstring #1#2#3{%  \replstring #1{stringA}{stringB}
   \_long\_def\_replacestringsA##1#2{\_def #1{##1}\_replacestringsB}%
   \_long\_def\_replacestringsB##1#2{\_ifx!##1\_relax \_else \_addto #1{#3##1}%
      \_ea\_replacestringsB\_fi}%
   \_ea\_replacestringsA #1?#2!#2%
   \_long\_def\_replacestringsA##1?{\_def #1{##1}}\_ea\_replacestringsA #1}
\_normalcatcodes

\_public \replstring ;
```

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

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

more-macros.opm

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

The \removespaces ⟨text with spaces ⟩{} expands to ⟨textwithoutspaces⟩.
The _ea\ignorept\the⟨dimen⟩ expands to a decimal number \the⟨dimen⟩ but without pt unit.
The \ignoreit⟨token⟩ just ignores the ⟨token⟩.

more-macros.opm

```
\_def\_removespaces #1 {\_isempty{#1}\_iffalse #1\_ea\_removespaces\_fi}
\_ea\_def \_ea\_ignorept \_ea#\_ea1\detokenize{pt}{#1}
\_def\_ignoreit#1{}

\public \removespaces \ignorept \ignoreit ;
```

You can use expandable \bp{⟨dimen⟩} convertor from TEX ⟨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} l S Q}
```

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

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

more-macros.opm

```
\_def\_decdigits{3} % digits after decimal point in \_bp and \_expr outputs.
\_def\_pttopb{%
   \_directlua{tex.print(string.format('\_pcent.\_decdigits f',
               token.scan_dimen()/65781.76))}%  pt to bp conversion
}
\def\_bp#1{\_ea\_pttopb\_dimexpr#1\_relax}
\def\_expr#1{\_directlua{tex.print(string.format('\_pcent.\_decdigits f',#1))}}

\_public \expr \bp ;
```

The pair _doc ... _cod is used for documenting macros and to printing the technical documentation of the OpTEX. The syntax is:

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

The ⟨documentation⟩ (and ⟨ignored text⟩ too) must be ⟨balanced text⟩. It means that you cannot document only the { but you must document the } too.

more-macros.opm

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

2.9 Plain TEX macros

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

plain-macros.opm

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

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

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

plain-macros.opm

```
\_def\_dospecials {\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\^^K\do\^^A\do\%\do\~}
\_chardef\_active = 13

\_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

```
\_def \_magstephalf{1095 }
\_def \_magstep#1{\_ifcase#1 1000\_or 1200\_or 1440\_or 1728\_or 2074\_or 2488\_fi\_space}
\_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

```
\_def\^^M{\ } % control <return> = control <space>
\_def\^^I{\ } % same for <tab>

\_def\lq{`} \def\rq{'}
\_def\lbrack{[} \_def\rbrack{]} % They are only public versions.
% \catcode`\^^L=\active \outer\def^^L{\par} % ascii form-feed is "\outer\par" % obsolete

\_let\_endgraf=\_par \_let\_endline=\_cr
\_public \endgraf \endline ;
```

Plain TEX classical \obeylines and \obeyspaces.

plain-macros.opm

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

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

plain-macros.opm

```
\_protected\_def\_thinspace {\_kern .16667em }
\_protected\_def\_negthinspace {\_kern-.16667em }
\_protected\_def\_enspace {\_kern.5em }
\_protected\_def\_enskip {\_hskip.5em\_relax}
\_protected\_def\_quad {\_hskip1em\_relax}
\_protected\_def\_qquad {\_hskip2em\_relax}
\_protected\_def\_smallskip {\_vskip\_smallskipamount}
\_protected\_def\_medskip {\_vskip\_medskipamount}
\_protected\_def\_bigskip {\_vskip\_bigskipamount}
\_def\_nointerlineskip {\_prevdepth=-1000pt }
\_def\_offinterlineskip {\_baselineskip=-1000pt \_lineskip=0pt \_lineskiplimit=\_maxdimen}

\_public \thinspace \negthinspace \enspace \enskip \quad \qquad \smallskip
   \medskip \bigskip \nointerlineskip \offinterlineskip ;

\_def\_topglue {\_nointerlineskip\_vglue-\_topskip\_vglue} % for top of page
\_def\_vglue {\_afterassignment\_vglA \_skip0=}
\_def\_vglA {\_par \_dimen0=\prevdepth \_hrule height0pt
```

```
  \_nobreak\_vskip\_skip0 \_prevdepth=\_dimen0 }
\_def\_hglue {\_afterassignment\_hglA \_skip0=}
\_def\_hglA {\_leavevmode \_count255=\_spacefactor \_vrule width0pt
  \_nobreak\_hskip\_skip0 \_spacefactor=\_count255 }
\_protected\def~{\penalty10000 \ } % tie
\_protected\_def\_slash {/\_penalty\_exhyphenpenalty} % a `/' that acts like a `-'

\_public \topglue \vglue \hglue \slash ;
```

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

plain-macros.opm

```
\_protected\_def \_break {\_penalty-10000 }
\_protected\_def \_nobreak {\_penalty10000 }
\_protected\_def \_allowbreak {\_penalty0 }
\_protected\_def \_filbreak {\_par\_vfil\_penalty-200\_vfilneg}
\_protected\_def \_goodbreak {\_par\_penalty-500 }
\_protected\_def \_eject {\_par\_break}
\_protected\_def \_supereject {\_par\_penalty-20000 }
\_protected\_def \_dosupereject {\_ifnum \_insertpenalties>0 % something is being held over
  \_line{}\_kern-\_topskip \_nobreak \_vfill \_supereject \_fi}
\_def \_removelastskip {\_ifdim\_lastskip=0pt \_else \_vskip-\_lastskip \_fi}
\_def \_smallbreak {\_par\_ifdim\_lastskip<\_smallskipamount
  \_removelastskip \_penalty-50 \_smallskip \_fi}
\_def \_medbreak {\_par\_ifdim\_lastskip<\_medskipamount
  \_removelastskip \_penalty-100 \_medskip \_fi}
\_def \_bigbreak {\_par\_ifdim\_lastskip<\_bigskipamount
  \_removelastskip \_penalty-200 \_bigskip \_fi}

\_public \break \nobreak \allowbreak \filbreak \goodbreak \eject \supereject \dosupereject
   \removelastskip \smallbreak \medbreak \bigbreak ;
```

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

plain-macros.opm

```
\_def \_line {\_hbox to\_hsize}
\_def \_leftline #1{\_line{#1\_hss}}
\_def \_rightline #1{\_line{\_hss#1}}
\_def \_centerline #1{\_line{\_hss#1\_hss}}
\_def \_rlap #1{\_hbox to0pt{#1\_hss}}
\_def \_llap #1{\_hbox to0pt{\_hss#1}}
\_def\_underbar #1{$\_setbox0=\_hbox{#1}\dp0=0pt \_math \_underline{\_box0}$}

\_public \line \leftline \rightline \centerline \rlap \llap \underbar ;
```

The \strutbox is declared as 10pt size dependent (like in plain TEX), but the macro _setbaselineskip (from fonts-opmac.opm) redefines it.

plain-macros.opm

```
\_newbox\_strutbox
\_setbox\_strutbox=\_hbox{\_vrule height8.5pt depth3.5pt width0pt}
\_def \_strut {\_relax\_ifmmode\_copy\_strutbox\_else\_unhcopy\_strutbox\_fi}

\_public \strutbox \strut ;
```

Alignment. \hidewidth \ialign \multispan.

plain-macros.opm

```
\_def \_hidewidth {\_hskip\_hideskip} % for alignment entries that can stick out
\_def \_ialign{\_everycr={}\_tabskip=\_zoskip \_halign} % initialized \halign
\_newcount\_mscount
\_def \_multispan #1{\_omit \_mscount=#1\_relax
   \_loop \_ifnum\_mscount>1 \_spanA \_repeat}
\_def \_spanA {\_span\_omit \_advance\_mscount by-1 }

\_public \hidewidth \ialign \multispan ;
```

Tabbing macros are omitted because they are obsolete.
Indentation and others. \textindent, \item, \itemitem, \narrower, \raggedright, \ttraggedright, \leavevmode.

plain-macros.opm

```
\_def \_hang {\_hangindent\_parindent}
\_def \_textindent #1{\_indent\_llap{#1\_enspace}\_ignorespaces}
```

```
\_def \_item {\_par\_hang\_textindent}
\_def \_itemitem {\_par\_indent \_hangindent2\_parindent \_textindent}
\_def \_narrower {\_advance\_leftskip\_parindent
   \_advance\_rightskip\_parindent}
\_def \_raggedright {\_rightskip=0pt plus2em
   \_spaceskip=.3333em \_xspaceskip=.5em\_relax}
\_def \_ttraggedright {\_tt \_rightskip=0pt plus2em\_relax} % for use with \tt only
\_def \_leavevmode {\_unhbox\_voidbox} % begins a paragraph, if necessary

\_public \hang \textindent \item \itemitem \narrower \raggedright \ttraggedright \leavevmode ;
```

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

plain-macros.opm

```
%\chardef\%=`\%
\_let\% = \_pcent  % more natural, can be used in lua codes.
\_chardef\&=`\&
\_chardef\#=`\#
\_chardef\$=`\$
\_chardef\ss="FF
\_chardef\ae="E6
\_chardef\oe="F7
\_chardef\o="F8
\_chardef\AE="C6
\_chardef\OE="D7
\_chardef\O="D8
\_chardef\i="11 \chardef\j="12 % dotless letters
\_chardef\aa="E5
\_chardef\AA="C5
\_chardef\S="9F
\_def\l{\_errmessage{\_usedirectly ł}}
\_def\L{\_errmessage{\_usedirectly Ł}}
\def\_{\_ifmmode \_kern.06em \_vbox{\hrule width.3em}\_else _\fi} % obsolete
\_def\dag{\_errmessage{\_usedirectly †}}
\_def\ddag{\_errmessage{\_usedirectly ‡}}
%\_def\copyright{\_errmessage{\_usedirectly ©}}
\_def\copyright{©} % << example, what to do
%\_def\Orb{\_mathhexbox20D} % obsolete (part of Copyright)
%\_def\P{\_mathhexbox27B}   % obsolete

\_def \_usedirectly #1{Load Unicoded font by \string\fontfam\space and use directly #1}
\_def \_mathhexbox #1#2#3{\_leavevmode \_hbox{$\_math \_mathchar"#1#2#3$}}
\_public \mathhexbox ;
```

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

plain-macros.opm

```
\_def \_oalign #1{\_leavevmode\_vtop{\_baselineskip=0pt \_lineskip=.25ex
   \_ialign{##\_crcr#1\_crcr}}}
\_def \_oalignA {\_lineskiplimit=0pt \_oalign}
\_def \_ooalign {\_lineskiplimit=-\_maxdimen \_oalign} % chars over each other
\_def \_shiftx #1{\_dimen0=#1\_kern\_ea\_ignorept \_the\_fontdimen1\_font
   \_dimen0 } % kern by #1 times the current slant
\_def \d #1{{\_oalignA{\_relax#1\_crcr\_hidewidth\_shiftx{-1ex}.\_hidewidth}}}
\_def \b #1{{\_oalignA{\_relax#1\_crcr\_hidewidth\_shiftx{-3ex}%
   \_vbox to.2ex{\_hbox{\_char\_macron}\_vss}\_hidewidth}}}
\_def \_c #1{{\_setbox0=\_hbox{#1}\_ifdim\_ht0=1ex\_accent\_cedilla #1%
   \_else\_ooalign{\_unhbox0\_crcr\_hidewidth\_cedilla\_hidewidth}\_fi}}
\_def\_dots{\_relax\_ifmmode\_ldots\_else$\_math\_ldots\_thinsk$\_fi}
\_public \oalign \ooalign \d \b \c \dots ;
```

The accents commands like \v, \., \H, etc. are not defined. Use the accented characters directly – it is best solution. But you can use the macro \oldaccents which defines accented macros.
Much more usable is to define these control sequences to other purposes.

plain-macros.opm

```
\def \_oldaccents {%
   \_def\`##1{{\_accent\_tgrave  ##1}}%
   \_def\'##1{{\_accent\_tacute  ##1}}%
```

```
   \_def\v##1{{\_accent\_caron  ##1}}%
   \_def\u##1{{\_accent\_tbreve  ##1}}%
   \_def\=##1{{\_accent\_macron ##1}}%
   \_def\^##1{{\_accent\_circumflex ##1}}%
   \_def\.##1{{\_accent\_dotaccent ##1}}%
   \_def\H##1{{\_accent\_hungarumlaut ##1}}%
   \_def\~##1{{\_accent\_ttilde  ##1}}%
   \_def\"##1{{\_accent\_dieresis ##1}}%
   \_def\r##1{{\_accent\_ring   ##1}}%
}
\_public \oldaccents ;

% ec-lmr encoding (will be changed after \fontfam macro):
\_chardef\_tgrave=0
\_chardef\_tacute=1
\_chardef\_circumflex=2
\_chardef\_ttilde=3
\_chardef\_dieresis=4
\_chardef\_hungarumlaut=5
\_chardef\_ring=6
\_chardef\_caron=7
\_chardef\_tbreve=8
\_chardef\_macron=9
\_chardef\_dotaccent=10
\_chardef\_cedilla=11

\_def \_uniaccents {% accents with Unicode
   \_chardef\_tgrave="0060
   \_chardef\_tacute="00B4
   \_chardef\_circumflex="005E
   \_chardef\_ttilde="02DC
   \_chardef\_dieresis="00A8
   \_chardef\_hungarumlaut="02DD
   \_chardef\_ring="02DA
   \_chardef\_caron="02C7
   \_chardef\_tbreve="02D8
   \_chardef\_macron="00AF
   \_chardef\_dotaccent="02D9
   \_chardef\_cedilla="00B8
   \_chardef\_ogonek="02DB
   \_let \_uniaccents=\_relax
}
```

The last part of plain TEX macros. \hrulefill, \dotfill, \rightarrowfill, \leftarrowfill, \magnification, \bye. Math macros are defined in the `math-macros.opm` file.

plain-macros.opm

```

\_def \_hrulefill {\_leaders\_hrule\_hfill}
\_def \_dotfill {\_cleaders\_hbox{$\_math \_mkern1.5mu.\_mkern1.5mu$}\_hfill}
\_def \_rightarrowfill {$\_math\_smash-\_mkern-7mu%
  \_cleaders\_hbox{$\_mkern-2mu\_smash-\_mkern-2mu$}\_hfill
  \_mkern-7mu\_mathord\_rightarrow$}
\_def \_leftarrowfill {$\_math\_mathord\_leftarrow\_mkern-7mu%
  \_cleaders\_hbox{$\_mkern-2mu\_smash-\_mkern-2mu$}\_hfill
  \_mkern-7mu\_smash-$}

\_public \hrulefill \dotfill \rightarrowfill \leftarrowfill ;

% \_downbracefil \_upbracefil will be re-defined when Unicode-math is used
\_mathchardef \_braceld="37A \_mathchardef \_bracerd="37B
\_mathchardef \_braceru="37C \_mathchardef \_braceru="37D
\_def \downbracefill {$\_math \_setbox0=\_hbox{$\_braceld$}%
  \_braceld \_leaders\_vrule height\_ht0 depth0pt \_hfill \_braceru
  \_braceru \_leaders\_vrule height\_ht0 depth0pt \_hfill \_bracerd$}
\_def \upbracefill {$\_math \_setbox0=\_hbox{$\_braceld$}%
  \_braceru \_leaders\_vrule height\_ht0 depth0pt \_hfill \_bracerd
  \_braceld \_leaders\_vrule height\_ht0 depth0pt \_hfill \_braceru$}

\_def \_magnification {\_afterassignment \_magA \_count255 }
```

```
\_def \_magA {\_mag=\_count255 \_truedimen\_hsize \_truedimen\_vsize
  \_dimen\_footins=8truein
}
% only for backward compatibility, but \margins macro is preferred.
\_public \magnification ;

\_def \_showhyphens #1{\_setbox0=\_vbox{\_parfillskip=0pt \_hsize=\_maxdimen \_tenrm
  \_pretolerance=-1 \tolerance=-1 \hbadness=0 \showboxdepth=0 \ #1}}

\_def \_bye {\_par \_vfill \_supereject \_end}
\_public \bye ;
```

2.10 Preloaded fonts for text mode

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

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

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

fonts-preload.opm

```
\_codedecl \tenrm {Latin Modern fonts (EC) preloaded <2020-01-23>} % loaded in format

% Only few text fonts are preloaded:

\_font\_tenrm=ec-lmr10   % roman text
\_font\_tenbf=ec-lmbx10  % boldface extended
\_font\_tenit=ec-lmri10  % text italic
\_font\_tenbi=ec-lmbxi10 % bold italic
\_font\_tentt=ec-lmtt10  % typewriter
\_tenrm

\_public \tenrm \tenbf \tenit \tenbi \tentt ;
```

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

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 initialization value in OpTEX is given by `\setfontsize{at10pt}`.

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

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

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

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

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

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

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

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

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

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

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

2.11.1 The `\fontdef` declarator

You can declare \⟨*newfont*⟩ by the `\fontdef` command.

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

This command runs ⟨*font modifiers*⟩ \⟨*variant-selector*⟩ in a group and sets the resulting current font as \⟨*newfont*⟩.

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

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

2.11.2 The `\fontlet` declarator

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

```
\fontlet \⟨newfont⟩ = \⟨fontswitch⟩ ⟨sizespec⟩
example:
\fontlet \bigfont = \_tenbf at15pt
```

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

Summary: you can declare font switches:

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

2.11.3 Optical sizes

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

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

2.11.4 Implementation notes

fonts-resize.opm

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

The `\setfontsize` {⟨*sizespec*⟩} saves the ⟨*sizespec*⟩ to the `\_sizespec` macro. The `\_optsize` value is calculated from the ⟨*sizespec*⟩. If the ⟨*sizepec*⟩ 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

```
\_newdimen \_optsize         \_optsize=10pt
\_newdimen \_defaultoptsize  \_defaultoptsize=10pt
\_newdimen\_lastmagsize

\_def\_setfontsize #1{%
   \_edef\_sizespec{#1}%
   \_ea \_setoptsize \_sizespec\_relax
   \_reloading
}
\_def\_setoptsize {\_isnextchar a{\_setoptsizeA}
                                {\_isnextchar m{\_setoptsizeC}{\_setoptsizeB}}}
\_def\_setoptsizeA at#1\_relax{\_optsize=#1\_relax\_lastmagsize=\_optsize}   % at<dimen>
\_def\_setoptsizeB scaled#1\_relax{\_optsize=\_defaultoptsize\_relax} % scaled<scalenum>
\_def\_setoptsizeC mag#1\_relax{%
   \_ifdim\_lastmagsize>0pt \_optsize=\_lastmagsize \_else \_optsize=\_pdffontsize\_font \_fi
   \_optsize=#1\_optsize
   \_lastmagsize=\_optsize
   \_edef\_sizespec{at\_the\_optsize}%
}
\_public \setfontsize \defaultoptsize ;
```

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

`\_font \_tenXX =` ⟨*fontname*⟩ `\_sizespec`

The ⟨*fontname*⟩ is generated by the `\fontname` TEX primitive where `\_rfontskipat` removes the `at`⟨*dimen*⟩ part of the `\_fontname` output. The ⟨*fontname*⟩ is generated differently in OTF mode, see `\_doresizeunifont` macro.
The `\_whatresize` is defined as ⟨*variant-name*⟩.

fonts-resize.opm

```
\_def\_resizefont#1#2{%
    \_edef\_whatresize{#1}%
    \_ifx \_fontselector \_undefined \_doresizefont#2%
    \_else \_ea \_doresizefont \_fontselector \_fi
    \_lastmagsize=0pt
    \_slet{_tryload#1}{_relax}%
}
\_def\_doresizetfmfont#1{\_logfont{#1}%
   \_ea\_font\_ea#1\_ea\_rfontskipat
      \_fontname \_cs{_ten\_whatresize} \_relax\_space \_sizespec \_relax
}
\_let\_doresizefont=\_doresizetfmfont
\_def\_logfont#1{} % default is no logging of used fonts
```

```
\def\_rfontskipat#1{\_ifx#1"\_ea\_rfskipatX \_else\_ea\_rfskipatN\_ea#1\_fi}
\def\_rfskipatX #1" #2\_relax{"\_whichtfm{#1}"}
\def\_rfskipatN #1 #2\_relax{\_whichtfm{#1}}
```

`\fontdef` ⟨*font switch*⟩`{`⟨*modifiers*⟩⟨*variant selector*⟩`}` opens group, runs ⟨*modifiers*⟩⟨*variant selector*⟩ (i.e. it runs `#2` parameter). The font switch `#1` saved in the `\_fontselector` macro is re-declared because the variant selector runs the `\_resizefont`. Now, we need to keep the current meaning of the font switch `#1` but we must leave the opened group. This is done by the `\_keepmeaning` macro.
`\fontlet` ⟨*font switch A*⟩ ⟨*font switch B*⟩ ⟨*size spec*⟩ does

`\font` ⟨*font switch A*⟩ `=` ⟨*fontname*⟩ ⟨*sizespec*⟩

The ⟨*fontname*⟩ is extracted using the primitive command `\_fontname` ⟨*font switch B*⟩.

fonts-resize.opm

```
\_def \_fontdef #1#2{\_begingroup
   \_ifx\_fontselector\_undefined \_def\_fontselector{#1}\_fi
   \_reloading #2%
   \_ea \_keepmeaning \_fontselector \_endgroup
}
\_def\_fontlet#1#2{\_ifx #2=\_ea\_fontlet \_ea#1\_else
  \_ea\_font\_ea#1\_ea\_rfontskipat\_fontname#2 \_relax\_space \_fi
}
\_def \_keepmeaning #1#2{\_global\_let\_keepmeaningdata=#1%
   #2\_let#1=\_keepmeaningdata \_global\_let\_keepmeaningdata=\_undefined
}
\_public \fontdef \fontlet ;
```

`\newcurrfontsize` ⟨*size spec*⟩ sets current font size to the ⟨*size spec*⟩ It is implemented by `\fontlet`. The font switch of the current font is extracted by `\_the\_font`. We must re-create the control sequence `\_the\_font` because its original meaning is set to "inaccessible" by TEX when `\font` primitive is started. `\resizethefont` is implemented by `\newcurrfontsize` using data from the `\_sizespec` macro.

fonts-resize.opm

```
\_def \_newcurrfontsize #1{% \newcurrfontsize{at25pt}
   \_edef\_tmp{\_ea\_csstring \_the\_font}%
   \_ea \_fontlet \_csname \_tmp\_ea\_endcsname \_the\_font \_space #1\_relax
   \_csname \_tmp\_endcsname
}
\_protected\_def \_resizethefont{\_newcurrfontsize\_sizespec}

 \_public \newcurrfontsize \resizethefont ;
```

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

fonts-resize.opm

```
\_def\_reloading{\_loadf{rm}\_tenrm \_loadf{bf}\_tenbf
   \_loadf{it}\_tenit \_loadf{bi}\_tenbi
}
\_def\_loadf#1#2{\sdef{_tryload#1}{\_ifmmode \_else \_resizefont{#1}#2\_fi}}
\_def\_tryloadtt{\_resizefont{tt}\_tentt}

\_let\_tryloadrm=\_relax
\_let\_tryloadbf=\_relax
\_let\_tryloadit=\_relax
\_let\_tryloadbi=\_relax
```

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

fonts-resize.opm

```
\_protected \_def \_currvar{\_cs{_currvar:\_ea \_csstring \_the\_font}}
\_sdef{_currvar:_tenrm}{\_rm}
\_sdef{_currvar:_tenbf}{\_bf}
\_sdef{_currvar:_tenit}{\_it}
\_sdef{_currvar:_tenbi}{\_bi}
\_sdef{_currvar:_tentt}{\_tt}
\_public \currvar ;
```

The _regtfm ⟨*font id*⟩ ⟨*optical size data*⟩ saves the ⟨*optical size data*⟩ concerned to ⟨*font id*⟩. The ⟨*optical size data*⟩ is in the form as show below in the code where \regtfm is used.
The _wichtfm ⟨*fontname*⟩ expands to the ⟨*fontname*⟩ or to the corrected ⟨*fontname*⟩ read from the ⟨*optical size data*⟩. It is used in the _rfontskipat macro and it is used in \fontlet macro. It means that each ⟨*fontname*⟩ generated by the \fontname primitive in the \fontlet macro is processed by the _whichtfm. The real ⟨*fontname*⟩ or corrected ⟨*fontname*⟩ (depending on the optical data does not exist or exist) is the output of the expansion before \font primitive takes this output as its parameter.

The implementation detail: The _⟨*font id*⟩:reg is defined as the ⟨*optical size data*⟩ and all control sequences _⟨*fontname*⟩:reg from this data line has the same meaning because of the _reversetfm macro. The _whichtfm expands this data line and apply _dowhichtfm. This macro select the right result from the data line by testing with the current _optsize value.

fonts-resize.opm

```
\_def\_regtfm #1 0 #2 *{\_ea\_def \_csname _#1:reg\_endcsname{#2 16380 \_relax}%
  \_def\_tmpa{#1}\_reversetfm #2 * %
}
\_def\_reversetfm #1 #2 {% we need this data for \_setmathfamily
   \_ea\_let\_csname _#1:reg\_ea\_endcsname
   \_csname _\_tmpa:reg\_endcsname
   \_if*#2\_else \_ea\_reversetfm \_fi
}
\_def\_whichtfm #1{%
   \_ifcsname _#1:reg\_endcsname
      \_ea\_ea\_ea \_dowhichtfm
      \_csname _#1:reg\_ea\_endcsname
   \_else
      #1%
   \_fi
}
\_def\_dowhichtfm #1 #2 {%
   \_ifdim\_optsize<#2pt #1\_ea\_ignoretfm\_else \_ea\_dowhichtfm
\_fi
}
\_def\_ignoretfm #1\_relax{}
```

Optical sizes data for preloaded 8bit Latin Modern fonts:

fonts-resize.opm

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

\_setfontsize {at10pt} % default font size
```

2.12 The Font Selection System

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

2.12.1 Terminology

We distinguish between

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

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

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

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

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

2.12.2 Font families, selecting fonts

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

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

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

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

⟨*family selector*⟩ ⟨*font modifiers*⟩ ⟨*variant selector*⟩

For example:

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

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

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

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

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

2.12.3 Math Fonts

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

If you want to use your specially selected Unicode-math font then use `\loadmath {[`⟨*font_file*⟩`]}` or `\loadmath {`⟨*font_name*⟩`}` before first `\fontfam` is used.

2.12.4 Declaring font commands

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.12.5 Modifying font features

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

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

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

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

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

2.12.6 Special font modifiers

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

```
\setfontsize{⟨sizespec⟩}    % sets the font size
\setff{⟨font feature⟩}      % adds the font feature
\setfontcolor{⟨color⟩}     % sets font color
\setletterspace{⟨number⟩} % sets letter spacing
\setwordspace{⟨scaling⟩}    % modifies word spacing
```

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

`\setfontcolor`

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

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

`\setletterspace`

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

`\setwordspace`

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

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

```
\setff{embolden=1.5}\rm % font is bolder because outline has nonzero width
\setff{slant=0.2}\rm    % font is slanted by a linear transformation
\setff{extend=1.2}\rm   % font is extended by a linear transformation.
```

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

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

2.12.7 How to create the font family file

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

f-heros.opm

```
\_famdecl [Heros] \Heros {TeX Gyre Heros fonts based on Helvetica}
     {\caps \cond} {\rm \bf \it \bi} {FiraMath}
     {[texgyreheros-regular]}
     {\_def\_fontnamegen{[texgyreheros\_condV-\_currV]:\_capsV\_fontfeatures}}

\_wlog{\_detokenize{%
Modifiers:^^J
 \caps ..... caps & small caps^^J
 \cond ..... condensed variants^^J
}}

\_moddef \resetmod {\_fsetV caps={},cond={} \_fvars regular bold italic bolditalic }
\_moddef \caps     {\_fsetV caps=+smcp;+onum; }
\_moddef \nocaps   {\_fsetV caps={} }
\_moddef \cond     {\_fsetV cond=cn }
\_moddef \nocond   {\_fsetV cond={} }

\_initfontfamily % new font family must be initialized

\_loadmath {[FiraMath-Regular]}
```

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

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

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

  This writes information about font family at the terminal and prevents loading such file twice. Moreover, it probes existence of ⟨*font-for-testing*⟩ in your system. If it doesn't exist, the file loading is skipped with a warning on the terminal. The `\_ifexistfam` macro returns false in such case. The `\_fontnamegen` macro must be defined in the last parameter of the `\_famdecl`. More about it is documented below.
- You can use `\_wlog{\_detokenize{...` to write aditional information into log file.
- You can declare optical sizes using `\_regoptsizes` if there are more font files with different optical sizes (like in Latin Modern). See `f-lmfonts.ofm` file for more information about this special feature.
- Declare font modifiers using `\_moddef` if they are present. The `\resetmod` must be declared in each font family.
- Check if all your declared modifiers does not produce any space in horizontal mode. For example check: `X\caps Y`, the letters `XY` must be printed without any space.
- Optionally, declare new variants by the `\famvardef` macro.
- Run `\_initfontfamily` in order to start the family.
- If math font should be loaded, use `\_loadmath{`⟨*math font*⟩`}`.

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

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

Note that the extended `\font` syntax `\font\`⟨*selector*⟩ `{`⟨*font name*⟩`:`⟨*font features*⟩`}` ⟨*size spec.*⟩ or `\font\`⟨*selector*⟩ `{[`⟨*font file name*⟩`]:`⟨*font features*⟩`}` ⟨*size spec.*⟩ is expected here.

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

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

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

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

The following auxiliary macros are used here:

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

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

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

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

is processed.

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

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

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

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

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

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

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

The \moddef macro does more things than simple _def:

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

The \famvardef macro has the same features.

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

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

The font context consists from

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

The _initfontfamily must be run after modifers decaration. It sets _let_resetmod=\resetmod and runs the \⟨*Familyselector*⟩. Finally, it runs _rm, so first font from new family is loaded and it is ready to use it.

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

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

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

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

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

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

2.12.8 How to write the font family file with optical sizes

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

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

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

In general:

`\_regoptsizes` ⟨*internal-template*⟩ ⟨*general-ouput-template*⟩ ⟨*resizing-data*⟩

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 $11.1 < \texttt{\_optsize} \leq 15$ then 12 is substituted instead question mark. The ⟨*resizing-data*⟩ virtually begins with zero `<0`, but it is not explicitly written. The right part of ⟨*resizing-data*⟩ must be terminated by `<*` which means "less than infinity".

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

We are using a special macro `\_LMregfont` in `f-lmfonts.opm`. It sets the file names to lowercase and enables to use a shortcasts isntead real ⟨*resizing-data*⟩. There are shortcats `\_regoptFS`, `\_regoptT`, etc. here. The collection of ⟨*internal-templates*⟩ are declared, each of them covers a collection of real file names.

The `\_optfontalias {`⟨*new-template*⟩`} {`⟨*internal-template*⟩`}` declares ⟨*new-template*⟩ with the same meaning as previously declared ⟨*internal-template*⟩.

The `\_optname` macro can be used even if no otical 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 ⟨*internal-templates*⟩ and do an alias from each ⟨*internal-template*⟩ to ⟨*real-font-name*⟩. 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 olique}
\fontalias crea-a-bolditalic    {Creative Bold plus italic}
\fontalias crea-b-regular       {Creative Regular subfam}
\fontalias crea-b-bold          {Creative subfam bold}
\fontalias crea-b-italic        {Creative-subfam Oblique}
\fontalias crea-b-bolditalic    {Creative Bold subfam Oblique}
```

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

Once you have prepared a font family file with the name `f-`⟨*famname*⟩`.opm` and TEX is able to see it in your filesystem then you can type `\fontfam[`⟨*famname*⟩`]` and the file is read, so the information about 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 gives more tolerance: you can write uppercase letters and spaces here. The spaces are ignored and letters are converted to lowercase. For example `\fontfam [LM Fonts]` is equivalent to

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

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

The information from the user's `fams-local.opm` file has precedence. For example `fams-ini.opm` declares aliases Times→Termes etc. If you have original Times purchased from Adobe then you can register your declaration about Times family in `fams-local.opm`. When an user write `\fontfam[Times]` then orginal Times (no Termes) is used in such case.

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

fams-ini.opm

```
% Version <2020-02-28>. Loaded in format and secondly on demand by \fontfam[catalog]

\_famtext {Special name for printing a catalogue:}

\_faminfo [Catalogue] {Catalogue of all registered font families} {fonts-catalog} {}
\_famalias [Catalog]

\_famtext {Computer Modern like family:}

\_faminfo [Latin Modern] {TeX Gyre fonts based on Coputer Modern} {f-lmfonts}
   { -,\nbold,\sans,\sans\nbold,\slant,\ttset,\ttset\slant,\ttset\caps,%
       \ttprop,\ttprop\bolder,\quotset: {\rm\bf\it\bi}
       \caps: {\rm\it}
       \ttlight,\ttcond,\dunhill: {\rm\it} \upital: {\rm} }
\_famalias [LMfonts] \_famalias [Latin Modern Fonts]

\_famtext {TeX Gyre fonts based o Adobe 35:}

\_faminfo [Termes] {TeX Gyre Termes fonts based on Times} {f-termes}
   { -,\caps: {\rm\bf\it\bi} }
\_famalias [Times]

\_faminfo [Heros] {TeX Gyre Heros fonts based on Helvetica} {f-heros}
   { -,\caps,\cond,\caps\cond: {\rm\bf\it\bi} }
\_famalias [Helvetica]
```

... etc.

The `\_faminfo` commad has the syntax:

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

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

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

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

2.12.10 Implementation of the Font Selection System

fonts-select.opm

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

The \initunifonts initializes extended \font primitive (to be able to load Unicode fonts). Unfortunately, this part of OpTEX depends on LATEX lua codes ltluatex.lua and luaotfload-main.lua. And this code need to be declared a control sequence \e@alloc@attribute@count by \countdef primitive. Moreover, the \initunifont switches with the _doresizefont macro to OTF mode which is represented by the macro _doresizeunifont. This mode includes a fallback to TFM mode if _fontnamegen is not defined. Finally, the \initunifnt sets itself to relax because we need not to do this work twice.

fonts-select.opm

```
\_def\_initunifonts {%
   \_ea\_newcount \_csname e@alloc@attribute@count\_endcsname
   \_global \_csname e@alloc@attribute@count\_endcsname=-1
   \_directlua{%
      require("ltluatex")
      require('luaotfload-main') local _void = luaotfload.main ()
   }%
   \_gdef\_rfskipatX ##1" ##2\_relax{"##1"}%
   \_global\_let \_doresizefont=\_doresizeunifont
   \_gdef\_tryloadtt {\_fontdef\_tentt{\def\_fontnamegen{[lmmono10-regular]}\_rm}}%
   \_global\_let \_initunifonts=\_relax % we need not to do this work twice
   \_global\_let \initunifonts=\_relax
}
\_gdef\_doresizeunifont #1{\_logfont{#1}%
   \_ifx\_fontnamegen\_undefined \_doresizetfmfont#1\_else
       \_font#1={\_fontnamegen} \_sizespec \_relax \_setwsp#1\_relax
   \_fi
}
\_public \initunifonts ;
```

The _famdecl [⟨*Family Name*⟩] \⟨*Famselector*⟩ {⟨*comment*⟩} {⟨*modifiers*⟩} {⟨*variants*⟩} {⟨*math*⟩} {⟨*font for testing*⟩} {\def_fontnamegen{⟨*data*⟩}} runs \initunifonts, then checks if \⟨*Famselector*⟩ is defined. If it is true, then closes the file by \endinput. Else it defines \⟨*Famselector*⟩ and saves it to the _mainfamcommand macro because the _initfontfamily needs it. The _currfamily is set to the ⟨*Famselector*⟩ because the following \moddef commands need to be in the right font family context. The _currfamily is set to the ⟨*Famselector*⟩ by the \⟨*Famselector*⟩ too, because \⟨*Famselector*⟩ must set the right faily context. The font family context is given by the current _currfamily value and by the actual meaning of the _fontnamegen macro.

fonts-select.opm

```
\_def\_famdecl [#1]#2#3#4#5#6#7#8{%
   \_initunifonts \_uniaccents
   \_ifx #2\_undefined
      \_isfont{#7}\_iffalse
         \_opwarning{Family [#1] skipped, font "#7" not found}\_ea\_ea\_ea\_endinput \_else
         \_edef\_currfamily {\_csstring #2}%
         \_def\_mainfamcommand{#2}\_def\_mathfaminfo{#6}%
         \_protected\_edef#2{\_def\_noexpand\_currfamily{\_csstring #2}\_unexpanded{#8\_resetmod}}%
         \_wterm {FONT: [#1] -- \string#2 \_detokenize{(#3)^^J mods:{#4} vars:{#5} math:{#6}}}%
      \_fi
   \_else \_ea #2\_ea\_endinput \_fi
}
\_def\_initfontfamily{%
   \_mainfamcommand \_reloading \_rm
}
```

_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

```
\_def\_regoptsizes #1 #2?#3 #4*{\_sdef{_oz:#1}{#2?#3 #4* }}
\_def\_optname #1{\_ifcsname _oz:#1\_endcsname
   \_ea\_ea\_ea \_optnameA \_csname _oz:#1\_ea\_endcsname
   \_else \_failedoptname{#1}\_fi
}
\_def\_failedoptname #1{optname-fails:(#1)}
\_def\_optnameA #1?#2 #3 <#4 {\_ifx*#4#1#3#2\_else
   \_ifdim\_optsize<#4pt #1#3#2\_optnameC
```

```
    \_else \_afterfifi \_optnameA #1?#2 \_fi\_fi
}
\_def\_optnameC #1* {\_fi\_fi}
\_def\_afterfifi #1\_fi\_fi{\_fi\_fi #1}
\_def\_optfontalias #1#2{\_slet{_oz:#1}{_oz:#2}}
```

_fvars ⟨*rm-template*⟩ ⟨*bf-template*⟩ ⟨*it-template*⟩ ⟨*bi-template*⟩ saves data for usage by the _currV macro. If a template is only dot then previous template is used (it can be used if the font family doesn't dispose with all standard variants).
_currV expands to a template declared by _fvars depending on the ⟨*variant name*⟩. Usable only of standard four variants. Next variants can be declared by the \famvardef macro.
fset ⟨*key*⟩=⟨*value*⟩,...,⟨*key*⟩=⟨*value*⟩ expands to \def⟨*key*⟩V{⟨*value*⟩} in the loop.
_onlyif ⟨*key*⟩=⟨*value-a*⟩,⟨*value-b*⟩...,⟨*value-z*⟩: {⟨*what*⟩} runs ⟨*what*⟩ only if the _⟨*key*⟩V is defined as ⟨*value-a*⟩ or ⟨*value-b*⟩ or ... or ⟨*value-z*⟩.

fonts-select.opm

```
\_def\_fvars #1 #2 #3 #4 {%
   \_sdef{_fvar:rm}{#1}%
   \_sdef{_fvar:bf}{#2}%
   \_ifx.#2\_slet{_fvar:bf}{_fvar:rm}\_fi
   \_sdef{_fvar:it}{#3}%
   \_ifx.#3\_slet{_fvar:it}{_fvar:rm}\_fi
   \_sdef{_fvar:bi}{#4}%
   \_ifx.#4\_slet{_fvar:bi}{_fvar:it}\_fi
}
\_def\_currV{\_cs{_fvar:\_whatresize}}
\_def\_V{ }
\_def \_fsetV #1 {\_fsetVa #1,=,}
\_def \_fsetVa #1=#2,{\_isempty{#1}\_iffalse
   \_ifx,#1\_else\_sdef{_#1V}{#2}\_ea\_ea\_ea\_fsetVa\_fi\_fi
}
\_def \_onlyif #1=#2:#3{%
    \_edef\_act{\_noexpand\_isinlist{,#2,}{,\_cs{_#1V},}}\_act
    \_iftrue #3\_fi
}
```

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

When user declare a prefixed variant of the \⟨*modifier*⟩ then unprefixed modifier name is used in internal macros, this is reason why we are using the _remifirstunderscore_tmp (where _tmp expands to _⟨*something*⟩ or to ⟨*something*⟩. The _remifirstunderscore redefines _tmp in the way that it expands only to ⟨*something*⟩ without the first _.

fonts-select.opm

```
\_def \_moddef #1#2{\_edef\_tmp{\_csstring#1}\_remfirstunderscore\_tmp
   \_sdef{_f:\_currfamily:\_tmp}{#2\_reloading}%
   \_protected \_edef #1{\noexpand\_famdepend\noexpand#1{_f:\noexpand\_currfamily:\_tmp}}%
   \_ea \_ifx \_csname\_tmp\_endcsname #1\_else
      \_ea \_public \_csname\_tmp\_endcsname ;\_fi
}
\_def\_remfirstunderscore#1{\_ea\_remfirstunderscoreA#1\_relax#1}
\_def\_remfirstunderscoreA#1#2\_relax#3{\_if _#1\_def#3{#2}\_fi}

\_protected \_def\_resetmod {\_cs{_f:\_currfamily:resetmod}} % private variant of \resetmod
\_def\_currfamily{} % default current family is empty

\_def\_famdepend#1#2{\_ifcsname#2\_endcsname \_csname#2\_ea\_endcsname \_else
   \_opwarning{\string#1 is undeclared in current family "\_currfamily", ignored}\_fi
}
\_public \moddef ;
```

The \famvardef \⟨*XX*⟩ {⟨*data*⟩} uses analogical trick like \moddef with the _famdepend macro. The auxiliary _famvardefA \⟨*XX*⟩ _ten⟨*XX*⟩ _tryload⟨*XX*⟩ {⟨*data*⟩} is used. It does:

- \protected\def \⟨*XX*⟩ {_famdepend \⟨*XX*⟩ {_f:_currfamily:⟨*XX*⟩}},
- \def _f:⟨*current family*⟩:⟨*XX*⟩ {_tryload⟨*XX*⟩_ten⟨*XX*⟩} keeps family dependent definition,

- `\def \_tryload⟨XX⟩ {\fontdef \_ten⟨XX⟩ {⟨data⟩}}` loads actually the font `\_ten⟨XX⟩`,
- `\def \_currvar:_ten⟨XX⟩ {\⟨XX⟩}` in ordef to the `\currvar` macro work correctly.

fonts-select.opm

```
\_def\_famvardef#1{\_edef\_tmp{\_csstring#1}\_remfirstunderscore\_tmp
   \_ea\_famvardefA \_ea#1\_csname _ten\_tmp\_ea\_endcsname
      \_csname _tryload:\_tmp\_endcsname
}
\_def\_famvardefA #1#2#3#4{% #1=\_XX #2=\_tenXX #3=\_tryloadXX #4=data
   \_isinlist{.\_rm\_bf\_it\_bi\currvar\_currvar}#1\iftrue
      \_opwarning{\string\famvardef:
          You cannot re-declare private standard variant selector \string#1}%
   \_else
      \_protected\_edef #1{\noexpand\_famdepend\noexpand#1{_f:\noexpand\_currfamily:\_tmp}}%
      \_sdef{_f:\_currfamily:\_tmp}{#3#2}%
      \_def#3{\_fontdef#2{#4}}%
      \_ifx#1\tt \addto#1{\_fam\_ttfam}\fi
      \_sdef{_currvar:\csstring#2}{#1}%
   \_fi
}
\_public \famvardef ;
```

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

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

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

fonts-select.opm

```
\_def\_fontfam[#1]{%
   \_lowercase{\_edef\_famname{\_ea\_removespaces #1 {} }}%
   \_isfile {f-\_famname.opm}\_iftrue \_opinput {f-\_famname.opm}%
   \_else
      \_tryloadfamslocal
      \_edef\_famfile{\_trycs{_famf:\_famname}{}}%
      \_ifx\_famfile\_empty \_listfamnames
      \_else \_opinput {\_famfile.opm}%
   \_fi\_fi
}
\_def\_tryloadfamslocal{%
   \_isfile {fams-local.opm}\_iftrue
      \_opinput {fams-local.opm}
   \_fi
   \_let \_tryloadfamslocal=\_relax  % need not to load fams-local.opm twice
}
\_def\_listfamnames {%
   \_wterm{===== List of font families ======}
   \_begingroup
      \_let\_famtext=\_wterm
      \_def\_faminfo [##1]##2##3##4{%
          \_wterm{ \_space\_noexpand\fontfam [##1] -- ##2}%
      \_let\_famalias=\_famaliasA}%
      \_opinput {fams-ini.opm}
      \_isfile {fams-local.opm}\_iftrue \_opinput {fams-local.opm}\_fi
      \_message{^^J}%
   \_endgroup
}
\_def\_famaliasA{\_message{ \_space\_space\_space\_space -- alias:}
   \_def\_famalias[##1]{\_message{[##1]}}\_famalias
}
\_public \fontfam ;
```

When the `fams-ini.opm` or `fams-loca.opm` files are read then we need to save ony 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 infrormation is used and printed.
`\_famtext` does nothing or prints the text on the terminal.
`\_faminfo` `[`⟨*Family Name*⟩`]` `{`⟨*comments*⟩`}` `{`⟨*file-name*⟩`}` `{`⟨*mod-plus-vars*⟩`}` does `\def \_famf:`⟨*familyname*⟩ `{`⟨*file-name*⟩`}` or prints information on the terminal.
`\_famalias` `[`⟨*Family Alias*⟩`]` does `\def \_famf:`⟨*familyalias*⟩ `{`⟨*file-name*⟩`}` where ⟨*file-name*⟩ is stored from the previous `\_faminfo` command. Or prints information on the terminal.

fonts-select.opm

```
\_def\_famtext #1{}
\_def\_faminfo [#1]#2#3#4{%
   \_lowercase{\_edef\_tmp{\_ea\_removespaces #1 {} }}%
   \_sdef{_famf:\_tmp}{#3}%
   \_def\_famfile{#3}%
}
\_def\_famalias [#1]{%
   \_lowercase{\_edef\_famname{\_ea\_removespaces #1 {} }}%
   \_sdef{_famf:\_famname\_ea}\_ea{\_famfile}%
}
\_input fams-ini.opm
\_let\_famfile=\_undefined
```

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

fonts-select.opm

```
\_newtoks \_catalogsample
\_newtoks \_catalogmathsample
\_newtoks \_catalogonly
\_newtoks \_catalogexclude
\_catalogsample={ABCDabcd Qsty fi fl áéíóúüů řžč ÁÉÍÓÚ ŘŽČ 0123456789}

\_public \catalogonly \catalogexclude \catalogsample \catalogmathsample ;
```

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

fonts-select.opm

```
\_def \_defaultfontfeatures {+tlig;}
\_def \_setff #1{%
   \_ifx^#1^\_let \_fontfeatures=\_defaultfontfeatures
   \_else \_edef\_fontfeatures{\_fontfeatures #1;}\_fi
   \_reloading
}
\_setff {} % default font features: +tlig;
\_def\_removefeature #1{%
    \_isinlist\_fontfeatures{#1}\_iftrue
        \_def\_tmp ##1#1##2;##3\_relax{\_def\_fontfeatures{##1##3}}%
        \_ea \_tmp \_fontfeatures \_relax
    \_fi
}
\_public \setff ;
```

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

fonts-select.opm

```
\_def\_savedfontcolor{}
\_def\_savedletterspace{}
\_def\_savedwsp{}

\_def \_setfontcolor #1{\_removefeature{color=}%
    \_edef\_tmp{\_calculatefontcolor{#1}}%
    \_ifx\_tmp\_empty \_else \_edef\_fontfeatures{\_fontfeatures color=\_tmp;}\_fi
```

```
    \_reloading
}
\_def \_setletterspace #1{\_removefeature{letterspace=}%
    \_if^#1^\_else \_edef\_fontfeatures{\_fontfeatures letterspace=#1;}\_fi
    \_reloading
}
\_def \_setwordspace #1{%
   \_if^#1^\_def\_setwsp##1{}\_removefeature{+Ws}%
   \_else \def\_setwsp{\_setwspA{#1}}\_setff{+Ws}\_fi
   \_reloading
}
\_def\_setwsp #1{}
\_def\_setwspA #1#2{\_fontdimen2#2=#1\_fontdimen2#2%
   \_fontdimen3#2=#1\_fontdimen3#2\_fontdimen4#2=#1\_fontdimen4#2}

\_def\_calculatefontcolor#1{\_trycs{_fc:#1}{#1}} % you can define more smart macro ...
\_sdef{_fc:red}{FF0000FF}    \_sdef{_fc:green}{00FF00FF} \_sdef{_fc:blue}{0000FFFF}
\_sdef{_fc:yellow}{FFFF00FF} \_sdef{_fc:cyan}{00FFFFFF}  \_sdef{_fc:magenta}{FF00FFFF}
\_sdef{_fc:white}{FFFFFFFF}  \_sdef{_fc:grey}{00000080}  \_sdef{_fc:lgrey}{00000025}
\_sdef{_fc:black}{} % ... you can declare more colors...

\_public \setfontcolor \setletterspace \setwordspace ;
```

2.13 Preloaded fonts for math mode

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

On the other hand, when `\fontfam` macro is used in the document then text font family and appropriate math family is loaded with Unicoded fonts, i.e. Unicoded-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.
```

math-preload.opm

```
\_codedecl \normalmath {Math fonts CM + AMS preloaded <2020-04-14>} % preloaded in format
```

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

math-preload.opm

```
\_def\_normalmath{%
  \_loadmathfamily 0 cmr  % CM Roman
  \_loadmathfamily 1 cmmi % CM Math Italic
  \_loadmathfamily 2 cmsy % CM Standard symbols
  \_loadmathfamily 3 cmex % CM extra symbols
  \_loadmathfamily 4 msam % AMS symbols A
  \_loadmathfamily 5 msbm % AMS symbols B
  \_loadmathfamily 6 rsfs % script
  \_loadmathfamily 7 eufm % fractur
  \_loadmathfamily 8 bfsans % sans serif bold
  \_loadmathfamily 9 bisans % sans serif bold slanted (for vectors)
  \_setmathfamily 10 \_tentt
  \_setmathfamily 11 \_tenit
  \_setmathdimens
}
\def\_boldmath{%
  \_loadmathfamily 0 cmbx  % CM Roman Bold Extended
  \_loadmathfamily 1 cmmib % CM Math Italic Bold
  \_loadmathfamily 2 cmbsy % CM Standard symbols Bold
  \_loadmathfamily 3 cmexb % CM extra symbols Bold
  \_loadmathfamily 4 msam  % AMS symbols A (bold not available?)
  \_loadmathfamily 5 msbm  % AMS symbols B (bold not available?)
  \_loadmathfamily 6 rsfs  % script (bold not available?)
  \_loadmathfamily 7 eufb  % fractur bold
```

```
  \_loadmathfamily 8 bbfsans % sans serif extra bold
  \_loadmathfamily 9 bbisans % sans serif extra bold slanted (for vectors)
  \_setmathfamily 10 \_tentt
  \_setmathfamily 11 \_tenbi
  \_setmathdimens
}
\count18=11   % families declared by \newfam are 12, 13, ...

\_def \normalmath {\_normalmath}  \_def\boldmath {\_boldmath}
```

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

math-preload.opm

```
\_chardef\_bffam = 8
\_chardef\_bifam = 9
\_chardef\_ttfam = 10
\_chardef\_itfam = 11

\_protected\_def \_rm {\_tryloadrm \_tenrm \_fam0 }
\_protected\_def \_bf {\_tryloadbf \_tenbf \_fam\_bffam}
\_protected\_def \_it {\_tryloadit \_tenit \_fam\_itfam}
\_protected\_def \_bi {\_tryloadbi \_tenbi \_fam\_bifam}
\_protected\_def \_tt {\_tryloadtt \_tentt \_fam\_ttfam}

\_protected\_def \_mit    {\_fam1 }
\_protected\_def \_cal    {\_fam2 }
\_protected\_def \_bbchar {\_fam5 }  % double stroked letters
\_protected\_def \_frak   {\_fam7 }  % fraktur
\_protected\_def \_script {\_fam6 }  % more extensive script than \cal

\_public \rm \bf \it \bi \tt \mit \cal \bbchar \frak \script ;
```

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

math-preload.opm

```
%% CM math fonts, optical sizes:

\_regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
                cmmi10 11.1 cmmi12 *
\_regtfm cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
\_regtfm cmtex 0 cstex8 8.5 cstex9 9.5 cstex10 *
\_regtfm cmsy 0 cmsy5 5.5 cmsy6 6.5 cmsy7 7.5 cmsy8 8.5 cmsy9 9.5 cmsy10 *
\_regtfm cmbsy 0 cmbsy5 5.5 cmbsy6 6.5 cmbsy7 7.5 cmbsy8 8.5 cmbsy9 9.5 cmbsy10 *
\_regtfm cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
\_regtfm cmexb 0 cmexb10 *

\_regtfm cmr  0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
                cmr10 11.1 cmr12 15 cmr17 *
\_regtfm cmbx 0 cmbx5 5.5 cmbx6 6.5 cmbx7 7.5 cmbx8 8.5 cmbx9 9.5
                cmbx10 11.1 cmbx12 *
\_regtfm cmti 0 cmti7 7.5 cmti8 8.5 cmti9 9.5 cmti10 11.1 cmti12 *
\_regtfm cmtt 0 cmtt10 11.1 cmtt12 *

%% AMS math fonts, optical sizes:

\_regtfm msam 0 msam5 5.5 msam6 6.5 msam7 7.5 msam8 8.5 msam9 9.5 msam10 *
\_regtfm msbm 0 msbm5 5.5 msbm6 6.5 msbm7 7.5 msbm8 8.5 msbm9 9.5 msbm10 *

%% fraktur, rsfs, optical sizes:

\_regtfm eufm 0 eufm5 5.5 eufm6 6.5 eufm7 7.5 eufm8 8.5 eufm9 9.5 eufm10 *
\_regtfm eufb 0 eufb5 5.5 eufb6 6.5 eufb7 7.5 eufb8 8.5 eufb9 9.5 eufb10 *
\_regtfm rsfs 0 rsfs5 6 rsfs7 8.5 rsfs10 *

%% bf and bi sansserif math alternatives:

\_regtfm bfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
             8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
\_regtfm bisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
```

```
            8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *
\_regtfm bbfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
            8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
\_regtfm bbisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
            8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *
```

`\_loadmathfamily` ⟨*number*⟩ ⟨*font*⟩ loads one math family, i. e. the triple of fonts in the text size, script size and script-script size. The ⟨*font*⟩ is ⟨*font-id*⟩ used in the `\_regtfm` parameter or the real TFM name. The family is saved as `\fam`⟨*number*⟩.
`\_setmathfamily` ⟨*number*⟩ `\`⟨*font-switch*⟩ loads one math family like `\_loadmathfamily` does it. But the second parameter is a `\`⟨*font-switch*⟩ declared previously by the `\font` primitive.
The font family is loaded at `\_sizemtext`, `\_sizemscript` and `\_sizemsscript` sizes. These sizes are set by the `\setmathsizes [`⟨*text-size*⟩`/`⟨*script-size*⟩`/`⟨*scriptscript-size*⟩`]` macro. These parameters are given in the `\ptmunit` unit, it is set to 1`\ptunit` and it is set to 1 pt by defaut.
`\_corrmsizes` should be used in the `\normalmath` and `\boldmath` macros if you need a size correction when a selected math family is loaded. It is similar as ex-height correction but for math fonts.

math-preload.opm

```
\_def\_corrmsizes{\_ptmunit=1\_ptunit\_relax} % for corrections of sizes in diferent fomts

\_def\_loadmathfamily #1 #2 {\_chardef\_tmp#1\_corrmsizes
  \_edef\_optsizesave{\_the\_optsize}%
  \_optsize=\_sizemtext    \_font\_mF=\_whichtfm{#2} at\_optsize \_textfont#1=\_mF
  \_optsize=\_sizemscript  \_font\_mF=\_whichtfm{#2} at\_optsize \_scriptfont#1=\_mF
  \_optsize=\_sizemsscript \_font\_mF=\_whichtfm{#2} at\_optsize \_scriptscriptfont#1=\_mF
  \_optsize=\_optsizesave \_relax
}
\_def\_setmathfamily #1 #2{\_let\_mF=#2\_chardef\_tmp#1\_corrmsizes
  \_edef\_optsizesave{\_the\_optsize}%
  \_optsize=\_sizemtext    \_fontlet#2=#2 at\_optsize \_textfont#1=#2%
  \_optsize=\_sizemscript  \_fontlet#2=#2 at\_optsize \_scriptfont#1=#2%
  \_optsize=\_sizemsscript \_fontlet#2=#2 at\_optsize \_scriptscriptfont#1=#2%
  \_optsize=\_optsizesave \_let#2=\_mF
}
\_def\_setmathsizes[#1/#2/#3]{%
   \_def\_sizemtext{#1\_ptmunit}\_def\_sizemscript{#2\_ptmunit}%
   \_def\_sizemsscript{#3\_ptmunit}%
}
\_newdimen\_ptunit   \_ptunit=1pt
\_newdimen\_ptmunit  \_ptmunit=1\_ptunit

\_public \setmathsizes \ptunit \ptmunit ;
```

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

math-preload.opm

```
\_def\_setmathdimens{% PlainTeX sets these dimens for 10pt size only:
  \_delimitershortfall=0.5\_fontdimen6\_textfont3
  \_nulldelimiterspace=0.12\_fontdimen6\_textfont3
  \_scriptspace=0.05\_fontdimen6\_textfont3
  \_skewchar\_textfont1=127 \_skewchar\_scriptfont1=127
  \_skewchar\_scriptscriptfont1=127
  \_skewchar\_textfont2=48  \_skewchar\_scriptfont2=48
  \_skewchar\_scriptscriptfont2=48
  \_skewchar\_textfont6=127 \_skewchar\_scriptfont6=127
  \_skewchar\_scriptscriptfont6=127
}
```

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

math-preload.opm

```
\_suppressfontnotfounderror=1
\_setmathsizes[10/7/5]\_normalmath
\_suppressfontnotfounderror=0
```

2.14 Math macros

math-macros.opm

```
\_codedecl \sin {Math macros plus mathchardefs <2020-03-14>} % preloaded in format
```

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

There is a problem: The `x_n` is tokenized as `x`, `_`, `n` and it works without problem. 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 our own control sequence in the form \⟨*word*⟩_ or \⟨*word*⟩_⟨*one-letter*⟩ (where ⟨*word*⟩ is sequence of letters) because such control sequences are unacessible: proprocessor rewrites it.

The \mathsbon macro activates the rewritting 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, is is activated in the \everyjob.

math-macros.opm

```
\catcode`\_ = 8   \let\sb = _
\catcode`\_ = 13  \let _ = \sb
\catcode`\_ = 11
\_private \sb ;

\_newifi\_ifmathsb   \_mathsbfalse
\def \_mathsbon {%
   \_directlua{
   callback.register("process_input_buffer",
     function (str)
       return string.gsub(str.." ", "(\_nbb[a-zA-Z]+)_([a-zA-Z]?[^_a-zA-Z])", "\_pcent 1 _\_pcent 2")
     end) }%
   \_global\_mathsbtrue
}
\_def \_mathsboff {%
   \_directlua{ callback.register("process_input_buffer", nil) }%
   \_global \_mathsbfalse
}
\_public \mathsboff \mathsbon ;
```

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

math-macros.opm

```
\_mathcode`\^^@="2201 % \cdot
\_mathcode`\^^A="3223 % \downarrow
\_mathcode`\^^B="010B % \alpha
\_mathcode`\^^C="010C % \beta
\_mathcode`\^^D="225E % \land
\_mathcode`\^^E="023A % \lnot
\_mathcode`\^^F="3232 % \in
\_mathcode`\^^G="0119 % \pi
\_mathcode`\^^H="0115 % \lambda
\_mathcode`\^^I="010D % \gamma
\_mathcode`\^^J="010E % \delta
\_mathcode`\^^K="3222 % \uparrow
\_mathcode`\^^L="2206 % \pm
\_mathcode`\^^M="2208 % \oplus
\_mathcode`\^^N="0231 % \infty
\_mathcode`\^^O="0140 % \partial
\_mathcode`\^^P="321A % \subset
\_mathcode`\^^Q="321B % \supset
\_mathcode`\^^R="225C % \cap
\_mathcode`\^^S="225B % \cup
\_mathcode`\^^T="0238 % \forall
\_mathcode`\^^U="0239 % \exists
\_mathcode`\^^V="220A % \otimes
\_mathcode`\^^W="3224 % \leftrightarrow
\_mathcode`\^^X="3220 % \leftarrow
\_mathcode`\^^Y="3221 % \rightarrow
\_mathcode`\^^Z="8000 % \ne
\_mathcode`\^^[="2205 % \diamond
```

```
\_mathcode`\^^\="3214 % \le
\_mathcode`\^^]="3215 % \ge
\_mathcode`\^^^="3211 % \equiv
\_mathcode`\^^_="225F % \lor
\_mathcode`\ ="8000 % \space
\_mathcode`\!="5021
\_mathcode`\'="8000 % ^\prime
\_mathcode`\(="4028
\_mathcode`\)="5029
\_mathcode`\*="2203 % \ast
\_mathcode`\+="202B
\_mathcode`\,="613B
\_mathcode`\-="2200
\_mathcode`\.="013A
\_mathcode`\/="013D
\_mathcode`\:="303A
\_mathcode`\;="603B
\_mathcode`\<="313C
\_mathcode`\=="303D
\_mathcode`\>="313E
\_mathcode`\?="503F
\_mathcode`\[="405B
\_mathcode`\\="026E % \backslash
\_mathcode`\]="505D
\_mathcode`\_="8000 % math-active subscript
\_mathcode`\{="4266
\_mathcode`\|="026A
\_mathcode`\}="5267
\_mathcode`\^^?="1273 % \smallint

\_delcode`\(="028300
\_delcode`\)="029301
\_delcode`\[="05B302
\_delcode`\]="05D303
\_delcode`\<="26830A
\_delcode`\>="26930B
\_delcode`\/="02F30E
\_delcode`\|="26A30C
\_delcode`\\="26E30F
```

All control sequences declared by `\mathchardef` are supposed (by default) only for public usage. It means that they are delcared 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

```
\_mathchardef\alpha="010B
\_mathchardef\beta="010C
\_mathchardef\gamma="010D
\_mathchardef\delta="010E
\_mathchardef\epsilon="010F
\_mathchardef\zeta="0110
\_mathchardef\eta="0111
\_mathchardef\theta="0112
\_mathchardef\iota="0113
\_mathchardef\kappa="0114
\_mathchardef\lambda="0115
\_mathchardef\mu="0116
\_mathchardef\nu="0117
\_mathchardef\xi="0118
\_mathchardef\pi="0119
```

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

The math functions like log, sin, cos are declared in tha same way as in plainTEX, but they are `\protected` in OpTEX.

math-macros.opm

```
\_protected\_def\log {\_mathop{\_rm log}\_nolimits}
\_protected\_def\lg {\_mathop{\_rm lg}\_nolimits}
```

```
\_protected\_def\ln {\_mathop{\_rm ln}\_nolimits}
\_protected\_def\lim {\_mathop{\_rm lim}}
\_protected\_def\limsup {\_mathop{\_rm lim\_thinsk sup}}
\_protected\_def\liminf {\_mathop{\_rm lim\_thinsk inf}}
\_protected\_def\sin {\_mathop{\_rm sin}\_nolimits}
\_protected\_def\arcsin {\_mathop{\_rm arcsin}\_nolimits}
\_protected\_def\sinh {\_mathop{\_rm sinh}\_nolimits}
\_protected\_def\cos {\_mathop{\_rm cos}\_nolimits}
\_protected\_def\arccos {\_mathop{\_rm arccos}\_nolimits}
\_protected\_def\cosh {\_mathop{\_rm cosh}\_nolimits}
\_protected\_def\tan {\_mathop{\_rm tan}\_nolimits}
\_protected\_def\arctan {\_mathop{\_rm arctan}\_nolimits}
\_protected\_def\tanh {\_mathop{\_rm tanh}\_nolimits}
\_protected\_def\cot {\_mathop{\_rm cot}\_nolimits}
\_protected\_def\coth {\_mathop{\_rm coth}\_nolimits}
\_protected\_def\sec {\_mathop{\_rm sec}\_nolimits}
\_protected\_def\csc {\_mathop{\_rm csc}\_nolimits}
\_protected\_def\max {\_mathop{\_rm max}}
\_protected\_def\min {\_mathop{\_rm min}}
\_protected\_def\sup {\_mathop{\_rm sup}}
\_protected\_def\inf {\_mathop{\_rm inf}}
\_protected\_def\arg {\_mathop{\_rm arg}\_nolimits}
\_protected\_def\ker {\_mathop{\_rm ker}\_nolimits}
\_protected\_def\dim {\_mathop{\_rm dim}\_nolimits}
\_protected\_def\hom {\_mathop{\_rm hom}\_nolimits}
\_protected\_def\det {\_mathop{\_rm det}}
\_protected\_def\exp {\_mathop{\_rm exp}\_nolimits}
\_protected\_def\Pr {\_mathop{\_rm Pr}}
\_protected\_def\gcd {\_mathop{\_rm gcd}}
\_protected\_def\deg {\_mathop{\_rm deg}\_nolimits}
```

These macros are defined similarly as in plainTEX. Only internal macro names from plainTEX with `@` character are re-written in more readable form.
\sp is alternative for ^. The \sb alternative for _ was defined at the line 27 of the file `math-macros.opm`.

math-macros.opm

```
\_let\_sp=^ \public \sp ;
% \sb=_ , defined at beginning of this file

\_def\_thinsk {\_mskip\_thinmuskip}
\_protected\_def\,{\_relax\_ifmmode \_thinsk \_else \_thinspace \_fi}
\_protected\_def\>{\_mskip\_medmuskip}    \let\_medsk   = \>
\_protected\_def\;{\_mskip\_thickmuskip}  \let\_thicksk = \;
\_protected\_def\!{\_mskip-\_thinmuskip}  \let\_thinneg = \!
%\_def\*{\discretionary{\thinspace\the\textfont2\char2}{}{}} % obsolete
```

Active \prime character is defined here.

math-macros.opm

```
{\_catcode`\'=\active \_gdef'{^\_bgroup\_primes}} % primes dance
\_def\_primes{\_prime\_isnextchar'{\_primesA}%
                                  {\_isnextchar^{\_primesB}{\_egroup}}}
\_def\_primesA #1{\_primes}
\_def\_primesB #1#2{#2\egroup}
\_private \prime ;
```

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

math-macros.opm

```
%{\catcode`\^^Z=\active \gdef^^Z{\not=}} % ^^Z is like \ne in math %obsolete

\_def\_scalebig#1#2{{\_left#1\_vbox to#2\_fontdimen6\_textfont1{}%
                      \_kern-\_nulldelimiterspace\_right.}}
\_protected\_def\_big#1{\_scalebig{#1}{.85}}
\_protected\_def\_Big#1{\_scalebig{#1}{1.15}}
\_protected\_def\_bigg#1{\_scalebig{#1}{1.45}}
\_protected\_def\_Bigg#1{\_scalebig{#1}{1.75}}
\_public \big \Big \bigg \Bigg ;

\_protected\_def\_bigl{\_mathopen\_big}
```

```
\_protected\_def\_bigm{\_mathrel\_big}
\_protected\_def\_bigr{\_mathclose\_big}
\_protected\_def\_Bigl{\_mathopen\_Big}
\_protected\_def\_Bigm{\_mathrel\_Big}
\_protected\_def\_Bigr{\_mathclose\_Big}
\_protected\_def\_biggl{\_mathopen\_bigg}
\_protected\_def\_biggm{\_mathrel\_bigg}
\_protected\_def\_biggr{\_mathclose\_bigg}
\_protected\_def\_Biggl{\_mathopen\_Bigg}
\_protected\_def\_Biggm{\_mathrel\_Bigg}
\_protected\_def\_Biggr{\_mathclose\_Bigg}
\_public \bigl \bigm \bigr \Bigl \Bigm \Bigr \biggl \biggm \biggr \Biggl \Biggm \Biggr ;
```

Math relations defined by the \jointrel plain TEX macro:

math-macros.opm

```
\_protected\_def\_joinrel{\_mathrel{\_mkern-2.5mu}}  % -3mu in plainTeX
\_protected\_def\relbar{\_mathrel{\_smash-}} % \_smash, because - has the same height as +
\_protected\_def\Relbar{\_mathrel=}
\_mathchardef\lhook="312C
\_protected\_def\hookrightarrow{\_lhook\_joinrel\_rightarrow}
\_mathchardef\rhook="312D
\_protected\_def\hookleftarrow{\_leftarrow\_joinrel\_rhook}
\_protected\_def\bowtie{\_mathrel\_triangleright\_joinrel\_mathrel\_triangleleft}
\_protected\_def\models{\_mathrel|\_joinrel=}
\_protected\_def\Longrightarrow{\_Relbar\_joinrel\_Rightarrow}
\_protected\_def\longrightarrow{\_relbar\_joinrel\_rightarrow}
\_protected\_def\longleftarrow{\_leftarrow\_joinrel\_relbar}
\_protected\_def\Longleftarrow{\_Leftarrow\_joinrel\_Relbar}
\_protected\_def\longmapsto{\_mapstochar\_longrightarrow}
\_protected\_def\longleftrightarrow{\_leftarrow\_joinrel\_rightarrow}
\_protected\_def\Longleftrightarrow{\_Leftarrow\_joinrel\_Rightarrow}
\_protected\_def\iff{\_thicksk\_Longleftrightarrow\_thicksk}
\_private \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
   \Relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
   \longrightarrow \Longleftrightarrow ;
\_public \joinrel ;
```

\ldots, \cdots, \vdots, \ddots from plain TEX

math-macros.opm

```
\_mathchardef\_ldotp="613A % ldot as a punctuation mark
\_mathchardef\_cdotp="6201 % cdot as a punctuation mark
\_mathchardef\_colon="603A % colon as a punctuation mark
\_public \ldotp \cdotp \colon ;

\_protected\_def\_ldots{\_mathinner{\_ldotp\_ldotp\_ldotp}}
\_protected\_def\_cdots{\_mathinner{\_cdotp\_cdotp\_cdotp}}
\_protected\_def\_vdots{\_vbox{\_baselineskip=.4em \_lineskiplimit=0pt
    \_kern.6em \_hbox{.}\_hbox{.}\_hbox{.}}}
\_protected\_def\_ddots{\_mathinner{%
    \_mkern1mu\_raise.7em\_vbox{\_kern.7em\_hbox{.}}\_mkern2mu
    \_raise.4em\_hbox{.}\_mkern2mu\_raise.1em\_hbox{.}\_mkern1mu}}

\_public \ldots \cdots \vdots \ddots ;
```

Math accents (encoding dependent declarations).

math-macros.opm

```
\_protected\_def\acute{\_mathaccent"7013 }
\_protected\_def\grave{\_mathaccent"7012 }
\_protected\_def\ddot{\_mathaccent"707F }
\_protected\_def\tilde{\_mathaccent"707E }
\_protected\_def\bar{\_mathaccent"7016 }
\_protected\_def\breve{\_mathaccent"7015 }
\_protected\_def\check{\_mathaccent"7014 }
\_protected\_def\hat{\_mathaccent"705E }
\_protected\_def\vec{\_mathaccent"017E }
\_protected\_def\dot{\_mathaccent"705F }
\_protected\_def\widetilde{\_mathaccent"0365 }
\_protected\_def\widehat{\_mathaccent"0362 }
```

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

math-macros.opm

```
\_def\_math{\_mathsurround0pt }
\_protected\_def\_overrightarrow #1{\_vbox{\_math\_ialign{##\_crcr
      \_rightarrowfill\_crcr\_noalign{\_kern-.1em \_nointerlineskip}
      $\_hfil\_displaystyle{#1}\_hfil$\_crcr}}}
\_protected\_def\_overleftarrow #1{\_vbox{\_math\_ialign{##\_crcr
      \_leftarrowfill\_crcr\_noalign{\_kern-.1em \_nointerlineskip}
      $\_hfil\_displaystyle{#1}\_hfil$\_crcr}}}
\_protected\_def\_overbrace #1{\_mathop{%
      \_vbox{\_math\_ialign{##\_crcr\_noalign{\_kern.3em}
      \_downbracefill\_crcr\_noalign{\_kern.3em \_nointerlineskip}
      $\_hfil\_displaystyle{#1}\_hfil$\_crcr}}}\_limits}
\_protected\_def\_underbrace #1{\_mathop{\_vtop{\_math\_ialign{##\_crcr
      $\_hfil\_displaystyle{#1}\_hfil$\_crcr\_noalign{\_kern.3em \_nointerlineskip}
      \_upbracefill\_crcr\_noalign{\_kern.3em}}}}\_limits}
\_protected\_def\_skew #1#2#3{{\_muskip0=#1mu\_divide\_muskip0=by2 \_mkern\_muskip0
   #2{\_mkern-\_muskip0{#3}\_mkern\_muskip0}\_mkern-\_muskip0}{}}

\_public \overrightarrow \overleftarrow \overbrace \underbrace \skew ;
```

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

math-macros.opm

```
\_protected\_def\lmoustache{\_delimiter"437A340 } % top from (, bottom from )
\_protected\_def\rmoustache{\_delimiter"537B341 } % top from ), bottom from (
\_protected\_def\lgroup{\_delimiter"462833A } % extensible ( with sharper tips
\_protected\_def\rgroup{\_delimiter"562933B } % extensible ) with sharper tips
\_protected\_def\arrowvert{\_delimiter"26A33C } % arrow without arrowheads
\_protected\_def\Arrowvert{\_delimiter"26B33D } % double arrow without arrowheads
\_protected\_def\bracevert{\_delimiter"77C33E } % the vertical bar that extends braces
\_protected\_def\Vert{\_delimiter"26B30D } \_let\|=\_Vert
\_protected\_def\vert{\_delimiter"26A30C }
\_protected\_def\uparrow{\_delimiter"3222378 }
\_protected\_def\downarrow{\_delimiter"3223379 }
\_protected\_def\updownarrow{\_delimiter"326C33F }
\_protected\_def\Uparrow{\_delimiter"322A37E }
\_protected\_def\Downarrow{\_delimiter"322B37F }
\_protected\_def\Updownarrow{\_delimiter"326D377 }
\_protected\_def\backslash{\_delimiter"26E30F } % for double coset G\_backslash H
\_protected\_def\rangle{\_delimiter"526930B }
\_protected\_def\langle{\_delimiter"426830A }
\_protected\_def\rbrace{\_delimiter"5267309 } \_let\}=\rbrace \_let\_rbrace=\rbrace
\_protected\_def\lbrace{\_delimiter"4266308 } \_let\{=\lbrace \_let\_lbrace=\lbrace
\_protected\_def\rceil{\_delimiter"5265307 }
\_protected\_def\lceil{\_delimiter"4264306 }
\_protected\_def\rfloor{\_delimiter"5263305 }
\_protected\_def\lfloor{\_delimiter"4262304 }

\_protected\_def\choose{\_atopwithdelims()}
\_protected\_def\brack{\_atopwithdelims[]}
\_protected\_def\brace{\_atopwithdelims\_lbrace\_rbrace}

\_protected\_def\_sqrt{\_radical"270370 }  \_public \sqrt ;
```

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

math-macros.opm

```
\_def\_mathpalette#1#2{\_mathchoice{#1\_displaystyle{#2}}%
  {#1\_textstyle{#2}}{#1\_scriptstyle{#2}}{#1\_scriptscriptstyle{#2}}}
\_newbox\_rootbox
\_protected\_def\root#1\of{\_setbox\_rootbox
  \_hbox{$\_math\_scriptscriptstyle{#1}$}\_mathpalette\_rootA}
\_def\_rootA#1#2{\_setbox0=\_hbox{$\_math#1\_sqrt{#2}$}\_dimen0=\_ht0
  \_advance\_dimen0by-\_dp0
  \_mkern5mu\_raise.6\_dimen0\_copy\_rootbox \_mkern-10mu\_box0 }
\_newifi\_ifvp \_newifi\_ifhp
\_protected\_def\_vphantom{\_vptrue\_hpfalse\_phant}
\_protected\_def\_hphantom{\_vpfalse\_hptrue\_phant}
\_protected\_def\_phantom{\_vptrue\_hptrue\_phant}
\_def\_phant{\_ifmmode\_def\_next{\_mathpalette\_mathphant}%
  \_else\_let\_next=\_makephant\_fi\_next}
\_def\_makephant#1{\_setbox0\_hbox{#1}\_finphant}
\_def\_mathphant#1#2{\_setbox0=\_hbox{$\_math#1{#2}$}\_finphant}
```

```
\_def\_finphant{\_setbox2=\_null
  \_ifvp \_ht2=\_ht0 \_dp2=\_dp0 \_fi
  \_ifhp \_wd2=\_wd0 \_fi \_box2 }
\_def\_mathstrut{\_vphantom()}
\_protected\_def\_smash{\_relax % \_relax, in case this comes first in \halign
  \_ifmmode\_def\_next{\_mathpalette\_mathsmash}\_else\_let\_next\_makesmash
  \_fi\_next}
\_def\_makesmash#1{\_setbox0=\_hbox{#1}\_finsmash}
\_def\_mathsmash#1#2{\_setbox0=\_hbox{$\_math#1{#2}$}\_finsmash}
\_def\_finsmash{\_ht0=0pt \_dp0=0pt \_box0 }
\_public \mathpalette \vphantom \hphantom \phantom \mathstrut \smash ;
```

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

math-macros.opm

```
\_protected\_def\_cong{\_mathrel{\_mathpalette\_overeq\_sim}} % congruence sign
\_def\_overeq#1#2{\_lower.05em\_vbox{\_lineskiplimit\_maxdimen\_lineskip=-.05em
    \_ialign{$\_math#1\_hfil##\_hfil$\_crcr#2\_crcr=\_crcr}}}
\_protected\_def\_notin{\_mathrel{\_mathpalette\_cancel\_in}}
\_def\_cancel#1#2{\_math\_ooalign{$\_hfil#1\_mkern1mu/\_hfil$\_crcr$#1#2$}}
\_protected\_def\_rightleftharpoons{\_mathrel{\_mathpalette\_rlhp{}}}
\_def\_rlhp#1{\_vcenter{\_math\_hbox{\_ooalign{\_raise.2em
          \_hbox{$#1\_rightharpoonup$}\_crcr
        $#1\_leftharpoondown$}}}}
\_protected\_def\_buildrel#1\over#2{\_mathrel{\_mathop{\_kern0pt #2}\_limits^{#1}}}
\_protected\_def\_doteq{\_buildrel\_textstyle.\over=}
\_public \cong \notin \rightleftharpoons \buildrel \doteq ;

\_protected\_def\_bmod{\_nonscript\_mskip-\_medmuskip\_mkern5mu
  \_mathbin{\_rm mod}\_penalty900\_mkern5mu\_nonscript\_mskip-\_medmuskip}
\_protected\_def\_pmod#1{\_allowbreak\_mkern18mu({\_rm mod}\_thinsk\_thinsk#1)}
\_public \bmod \pmod ;
```

\cases, \matrix, \pmatrix and \bordermatrix macros from plain TEX

math-macros.opm

```
\_protected\_def\_cases#1{\_left\{\_thinsk\_vcenter{\_normalbaselines\_math
    \_ialign{$##\_hfil$&\_quad##\_hfil\_crcr#1\_crcr}}\_right.}
\_protected\_def\_matrix#1{\_null\_thinsk\_vcenter{\_normalbaselines\_math
    \_ialign{\_hfil$###$\_hfil&&\_quad\_hfil$###$\_hfil\_crcr
      \_mathstrut\_crcr\_noalign{\_kern-\_baselineskip}
      #1\_crcr\_mathstrut\_crcr\_noalign{\_kern-\_baselineskip}}}\_thinsk}
\_protected\_def\_pmatrix#1{\_left(\_matrix{#1}\_right)}
\_newdimen\_ptrenwd
\_ptrenwd=0.875\fontdimen6\textfont1 % width of the big left (
\_protected\_def\_bordermatrix#1{\_begingroup \_math
  \_setbox0=\_vbox{\_def\_cr{\_crcr\_noalign{\_kern.2em\_global\_let\_cr\_endline}}%
    \_ialign{$###$\_hfil\_kern.2em\_kern\_ptrenwd&\_thinspace\_hfil$###$\_hfil
      &&\_quad\_hfil$###$\_hfil\_crcr
      \_omit\_strut\_hfil\_crcr\_noalign{\_kern-\_baselineskip}%
      #1\_crcr\_omit\_strut\_cr}}%
  \_setbox2=\_vbox{\_unvcopy0 \_global\_setbox1=\_lastbox}%
  \_setbox2=\_hbox{\_unhbox1 \_unskip\_global\_setbox1=\_lastbox}%
  \_setbox2=\_hbox{$\_kern\_wd1 \_kern-\_ptrenwd\_left(\_kern-\_wd1
    \_global\_setbox1=\_vbox{\_box1 \_kern.2em}%
    \_vcenter{\_kern-\_ht1 \_unvbox0 \_kern-\_baselineskip}\_thinsk\_right)$}%
  \_null\_thicksk\_vbox{\_kern\_ht1 \_box2}\_endgroup}
\_public \cases \matrix \pmatrix \bordermatrix ;
```

\openup, \eqalign, \displaylines and \eqalignno macros from plain TEX.

math-macros.opm

```
\_def\_openup{\_afterassignment\_openupA\_dimen0=}
\_def\_openupA{\_advance\_lineskip by\_dimen0
  \_advance\_baselineskip by\_dimen0
  \_advance\_lineskiplimit by\_dimen0 }
\_def\_eqalign#1{\_null\_thinsk\_vcenter{\_openup\_jot\_math
  \_ialign{\_strut\_hfil$\_displaystyle{##}$&$\_displaystyle{{}##}$\_hfil
      \_crcr#1\_crcr}}\_thinsk}
\_newifi\_ifdtop
\_def\_display{\_global\_dtoptrue\_openup\_jot\_math
  \_everycr{\_noalign{\_ifdtop \_global\_dtopfalse \_ifdim\_prevdepth>-1000pt
      \_vskip-\_lineskiplimit \_vskip\_normallineskiplimit \_fi
```

```
      \_else \_penalty\_interdisplaylinepenalty \_fi}}}
\_def\_elign{\_tabskip=\_zoskip\_everycr{}} % restore inside \_display
\_def\_displaylines#1{\_display \_tabskip=\_zoskip
  \_halign{\_hbox to\_displaywidth{$\_elign\_hfil\_displaystyle##\_hfil$}\_crcr
    #1\_crcr}}
\_def\_eqalignno#1{\_display \_tabskip=\_centering
  \_halign to\_displaywidth{\_hfil$\_elign\_displaystyle{##}$\_tabskip=\_zoskip
    &$\_elign\_displaystyle{{}##}$\_hfil\_tabskip\_centering
    &\_llap{$\_elign##$}\_tabskip\_zoskip\_crcr
    #1\_crcr}}
\_def\_leqalignno#1{\_display \_tabskip=\_centering
  \_halign to\_displaywidth{\_hfil$\_elign\_displaystyle{##}$\_tabskip=\_zoskip
    &$\_elign\_displaystyle{{}##}$\_hfil\_tabskip=\_centering
    &\_kern-\_displaywidth\_rlap{$\_elign##$}\_tabskip\_displaywidth\_crcr
    #1\_crcr}}
\_public \openup \eqalign \displaylines \eqalignno ;
```

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

math-macros.opm

```
\_def\_amsafam{4} \_def\_amsbfam{5}

\_mathchardef \boxdot    "2\_amsafam 00
\_mathchardef \boxplus   "2\_amsafam 01
\_mathchardef \boxtimes   "2\_amsafam 02
\_mathchardef \square   "0\_amsafam 03
\_mathchardef \blacksquare   "0\_amsafam 04
\_mathchardef \centerdot   "2\_amsafam 05
\_mathchardef \lozenge   "0\_amsafam 06
\_mathchardef \blacklozenge   "0\_amsafam 07
\_mathchardef \circlearrowright   "3\_amsafam 08
\_mathchardef \circlearrowleft   "3\_amsafam 09
\_mathchardef \rightleftharpoons   "3\_amsafam 0A
\_mathchardef \leftrightharpoons   "3\_amsafam 0B
\_mathchardef \boxminus   "2\_amsafam 0C
```

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

The \not macro is re-defined to be mote inteligent 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

```
\_mathchardef \_notchar  "3236

\_protected\_def \_not#1{%
  \_ifx #1<\_nless \_else
  \_ifx #1>\_ngtr \_else
  \_edef\_tmpn{\_csstring#1}%
  \_ifcsname _not\_tmpn\_endcsname \_csname _not\_tmpn\_endcsname
  \_else \_ifcsname _n\_tmpn\_endcsname \_csname _n\_tmpn\_endcsname
  \_else \_mathrel{\_mathord{\_notchar}\_mathord{#1}}%
  \_fi \_fi \_fi \_fi}
\_private
  \nleq \ngeq \nless \ngtr \nprec \nsucc \nleqslant \ngeqslant \npreceq
  \nsucceq \nleqq \ngeqq \nsim \ncong \nsubseteqq \nsupseteqq \nsubseteq
  \nsupseteq \nparallel \nmid \nshortmid \nshortparallel \nvdash \nVdash
  \nvDash \nVDash \ntrianglerighteq \ntrianglelefteq \ntriangleleft
  \ntriangleright \nleftarrow \nrightarrow \nLeftarrow \nRightarrow
  \nLeftrightarrow \nleftrightarrow \nexists ;
\_public \not ;
```

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

math-macros.opm

```
\_def\_mathbox#1{{\_mathchoice{\_mathboxA\_displaystyle[]{#1}}{\_mathboxA\_textstyle[]{#1}}
    {\_mathboxA\_textstyle[700]{#1}}{\_mathboxA\_textstyle[500]{#1}}}}
```

```
\_def\_mathboxA#1[#2]#3{\_hbox{\_everymath={#1}\_if^#2^\_else\_typoscale[#2/]\_relax\_fi #3}}
\_public \mathbox ;
```

2.15 Unicode-math fonts

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

`\loadboldmath`

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

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

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

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

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

The `\loadmath` macro was succesfully tested on:

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

2.15.1 Unicode-math macros preloaded in the format

math-unicode.opm

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

`\loadmath {`⟨*Unicode-math font*⟩`}` loads given font. It does:

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

`\noloadmath` disallows Unicode-math loading by `\_mathloadingfalse`.
`\doloadmath` allows Unicode-math loading by `\_mathloadingtrue`.

math-unicode.opm

```
\_newifi \_ifmathloading   \_mathloadingtrue

\_def\_noloadmath{\_mathloadingfalse}
\_def\_doloadmath{\_mathloadingtrue}

\_def\_loadmath#1{%
   \_ifmathloading
   \_initunifonts
   \_isfont{#1}\_iffalse
      \_opwarning{Math font "#1" not found, skipped...}%
   \_else
      \_def\_unimathfont{#1}%
      \_let\_normalmath = \_normalunimath  \_let\_boldmath = \_boldunimath
```

```
      \_normalmath
      \_wterm {MATH-FONT: "#1" -- unicode math prepared.}%
      \_opinput {unimath-codes.opm}%
      \_mathloadingfalse
   \_fi\_fi}

\_public \loadmath \noloadmath \doloadmath ;
```

`\loadboldmath {`⟨*bold-font*⟩`} \to {`⟨*normal-font*⟩`}` defines `\_unimathboldfont` as ⟨*bold-font*⟩ only if `\_unimathfont` is defined as ⟨*normal-font*⟩. It is used when `\boldmath` macro is run. When no `\_unimathboldfont` is defined then the `\boldmath` macro use "fake bold" generated by `embolden` LuaTEX font feature.

math-unicode.opm

```
\_def\_loadboldmath#1#2\to #3{%
   \_def\_tmp{#3}\_ifx\_unimathfont\_tmp % do work only if #3 is loaded as normal Math
   \_isfont"#1"\_iffalse
      \_opwarning{Bold-Math font "#1" not found, skipped...}
   \_else
      \_def\_unimathboldfont{#1}%
      \_wterm {MATH-FONT: "#1" -- unicode math bold prepared.}%
   \_fi\_fi}

\_public \loadboldmath ;
```

The Unicode version of the `\normalmath` and `\boldmath` macros are defined here as `\_normalunimath` and `\_boldunimath` macros. They are using `\_setunimathdimens` in similar sense as `\_setmathdimens`.

math-unicode.opm

```
\_def\_normalunimath{%
    \_loadumathfamily 1 {\_unimathfont}{} % Base font
    \_loadmathfamily  4 rsfs              % script
    \_setunimathdimens
}%
\_def\_boldunimath{%
    \_ifx\_unimathboldfont \_undefined
       \_loadumathfamily 1 {\_unimathfont}{embolden=1.7;} % Base faked bold
    \_else
       \_loadumathfamily 1 {\_unimathboldfont}{} % Base real bold font
    \_fi
    \_loadmathfamily  4 rsfs              % script
    \_setunimathdimens
}%
\_def\_setunimathdimens{% PlainTeX sets these dimens for 10pt size only:
  \_delimitershortfall=0.5\_fontdimen6\_textfont3
  \_nulldelimiterspace=0.12\_fontdimen6\_textfont3
  \_scriptspace=0.05\_fontdimen6\_textfont3
}
```

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

math-unicode.opm

```
\_def\_umathname#1#2{"#1:\_mfontfeatures#2"}
\_def\_mfontfeatures{mode=base;script=math;}

\_def\_loadumathfamily #1 #2#3 {%
  \_edef\_optsizesave{\_the\_optsize}%
  \_optsize=\_sizemtext  \_font\_mF=\_umathname{#2}{#3} at\_optsize \_textfont#1=\_mF
  \_ifnum#1=1 \_textfont2=\_mF \_textfont3=\_mF \_fi
  \_optsize=\_sizemscript \_font\_mF=\_umathname{#2}{+ssty=0;#3} at\_optsize \_scriptfont#1=\_mF
  \_ifnum#1=1 \_scriptfont2=\_mF \_scriptfont3=\_mF \_fi
  \_optsize=\_sizemsscript \_font\_mF=\_umathname{#2}{+ssty=1;#3} at\_optsize\_scriptscriptfont#1=\_mF
  \_ifnum#1=1 \_scriptscriptfont2=\_mF \_scriptscriptfont3=\_mF \_fi
  \_optsize=\_optsizesave \_relax
}
```

Unicode math font includes all typical math alphabets together, user needs not to load more TEX math families. These math aphabets are encoded by different parts of Unicode table. We need auxiliary macros for setting mathcodes by selected math alphabet.
_umathrange {⟨*from*⟩-⟨*to*⟩}\⟨*first*⟩ sets \Umathcodes of the characters in the interval ⟨*from*⟩-⟨*to*⟩ to \⟨*first*⟩, \⟨*first*⟩+1, \⟨*first*⟩+2 etc., but _umathcharholes are skipped (_umathcharholes are parts of the Unicode table not designed for math alphabets but they causes that the math alphabets are not continuously spread out in the table; I mean that the designers were under the influence of drugs when they created this part of the Unicode table). The ⟨*from*⟩-⟨*to*⟩ clause includes normal letters like A-Z.
_umahrangegreek \⟨*first*⟩ is the same as _umathrange {⟨*alpha*⟩-⟨*omega*⟩}\⟨*first*⟩.
_umahrangeGREEK \⟨*first*⟩ is the same as _umathrange {⟨*Alpha*⟩-⟨*Omega*⟩}\⟨*first*⟩.
_greekdef ⟨*control sequences*⟩ _relax defines each control sequence as a normal character with codes _umathnumB, _umathnumB+1, _umathnumB+2 etc. It is used for redefinig the contol sequences for math Greek \alpha, \beta, \gamma etc.

math-unicode.opm

```
\_newcount\_umathnumA  \_newcount\_umathnumB

\_def\_umathcorr#1#2{\_ea#1\_ea{\_the#2}}
\_def\_umathprepare#1{\_def\_umathscanholes##1[#1]##2##3\_relax{##2}}
\_def\_umathvalue#1{\_ea\_umathscanholes\_umathcharholes[#1]{#1}\_relax}

\_def\_umathcharholes{% holes in math alphabets:
   [119893]{"210E}[119965]{"212C}[119968]{"2130}[119969]{"2131}%
   [119971]{"210B}[119972]{"2110}[119975]{"2112}[119976]{"2133}[119981]{"211B}%
   [119994]{"212F}[119996]{"210A}[120004]{"2134}%
   [120070]{"212D}[120075]{"210C}[120076]{"2111}[120085]{"211C}[120093]{"2128}%
   [120122]{"2102}[120127]{"210D}[120133]{"2115}[120135]{"2119}
   [120136]{"211A}[120137]{"211D}[120145]{"2124}%
}
\_def\_umathrange#1#2{\_umathnumB=#2\_relax \_umathrangeA#1}
\_def\_umathrangeA#1-#2{\_umathnumA=`#1\_relax
   \_loop
      \_umathcorr\_umathprepare\_umathnumB
      \_Umathcode \_umathnumA = 7 1 \_umathcorr\_umathvalue{\_umathnumB}
      \_ifnum\_umathnumA<`#2\_relax
         \_advance\_umathnumA by1 \_advance\_umathnumB by1
   \_repeat
}
\_def\_umathrangeGREEK{\_begingroup
   \_lccode`A="0391 \_lccode`Z="03A9
   \_lowercase{\_endgroup \_umathrange{A-Z}}}
\_def\_umathrangegreek{\_begingroup
   \_lccode`A="03B1 \_lccode`Z="03D6
   \_lowercase{\_endgroup \_umathrange{A-Z}}}
\_def\_greekdef#1{\_ifx#1\_relax \_else
   \_begingroup \_lccode`X=\_umathnumB \_lowercase{\_endgroup \_def#1{X}}%
   \_advance\_umathnumB by 1
   \_expandafter\_greekdef \_fi
}
```

2.15.2 Macros and codes set when \loadmatfont is processed

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

unimath-codes.opm

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

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

unimath-codes.opm

```
\_umathnumB="0391
\_greekdef \Alpha \Beta \Gamma \Delta \Epsilon \Zeta \Eta \Theta \Iota \Kappa
   \Lambda \Mu \Nu \Xi \Omicron \Pi \Rho \varTheta \Sigma \Tau \Upsilon \Phi
   \Chi \Psi \Omega \_relax

\_umathnumB="03B1
```

```
\_greekdef \alpha \beta \gamma \delta \epsilon \zeta \eta \theta \iota \kappa
   \lambda \mu \nu \xi \omicron \pi \rho \varsigma \sigma \tau \upsilon
   \varphi \chi \psi \omega \varbeta \vartheta \phi \varpi \_relax
```

The math alphabets are declared here using the _umathrange macro.

unimath-codes.opm

```
\_chardef\_ncharrmA="A       \_chardef\_ncharrma="a
\_chardef\_ncharbfA="1D400   \_chardef\_ncharbfa="1D41A
\_chardef\_ncharitA="1D434   \_chardef\_ncharita="1D44E
\_chardef\_ncharbiA="1D468   \_chardef\_ncharbia="1D482
\_chardef\_ncharclA="1D49C   \_chardef\_ncharcla="1D4B6
\_chardef\_ncharbcA="1D4D0   \_chardef\_ncharbca="1D4EA
\_chardef\_ncharfrA="1D504   \_chardef\_ncharfra="1D51E
\_chardef\_ncharbrA="1D56C   \_chardef\_ncharbra="1D586
\_chardef\_ncharbbA="1D538   \_chardef\_ncharbba="1D552
\_chardef\_ncharsnA="1D5A0   \_chardef\_ncharsna="1D5BA
\_chardef\_ncharbsA="1D5D4   \_chardef\_ncharbsa="1D5EE
\_chardef\_ncharsiA="1D608   \_chardef\_ncharsia="1D622
\_chardef\_ncharsxA="1D63C   \_chardef\_ncharsxa="1D656
\_chardef\_nchartA="1D670    \_chardef\_nchartta="1D68A

\_protected\_def\_rmvariables     {\_umathrange{A-Z}\_ncharrmA \_umathrange{a-z}\_ncharrma}
\_protected\_def\_bfvariables     {\_umathrange{A-Z}\_ncharbfA \_umathrange{a-z}\_ncharbfa}
\_protected\_def\_nitvariables    {\_umathrange{A-Z}\_ncharitA \_umathrange{a-z}\_ncharita}
\_protected\_def\_bivariables     {\_umathrange{A-Z}\_ncharbiA \_umathrange{a-z}\_ncharbia}
\_protected\_def\_calvariables    {\_umathrange{A-Z}\_ncharclA \_umathrange{a-z}\_ncharcla}
\_protected\_def\_bcalvariables   {\_umathrange{A-Z}\_ncharbcA \_umathrange{a-z}\_ncharbca}
\_protected\_def\_frakvariables   {\_umathrange{A-Z}\_ncharfrA \_umathrange{a-z}\_ncharfra}
\_protected\_def\_bfrakvariables  {\_umathrange{A-Z}\_ncharbrA \_umathrange{a-z}\_ncharbra}
\_protected\_def\_bbvariables     {\_umathrange{A-Z}\_ncharbbA \_umathrange{a-z}\_ncharbba}
\_protected\_def\_sansvariables   {\_umathrange{A-Z}\_ncharsnA \_umathrange{a-z}\_ncharsna}
\_protected\_def\_bsansvariables  {\_umathrange{A-Z}\_ncharbsA \_umathrange{a-z}\_ncharbsa}
\_protected\_def\_isansvariables  {\_umathrange{A-Z}\_ncharsiA \_umathrange{a-z}\_ncharsia}
\_protected\_def\_bisansvariables {\_umathrange{A-Z}\_ncharsxA \_umathrange{a-z}\_ncharsxa}
\_protected\_def\_ttvariables     {\_umathrange{A-Z}\_nchartA \_umathrange{a-z}\_nchartta}

\_chardef\_greekrmA="0391   \_chardef\_greekrma="03B1
\_chardef\_greekbfA="1D6A8  \_chardef\_greekbfa="1D6C2
\_chardef\_greekitA="1D6E2  \_chardef\_greekita="1D6FC
\_chardef\_greekbiA="1D71C  \_chardef\_greekbia="1D736
\_chardef\_greeksnA="1D756  \_chardef\_greeksna="1D770
\_chardef\_greeksiA="1D790  \_chardef\_greeksia="1D7AA

\_protected\_def\_nitgreek   {\_umathrangeGREEK\_greekrmA \_umathrangegreek\_greekita}
\_protected\_def\_rmgreek    {\_umathrangeGREEK\_greekrmA \_umathrangegreek\_greekrma}
\_protected\_def\_bfgreek    {\_umathrangeGREEK\_greekbfA \_umathrangegreek\_greekbfa}
\_protected\_def\_bigreek    {\_umathrangeGREEK\_greekbfA \_umathrangegreek\_greekbia}
\_protected\_def\_sansgreek  {\_umathrangeGREEK\_greeksnA \_umathrangegreek\_greeksna}
\_protected\_def\_isansgreek {\_umathrangeGREEK\_greeksnA \_umathrangegreek\_greeksia}

% Another possibility (slanted capitals in \_nitgreek, \_bigreek, \_isansgreek):
%\_protected\_def\_nitgreek   {\_umathrangeGREEK\_greekitA \_umathrangegreek\_greekita}
%\_protected\_def\_rmgreek    {\_umathrangeGREEK\_greekrmA \_umathrangegreek\_greekrma}
%\_protected\_def\_bfgreek    {\_umathrangeGREEK\_greekbfA \_umathrangegreek\_greekbfa}
%\_protected\_def\_bigreek    {\_umathrangeGREEK\_greekbiA \_umathrangegreek\_greekbia}
%\_protected\_def\_sansgreek  {\_umathrangeGREEK\_greeksnA \_umathrangegreek\_greeksna}
%\_protected\_def\_isansgreek {\_umathrangeGREEK\_greeksiA \_umathrangegreek\_greeksia}

\_chardef\_digitrmO=`0
\_chardef\_digitbfO="1D7CE
\_chardef\_digitbbO="1D7D8
\_chardef\_digitsnO="1D7E2
\_chardef\_digitbsO="1D7EC
\_chardef\_digitttO="1D7F6

\_protected\_def\_rmdigits    {\_umathrange{0-9}\_digitrmO}
\_protected\_def\_bfdigits    {\_umathrange{0-9}\_digitbfO}
\_protected\_def\_bbdigits    {\_umathrange{0-9}\_digitbbO}
\_protected\_def\_sansdigits  {\_umathrange{0-9}\_digitsnO}
\_protected\_def\_bsansdigits {\_umathrange{0-9}\_digitbsO}
```

```
\_protected\_def\_ttdigits    {\_umathrange{0-9}\_digitttO}
```

The `\rm`, `\it`, `\cal` etc. are redefined here.
You can redefine them if you need different behavior (for example you don't want to use sans serif bold in math). When you do this then you must repeat `\_public \bf ;`
`\_inmath {`⟨*cmds*⟩`}` applies ⟨*cmds*⟩ only in math mode.

unimath-codes.opm

```
\_protected\_def\_inmath#1{\_relax \_ifmmode#1\_fi} % to keep off \loop processing in text mode

% You can redefine these macros to follow your wishes.
% For example you need upgright lowercase greek letters, you don't need
% \bf and \bi behaves as sans serif in math, ...

\_protected\_def\_rm {\_tryloadrm \_tenrm \_inmath{\_rmvariables \_rmdigits}}
\_protected\_def\_it {\_tryloadit \_tenit \_inmath{\_nitvariables}}
\_protected\_def\_bf {\_tryloadbf \_tenbf \_inmath{\_bsansvariables \_sansgreek \_bsansdigits}}
\_protected\_def\_bi {\_tryloadbi \_tenbi \_inmath{\_bisansvariables \_isansgreek \_bsansdigits}}
\_protected\_def\_tt {\_tryloadtt \_tentt \_inmath{\_ttvariables \_ttdigits}}
\_protected\_def\_bbchar  {\_bbvariables \_bbdigits}
\_protected\_def\_cal     {\_calvariables}
\_protected\_def\_frak    {\_frakvariables}
\_protected\_def\_misans  {\_isansvariables \_isansgreek \_sansdigits}
\_protected\_def\_mbisans {\_bisansvariables \_isansgreek \_bsansdigits}
\_protected\_def\_script  {\_rmvariables \_fam4 }

\_public \rm \it \bf \bi \tt \bbchar \cal \frak \misans \mbisans \script ;
```

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

mathclass.opm

```
002E;P
002F;B
0030..0039;N
003A;P
003B;P
003C;R
003D;R
003E;R
003F;P
0040;N
0041..005A;A
005B;O
005C;B
005D;C
005E;N
005F;N
```

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

unimath-codes.opm

```
\_begingroup  % \input mathclass.opm  (which is a copy of MathClass.txt):
   \_def\_p#1;#2{\_edef\_tmp{\_pB#2}\_ifx\_tmp\_empty \_else\_pA#1....\_end#2\_fi}
   \_def\_pA#1..#2..#3\_end#4{%
      \_ifx\_relax#2\_relax \_pset{"#1}{#4}\_else
         \_umathnumA="#1
         \_loop
            \_pset{\_umathnumA}{#4}%
            \_ifnum\_umathnumA<"#2 \_advance\_umathnumA by1
         \_repeat
      \_fi
   }
   \_def\_pB#1{\_if#1L1\_fi \_if#1B2\_fi \_if#1V2\_fi \_if#1R3\_fi \_if#1N0\_fi \_if#1U0\_fi
             \_if#1F0\_fi \_if#1O4\_fi \_if#1C5\_fi \_if#1P6\_fi \_if#1A7\_fi}
   \_def\_pset#1#2{\_global\_Umathcode#1=\_tmp\_space 1 #1\_relax
      \_if#2O\_global\_Udelcode#1=1 #1\_relax\_fi
      \_if#2C\_global\_Udelcode#1=1 #1\_relax\_fi
      \_if#2F\_global\_Udelcode#1=1 #1\_relax\_fi
   }
   \_catcode`#=14
   \_everypar={\_setbox0=\_lastbox \_par \_p}
```

```
  \_input mathclass.opm
\_endgroup
```

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

unimath-table.opm

```
\UnicodeMathSymbol{"00397}{\mupEta                  }{\mathalpha}{capital eta, greek}%
\UnicodeMathSymbol{"00398}{\mupTheta                }{\mathalpha}{capital theta, greek}%
\UnicodeMathSymbol{"00399}{\mupIota                 }{\mathalpha}{capital iota, greek}%
\UnicodeMathSymbol{"0039A}{\mupKappa                }{\mathalpha}{capital kappa, greek}%
\UnicodeMathSymbol{"0039B}{\mupLambda               }{\mathalpha}{capital lambda, greek}%
\UnicodeMathSymbol{"0039C}{\mupMu                   }{\mathalpha}{capital mu, greek}%
\UnicodeMathSymbol{"0039D}{\mupNu                   }{\mathalpha}{capital nu, greek}%
\UnicodeMathSymbol{"0039E}{\mupXi                   }{\mathalpha}{capital xi, greek}%
\UnicodeMathSymbol{"0039F}{\mupOmicron              }{\mathalpha}{capital omicron, greek}%
\UnicodeMathSymbol{"003A0}{\mupPi                   }{\mathalpha}{capital pi, greek}%
\UnicodeMathSymbol{"003A1}{\mupRho                  }{\mathalpha}{capital rho, greek}%
\UnicodeMathSymbol{"003A3}{\mupSigma                }{\mathalpha}{capital sigma, greek}%
\UnicodeMathSymbol{"003A4}{\mupTau                  }{\mathalpha}{capital tau, greek}%
\UnicodeMathSymbol{"003A5}{\mupUpsilon              }{\mathalpha}{capital upsilon, greek}%
\UnicodeMathSymbol{"003A6}{\mupPhi                  }{\mathalpha}{capital phi, greek}%
\UnicodeMathSymbol{"003A7}{\mupChi                  }{\mathalpha}{capital chi, greek}%
```

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

unimath-codes.opm

```
\_begingroup  % \input unimath-table.opm (it is a copy of unicode-math-table.tex):
   \_def\UnicodeMathSymbol #1#2#3#4{%
      \_global\_Umathcharnumdef#2=\_Umathcodenum#1\_relax
      \_ifx#3\_mathopen   \_gdef#2{\_Udelimiter 4 1 #1 }\_fi
      \_ifx#3\_mathclose  \_gdef#2{\_Udelimiter 5 1 #1 }\_fi
      \_ifx#3\_mathaccent \_gdef#2{\_Umathaccent fixed 7 1 #1 }\_fi
   }
   \_input unimath-table.opm
\_endgroup
```

Many special characters must be declared with care...

unimath-codes.opm

```
\_global\_Udelcode`<=1 "027E8 % these characters have different meaning
\_global\_Udelcode`>=1 "027E9 % as normal and as delimeter

\_nitgreek \_nitvariables \_rmdigits   % default setting

\_Umathcode `- = 2 1 "2212
\_let\{=\lbrace \_let\}=\rbrace

\_protected\_def \_sqrt       {\_Uradical 1 "0221A }
\_protected\_def \_cuberoot   {\_Uradical 1 "0221B }
\_protected\_def \_fourthroot {\_Uradical 1 "0221C }

\_public \sqrt \cuberoot \fourthroot ;

\_def\_intwithnolimits#1#2 {\_ifx#1\_relax \_else
   \_ea\_let\_csname\_csstring#1op\_endcsname=#1%
   \_ea\_def\_ea #1\_ea{\_csname\_csstring#1op\_endcsname \_nolimits}%
   \_bgroup \_lccode`\~=#2 \_lowercase{\_egroup \_mathcode`~="8000 \_let ~=#1}%
   \_ea \_intwithnolimits \_fi
}
\_intwithnolimits \int "0222B \iint "0222C \iiint "0222D
   \oint "0222E \oiint "0222F \oiiint "02230
   \intclockwise "02231 \varointclockwise "02232 \ointctrclockwise "02233
   \sumint "02A0B \iiiint "02A0C \intbar "02A0D \intBar "02A0E \fint "02A0F
   \pointint "02A15 \sqint "02A16 \intlarhk "02A17 \intx "02A18
   \intcap "02A19 \intcup "02A1A \upint "02A1B \lowint "02A1C \_relax "0

\_protected\_def \vert  {\_Udelimiter 0 1 "07C }
\_protected\_def \Vert  {\_Udelimiter 0 1 "02016 }
\_protected\_def \Vvert {\_Udelimiter 0 1 "02980 }

\_protected\_def \_overbrace    #1{\mathop {\Umathaccent  7 1 "023DE{#1}}\limits}
\_protected\_def \_underbrace   #1{\mathop {\Umathaccent bottom 7 1 "023DF{#1}}\limits}
```

```
\_protected\_def \_overparen    #1{\mathop {\Umathaccent  7 1 "023DC{#1}}\limits}
\_protected\_def \_underparen   #1{\mathop {\Umathaccent bottom 7 1 "023DD{#1}}\limits}
\_protected\_def \_overbracket  #1{\mathop {\Umathaccent  7 1 "023B4{#1}}\limits}
\_protected\_def \_underbracket #1{\mathop {\Umathaccent bottom 7 1 "023B5{#1}}\limits}

\_public \overbrace \underbrace \overparen \underparen \overbracket \underbracket ;

\_protected\def \widehat            {\Umathaccent 7 1 "00302 }
\_protected\def \widetilde          {\Umathaccent 7 1 "00303 }
\_protected\def \overleftharpoon    {\Umathaccent 7 1 "020D0 }
\_protected\def \overrightharpoon   {\Umathaccent 7 1 "020D1 }
\_protected\def \overleftarrow      {\Umathaccent 7 1 "020D6 }
\_protected\def \overrightarrow     {\Umathaccent 7 1 "020D7 }
\_protected\def \overleftrightarrow {\Umathaccent 7 1 "020E1 }

\_mathchardef\ldotp="612E
\_let\|=\Vert
\_mathcode`\_="8000
```

Aliases are declared here. They are names not menitioned in the `unimath-table.opm` file but commonly used in TEX.

unimath-codes.opm

```
\_let \setminus=\smallsetminus
\_let \diamond=\smwhtdiamond
\_let \bullet=\smblkcircle
\_let \circ=\vysmwhtcircle
\_let \bigcirc=\mdlgwhtcircle
\_let \to=\rightarrow
\_let \le=\leq
\_let \ge=\geq
\_let \neq=\ne
\_protected\_def \triangle {\mathord{\bigtriangleup}}
\_let \emptyset=\varnothing
\_let \hbar=\hslash
\_let \land=\wedge
\_let \lor=\vee
\_let \owns=\ni
\_let \gets=\leftarrow
\_let \mathring=\ocirc
\_let \lnot=\neg
\_let \longdivision=\longdivisionsign
\_let \backepsilon=\upbackepsilon
\_let \eth=\matheth
\_let \dbkarow=\dbkarrow
\_let \drbkarow=\drbkarrow
\_let \hksearow=\hksearrow
\_let \hkswarow=\hkswarrow

\_let \varepsilon=\epsilon
\_let \upalpha=\mupalpha
\_let \upbeta=\mupbeta
\_let \upgamma=\mupgamma
\_let \updelta=\mupdelta
\_let \upepsilon=\mupvarepsilon
\_let \upvarepsilon=\mupvarepsilon
\_let \upzeta=\mupzeta
\_let \upeta=\mupeta
\_let \uptheta=\muptheta
\_let \upiota=\mupiota
\_let \upkappa=\mupkappa
\_let \uplambda=\muplambda
\_let \upmu=\mupmu
\_let \upnu=\mupnu
\_let \upxi=\mupxi
\_let \upomicron=\mupomicron
\_let \uppi=\muppi
\_let \uprho=\muprho
\_let \upvarrho=\mupvarrho
\_let \upvarsigma=\mupvarsigma
```

```
\_let \upsigma=\mupsigma
\_let \uptau=\muptau
\_let \upupsilon=\mupupsilon
\_let \upvarphi=\mupvarphi
\_let \upchi=\mupchi
\_let \uppsi=\muppsi
\_let \upomega=\mupomega
\_let \upvartheta=\mupvartheta
\_let \upphi=\mupphi
\_let \upvarpi=\mupvarpi
```

The \not macro is redefined here.

unimath-codes.opm

```
\_protected\_def\_not#1{%
  \_ifcsname _not!\_csstring#1\_endcsname \_csname _not!\_csstring#1\_endcsname
  \_else \_mathrel{\_mathord{\_rlap{\_kern1pt/}}\_mathord{#1}}%
  \_fi
}
\_def\_negationof #1#2{\_ea\_let \_csname _not!\_csstring#1\_endcsname =#2}
\_negationof =         \neq
\_negationof <         \nless
\_negationof >         \ngtr
\_negationof \gets     \nleftarrow
\_negationof \simeq    \nsime
\_negationof \equal    \ne
\_negationof \le       \nleq
\_negationof \ge       \ngeq
\_negationof \greater  \ngtr
\_negationof \forksnot \forks
\_negationof \in       \notin

\_public \not ;
```

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

unimath-codes.opm

```
\_private
   \ldotp \cdotp \bullet \triangleleft \trianglerigt \mapstochar \rightarrow
   \prime \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
   \Relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
   \longrightarrow \Longleftrightarrow \vdots \ddots ;
```

2.15.3 A few observations

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

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

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

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

2.16 Scaling fonts in document (high-level macros)

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

fonts-opmac.opm

```
\_codedecl \typosize {Font managing macros from OPmac <2020-04-14>} % loaded in format
```

\typosize [⟨*font-size*⟩/⟨*baselineskip*⟩] sets given parameters. It sets text font size by the \setfontsize macro and math font sizes by setting internal macros _sizemtext, _sizemscript and _sizemsscript.

It uses common concept font thes sizes: 100 %, 70 % and 50 %. The _setmainvalues sets the parameters as main values when the _typosize is called first.

fonts-opmac.opm

```
\_protected\_def \_typosize [#1/#2]{%
   \_textfontsize{#1}\_mathfontsize{#1}\_setbaselineskip{#2}%
   \_setmainvalues \_ignorespaces
}
\_protected\_def \_textfontsize #1{\_if$#1$\_else \_setfontsize{at#1\_ptunit}\_fi}

\_def \_mathfontsize #1{\_if$#1$\_else
    \_tmpdim=#1\_ptunit
    \_edef\_sizemtext{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
    \_tmpdim=0.7\_tmpdim
    \_edef\_sizemscript{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
    \_tmpdim=#1\_ptunit \_tmpdim=0.5\_tmpdim
    \_edef\_sizemsscript{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
    \_fi
}
\_public \typosize ;
```

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

fonts-opmac.opm

```
\_protected\_def \_typoscale [#1/#2]{%
   \_ifx$#1$\_def\_tmp{[/}\_else
      \_settmpdim{#1}\_optsize
      \_edef\_tmp{[\_ea\_ignorept\_the\_tmpdim/}\_fi
   \_ifx$#2$\_edef\_tmp{\_tmp]}\_else
      \_settmpdim{#2}\_baselineskip
      \_edef\_tmp{\_tmp \_ea\_ignorept\_the\_tmpdim]}\fi
   \_ea\_typosize\_tmp
}
\def\_settmpdim#1#2{%
   \_tmpdim=#1pt \_divide\_tmpdim by1000
   \_tmpdim=\_ea\_ignorept \_the#2\_tmpdim
}
\_public \typoscale ;
```

_setbaselineskip {⟨*baselineskip*⟩} sets new \baselineskip and more values of registers which are dependent on the ⟨*baselineskip*⟩ including the \strutbox.

fonts-opmac.opm

```
\_def \_setbaselineskip #1{\_if$#1$\_else
   \_tmpdim=#1\_ptunit
   \_baselineskip=\_tmpdim \_relax
   \_bigskipamount=\_tmpdim plus.33333\_tmpdim minus.33333\_tmpdim
   \_medskipamount=.5\_tmpdim plus.16666\_tmpdim minus.16666\_tmpdim
   \_smallskipamount=.25\_tmpdim plus.08333\_tmpdim minus.08333\_tmpdim
   \_normalbaselineskip=\_tmpdim
   \_jot=.25\_tmpdim
   \_maxdepth=.33333\_tmpdim
   \_setbox\_strutbox=\_hbox{\_vrule height.709\_tmpdim depth.291\_tmpdim width0pt}%
   \_fi
}
```

_setmainvalues sets the current font size and \baselineskip values to the \mainfosize and \mainbaselineskip registers. It redefines itself in order to set the main values only first.
\scalemain returns to these values if they were set. Else they are set to 10/12 pt.

fonts-opmac.opm

```
\_newskip   \_mainbaselineskip   \_mainbaselineskip=0pt \_relax
\_newdimen  \_mainfosize         \_mainfosize=0pt

\_def\_setmainvalues {%
   \_mainbaselineskip=\_baselineskip
   \_mainfosize=\_optsize
   \_topskip=\_mainfosize \_splittopskip=\_topskip
   \_ifmmode \_else \_bf \_it \_bi \_rm \_fi % load all basic variants of the family
   \_normalmath  % load fonts if \typosize is running first
   \_let \_setmainvalues =\_setmainvaluesL
}
```

```
\_def\_setmainvaluesL {\_ifmmode \_normalmath \_else
   \_rm \_everymath={\_normalmath}\_everydisplay={\_normalmath}\_fi}
\_def\_scalemain {%
   \_ifdim \_mainfosize=0pt
      \_mainfosize=10pt  \_mainbaselineskip=12pt
      \_let \_setmainvalues=\_setmainvaluesL
    \fi
   \_optsize=\_mainfosize  \_baselineskip=\_mainbaselineskip
}
\_public \scalemain \mainfosize \mainbaselineskip ;
```

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

fonts-opmac.opm

```
\_protected\_def\_thefontsize[#1]{\_if$#1$\_else
     \_tmpdim=#1\_ptunit
     \_newcurrfontsize{at\_tmpdim}%
  \_fi
  \_ignorespaces
}
\_protected\_def\_thefontscale[#1]{\_ifx$#1$\_else
     \_tmpdim=#1pt \_divide\_tmpdim by1000
     \_tmpdim=\_ea\_ea\_ea\_ignorept \_pdffontsize\_font \_tmpdim
     \_newcurrfontsize{at\_tmpdim}%
  \_fi
  \_ignorespaces
}
\_public \thefontsize \thefontscale ;
```

\em keeps the weight of the current vaiant and switches roman ↔ italic. It adds the italic correction by the _additcorr and _afteritcorr macros. The second does not add italic correction if the next character is dot or comma.

fonts-opmac.opm

```
\_protected\_def\_em {%
   \_ea\_ifx \_the\_font \_tenit \_additcorr \_rm  \_else
   \_ea\_ifx \_the\_font \_tenbf \_bi\_aftergroup\_afteritcorr\_else
   \_ea\_ifx \_the\_font \_tenbi \_additcorr \_bf  \_else
   \_it \_aftergroup\_afteritcorr\_fi\_fi\_fi
}
\_def\_additcorr{\_ifdim\_lastskip>0pt
   \_skip0=\_lastskip \_unskip\_italcorr \_hskip\_skip0 \_else\_italcorr \_fi}
\_def\_afteritcorr{\_futurelet\_next\_afteritcorrA}
\_def\_afteritcorrA{\_ifx\_next.\_else\_ifx\_next,\_else \_italcorr \_fi\_fi}
\_let\_italcorr=\/
```

The \boldify macro does \let\it\bi and \let\normalmath=\boldmath.

fonts-opmac.opm

```
\_protected\_def \_boldify {%
   \_let \_setmainvalues=\_setmainvaluesL
   \_let\it =\_bi \_let\rm =\_bf \_let\_normalmath=\_boldmath
   \_let\_it=\_bi \_let\_rm=\_bf \_rm
}
\_public \em \boldify ;
```

We need to use a font selector for default pagination. Because we don't know what default font size will be selected by the user, we use this _rmfixed macro. It sets the \rm font from default font size (declared by first \typosize command and redefines itself be only the font switch for next pages.

fonts-opmac.opm

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

2.17 Output routine

The output routine _optexoutput is similar as in plain TEX. 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 –
  - backrground box, if \pgbackground is non-empty,
  - headline box by _makeheadline, if the \headline is nonempty,
  - \vbox to\vsize of _pagecontents which cosnists of –
    - _pagedest, the page destination pg:⟨*gpageno*⟩ for hyperlinks is created here,
    - \topins box if non-empty (from \topinserts),
    - \box255 with completed vertical material from main vertical mode,
    - _footnoterule and \footins box if nonempty (from \fnote, \footnote),
    - \pgbottomskip (default is 0 pt).
  - footline box by _makefootline, if the \footline is nonempty
- _endoutput which does:
  - increments \pageno using \advancepageno
  - runs output routine repeatedly if \dosuperject is activated.

output.opm

```
\_codedecl \nopagenumbers {Output routine <2020-03-28>} % preloaded in format
```

_optexoutput is default output routine. You can create another...

output.opm

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

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

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

The _regmark tokens list includes accumulated #2 from the \regmacro. Logos and another macros are re-defined here (locally) for their usage in headlines or footlines.

output.opm

```
\_def \_begoutput{\_incr\_gpageno
   \_immediate\_wref\_Xpage{{\the\gpageno}{\_folio}}%
   \_prepoffsets \_the\_regmark} %
\_def \_endoutput{\_advancepageno
   {\_globaldefs=1 \_the\_nextpages \_nextpages={}}%
   \_ifnum\_outputpenalty>-20000 \_else\_dosuperject\_fi
}
\_def \_prepoffsets {}
```

\gpageno counts pages from one in whole document

output.opm

```
\_newcount\_gpageno
\_public \gpageno ;
```

The _completepage is similar what plain TEX does in its output routine. New is only _backgroundbox. It is \vbox with zero height with its contents (from \pgbackground) llaped down. It is shifted directly to the left-upper corner of the paper.

The _ensureblack sets the typesetting of its parameter locally to \Black color. We needn't do this if colors are never used in the document. So, default value of the _ensureblack macro is empty. But first usage of color macros in the document re-defines _ensureblack. See the section 2.19 for more details.

output.opm

```
\_def\_completepage{\_vbox{%
    \_istoksempty \_pgbackground
       \_iffalse \_ensureblack{\_backgroundbox{\_the\_pgbackground}}\_nointerlineskip \_fi
    \_ensureblack{\_makeheadline}%
```

```
      \_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
      \_ensureblack{\_makefootline}}%
}
\_def \_ensureblack #1{#1} % will be re-defined by color macros
\_let \_openfnotestack = \_relax % will be re-defined by color macros
\_def \_backgroundbox #1{\_moveleft\_hoffset\_vbox to0pt{\_kern-\voffset #1\_vss}}
```

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

output.opm

```
\_def\_makeheadline {\_istoksempty \_headline \_iffalse
   \_vbox to0pt{\_vss
                \_baselineskip=\_headlinedist \_lineskiplimit=-\maxdimen
                \_line{\_the\_headline}\_hbox{}}\_nointerlineskip
   \_fi
}
```

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

output.opm

```
\_def\_makefootline{\_istoksempty \_footline \_iffalse
      \_baselineskip=\_footlinedist
      \_lineskiplimit=-\maxdimen \_line{\_the\_footline}
   \_fi
}
```

The _pagecontents is similar as in plain TEX. The only differnece is that the _pagedest is inserted at the top of _pagecontents and _ensureblack is applied to the \topins and \footins material. The _footnoterule is defined here.

output.opm

```
\_def\_pagecontents{\_pagedest % destination of the page
  \_ifvoid\_topins \_else \_ensureblack{\_unvbox\_topins}\_fi
  \_dimen0=\dp255 \_unvbox255 % open up \box255
  \_ifvoid\_footins \_else % footnote info is present
    \_vskip\_skip\_footins
    \_ensureblack{\_footnoterule \_openfnotestack \_unvbox\_footins}\fi
  \kern-\_dimen0 \_vskip \_pgbottomskip
}
\_def \_pagedest {{\_def\_destheight{25pt}\_dest[pg:\_the\_gpageno]}}
\_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 0 pt by default.

output.opm

```
\_countdef\_pageno=0 \_pageno=1 % first page is number 1
\_def \_folio {\_ifnum\_pageno<0 \_romannumeral-\_pageno \_else \_number\_pageno \_fi}
\_def \_nopagenumbers {\_footline={}}
\_def \_advancepageno {%
   \_ifnum\_pageno<0 \_global\_advance\_pageno by-1 \_else \_incr\_pageno \_fi
} % increase |pageno|
\_def \_raggedbottom {\_topskip=\_dimexpr\_topskip plus60pt \_pgbottomskip=0pt plus1fil\_relax}
\_def \_normalbottom {\_topskip=\_dimexpr\_topskip \_pgbottomskip=0pt\_relax}

\_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 a local settings inside the \footins group and it does it when \fnote macro is used.
The _opfootnote nor \vfootnote don't take the footnote text as a parameter. This is due to user can do catcode settings (like inline verbatim) in the footnote text. This idea is adapted from plain TEX.
The \footnote and \footstrut is defined as in plain TEX.

output.opm

```
\_newinsert\_footins
\_def \_footnote #1{\_let\_osf=\_empty % parameter #2 (the text) is read later
   \_ifhmode \_edef\_osf{\_spacefactor\_the\_spacefactor}\/\_fi
  #1\_osf\_vfootnote{#1}}
\_def\_vfootnote{\_opfootnote{}}
\_def \_opfootnote #1#2{\_insert\_footins\_bgroup
  \_interlinepenalty=\_interfootnotelinepenalty
  \_leftskip=0pt \_rightskip=0pt \_spaceskip=0pt \_xspaceskip=0pt \_relax
```

```
  \_let\_colorstackcnt=\_fnotestack % special color stack for footnotes
  #1\_relax % local settings used by \fnote macro
  \_splittopskip=\_ht\_strutbox % top baseline for broken footnotes
  \_splitmaxdepth=\_dp\_strutbox \_floatingpenalty=20000
  \_textindent{#2}\_footstrut
  \_isnextchar \_bgroup
     {\_bgroup \_aftergroup\_vfootA \_afterassignment\_ignorespaces \_let\_next=}{\_vfootB}%
}
\_def\_vfootA{\_unskip\_strut\_isnextchar\_colorstackpop\_closefncolor\_vfootF}
\_def\_vfootB #1{#1\_uskip\_strut\_vfootF}
\_def\_vfootF{\_egroup} % close \_insert\_footins\_bgroup
\_def\_closefncolor#1{#1\_isnextchar\_colorstackpop\_closefncolor\_vfootF}
\_def \_footstrut {\_vbox to\_splittopskip{}}
\_skip\_footins=\_bigskipamount % space added when footnote is present
\_count\_footins=1000 % footnote magnification factor (1 to 1)
\_dimen\_footins=8in % maximum footnotes per page
\_public
   \footins \footnote \vfootnote \footstrut ;
```

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

output.opm

```
\_newinsert\_topins
\_newifi\_ifupage \_newifi\_ifumid
\_def \_topinsert {\_umidfalse \_upagefalse \_oins}
\_def \_midinsert {\_umidtrue \_oins}
\_def \_pageinsert {\_umidfalse \_upagetrue \_oins}
\_skip\_topins=\_zoskip % no space added when a topinsert is present
\_count\_topins=1000 % magnification factor (1 to 1)
\_dimen\_topins=\_maxdimen % no limit per page
\_def \_oins {\_par \_begingroup\_setbox0=\_vbox\_bgroup} % start a \_vbox
\_def \_endinsert {\_par\_egroup % finish the \_vbox
  \_ifumid \_dimen0=\_ht0 \_advance\_dimen0 by\_dp0 \_advance\_dimen0 by\_baselineskip
    \_advance\_dimen0 by\_pagetotal \_advance\_dimen0 by-\_pageshrink
    \_ifdim\_dimen0>\_pagegoal \_umidfalse \_upagefalse \_fi \_fi
  \_ifumid \_bigskip \_box0 \_bigbreak
  \_else \_insert \_topins {\_penalty100 % floating insertion
    \_splittopskip=0pt
    \_splitmaxdepth=\_maxdimen \_floatingpenalty=0
    \_ifupage \_dimen0=\_dp0
    \_vbox to\_vsize {\_unvbox0 \_kern-\_dimen0}% depth is zero
    \_else \_box0 \_nobreak \_bigskip \_fi}\_fi\_endgroup}

\_public \topins \topinsert \midinsert \pageinsert \endinsert ;
```

The \draft macro is an example of usage _pgbackground to create water color marks.

output.opm

```
\_def \_draft {\_pgbackground={\_draftbox{\_draftfont DRAFT}}%
   \_fontdef\_draftfont{\_setfontsize{at10pt}\_bf}%
   \_global\_let\_draftfont=\_draftfont
}
\_def \_draftbox #1{\_setbox0=\_hbox{#1}%
   \_kern.5\_vsize \_kern4.5\_wd0
   \_hbox to0pt{\_kern.5\_hsize \_kern-1\_wd0
   \_pdfsave \_pdfrotate{55}\_pdfscale{10}{10}%
   \_hbox to0pt{\_localcolor\LightGrey \_box0\_hss}%
   \_pdfrestore
   \_hss}%
}
\_public \draft ;
```

2.18 Margins

The \margins macro is documented in the section 1.2.1.

margins.opm

```
\_codedecl \margins {Macros for margins setting <2020-03-14>} % preloaded in format
```

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

margins.opm

```
\_newdimen\_pgwidth  \_newdimen\_pgheight  \_pgwidth=0pt
\_newdimen\_shiftoffset

\_def\_margins/#1 #2 (#3,#4,#5,#6)#7 {\_def\_tmp{#7}%
   \_ifx\_tmp\_empty
      \_opwarning{\_string\_margins: missing unit, mm inserted}\_def\_tmp{mm}\_fi
   \_setpagedimens #2 % setting \_pgwidth, \_pgheight
   \_ifdim\_pgwidth=0pt \_else
      \_hoffset=0pt \_voffset=0pt
      \_if$#3$\_if$#4$\_hoffset =\_dimexpr (\_pgwidth -\_hsize)/2 \_relax
              \_else  \_hoffset =\_dimexpr \_pgwidth -\_hsize - #4\_tmp \_relax % only right margin
              \_fi
      \_else  \_if$#4$\_hoffset = #3\_tmp \_relax  % only left margin
              \_else  \_hsize =\_dimexpr \_pgwidth - #3\_tmp - #4\_tmp \_relax % left+right margin
                      \_hoffset = #3\_tmp \_relax
      \_fi\_fi
      \_if$#5$\_if$#6$\_voffset =\_dimexpr (\_pgheight -\_vsize)/2 \_relax
              \_else  \_voffset =\_dimexpr \_pgheight -\_vsize - #6\_tmp \_relax % only bottom margin
              \_fi
      \_else  \_if$#6$\_voffset = #5\_tmp \_relax  % only top margin
              \_else  \_vsize=\_dimexpr \_pgheight - #5\_tmp - #6\_tmp \_relax % top+bottom margin
                      \_voffset = #5\_tmp \_relax
      \_fi\_fi
      \_if 1#1\_shiftoffset=0pt \_def\_prepoffsets{}\_else \_if 2#1% double-page layout
         \_shiftoffset = \_dimexpr \_pgwidth -\_hsize -2\_hoffset \_relax
         \_def\_prepoffsets{\_ifodd\_pageno \_else \_advance\_hoffset \_shiftoffset \_fi}%
      \_else \_opwarning{use \_string\_margins/1 or \_string\_margins/2}%
   \_fi\_fi\_fi
}
\_def\_setpagedimens{\_isnextchar({\_setpagedimensB}{\_setpagedimensA}}
\_def\_setpagedimensA#1 {\_ifcsname _pgs:#1\_endcsname
   \_ea\_ea\_ea\_setpagedimensB \_csname _pgs:#1\_ea\_endcsname\_space
   \_else \_opwarning{page specification "#1" is undefined}\_fi}
\_def\_setpagedimensB (#1,#2)#3 {\_setpagedimensC\_pgwidth=#1:#3
                                 \_setpagedimensC\_pgheight=#2:#3
        \_pdfpagewidth=\_pgwidth \_pdfpageheight=\_pgheight
}
\_def\_setpagedimensC #1=#2:#3 {#1=#2\_ifx^#3^\_tmp\_else#3\_fi\_relax\_truedimen#1}

\_public \margins ;
```

The common page dimensions are defined here.

margins.opm

```
\_sdef{_pgs:a3}{(297,420)mm}  \_sdef{_pgs:a4}{(210,297)mm}  \_sdef{_pgs:a5}{(148,210)mm}
\_sdef{_pgs:a3l}{(420,297)mm} \_sdef{_pgs:a4l}{(297,210)mm} \_sdef{_pgs:a5l}{(210,148)mm}
\_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.

margins.opm

```
\_def\_trueunit{}
\_def\_magscale[#1]{\_mag=#1\_def\_trueunit{true}%
   \_ifdim\_pgwidth=0pt \_else \_truedimen\_pgwidth \_truedimen\_pgheight \_fi
   \_truedimen\_pdfpagewidth \_truedimen\_pdfpageheight
}
\_def\_truedimen#1{\_ifx\_trueunit\_empty \_else#1=\_ea\_ignorept\_the#1truept \_fi}

\_public \magscale ;
```

2.19 Colors

The colors have different behavior than fonts. A marks (whatsits) with color information are stored into PDF output and TEX doesn't interpret them. The PDF viewer (or PDF interpreter in a printer) reads these marks and switches colors according to them. This is totally independent on TEX group mechanism. You can declare \nolocalcolor at the beginning of the document, if you want this behavior. In this case, if you set a color then you must to return back to black color using \Black manually.

By default, OpTEX sets \localcolor. It means that the typesetting returns back to a previous color at the end of current group, so you cannot write \Black explicitly. This is implemented using

\aftergroup feature. There is a limitation of this feature: when a color selector is used in a group of a box, which is saved by \setbox, then the activity or reconstruction of previous color are processed at \setbox time, no in the box itself. You must correct it by double group:

```
\setbox0=\hbox{\Red text}   % bad:  \Black is done after \setbox
\setbox0=\hbox{{\Red text}} % good: \Black is done after group inside the box
```

The implementation of colors is based on colorstack, so the current color can follow across more pages. It is not so obvious because PDF viewer (or PDF interpreter) manipulates with colors locally at each PDF page and it initializes each PDF page with black on white color.

Macros \setcmykcolor{⟨C⟩ ⟨M⟩ ⟨Y⟩ ⟨K⟩} or \setrgbcolor{⟨R⟩ ⟨G⟩ ⟨B⟩} or \setgreycolor{⟨Grey⟩} should be used in color selectors or user can specify these macros explicitly.

The color mixing processed by the \colordef is done in the substractive color model CMYK. If the result has a component greater than 1 then all components are multiplied by a coefficient in order to maximal component is equal to 1.

You can move a shared amount of CMY components (i.e. their minimum) to the K component. This saves the color tonners and the result is more true. This should be done by \useK command at the end of a linear combination used in \colordef. For example

```
\colordef \myColor {.3\Green + .4\Blue \useK}
```

The \useK command exactly does:

$$k' = \min(C, M, Y),$$
$$C = (C - k') \,/\, (1 - k'),\ M = (M - k') \,/\, (1 - k'),\ Y = (Y - k') \,/\, (1 - k'),$$
$$K = \min(1, K + k').$$

You can use minus instead plus in the linear combination in \colordef. The given color is substracted in such case and the negative components are rounded to zero immediately. For example

```
\colordef \Color {\Brown-\Black}
```

can be used for removing black component from the color. You can use the -\Black trick after \useK command in order to remove grey components occured during color mixing.

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

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

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

If a CMYK to RGB or RGB to CMYK conversion is needed then the following simple formulae are used (ICC profiles are not supported):

CMYK to RGB:
$$R = (1 - C)(1 - K),\ G = (1 - M)(1 - K),\ B = (1 - Y)(1 - K).$$
RGB to CMYK:
$$K' = \max(R, G, B),\ C = (K' - R) \,/\, K',\ M = (K' - G) \,/\, K',\ Y = (K' - B) \,/\, K',\ K = 1 - K'.$$

The RGB to CMYK conversion is invoked when a color is declared using \setrgbcolor and it is used in \colordef or if it is printed when \onlycmyk is declared. The CMYK to RGB conversion is invoked when a color is declared using \setcmykcolor and it is used in \rgbcolordef or if it is printed when \onlyrgb is declared.

colors.opm

```
\_codedecl \colordef {Colors <2020-03-18>} % loaded in format
```

We declare internal boolean value _iflocalcolor ad do \localcolor as default.

colors.opm

```
\_newifi \_iflocalcolor \_localcolortrue
\_protected\_def \_localcolor   {\_localcolortrue}
\_protected\_def \_nolocalcolor {\_localcolorfalse}
\_public \localcolor \nolocalcolor ;
```

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

colors.opm

```
\_def\Blue      {\_setcmykcolor{1 1 0 0}}
\_def\Red       {\_setcmykcolor{0 1 1 0}}
\_def\Brown     {\_setcmykcolor{0 0.67 0.67 0.5}}
\_def\Green     {\_setcmykcolor{1 0 1 0}}
\_def\Yellow    {\_setcmykcolor{0 0 1 0}}
\_def\Cyan      {\_setcmykcolor{1 0 0 0}}
\_def\Magenta   {\_setcmykcolor{0 1 0 0}}
\_def\Grey      {\_setcmykcolor{0 0 0 0.5}}
\_def\LightGrey {\_setcmykcolor{0 0 0 0.2}}
\_def\White     {\_setgreycolor{1}}
\_def\Black     {\_setgreycolor{0}}
```

By default, the \setcmykcolor \setrgbcolor and \setgreycolor macros with {⟨*componetns*⟩} parameter expand to _setcolor{⟨*pdf-primitive*⟩} using _formatcmyk or _formatrgb or _formatgrey expandable macros. For example \setrgbcolor{1 0 0} expands to _setcolor{1 0 0 rg 1 0 0 RG}. We set both types of colors (for lines (K or RG or G) and for fills (r or rg or g) together in the ⟨*pdf-primitive*⟩ command. This is the reason why the _fillstroke uses both its parameters. If only fills are needed you can do \def_fillstroke#1#2{#1}. If only strokes are needed you can do \def_fillstroke#1#2{#2}.

colors.opm

```
\_def\_setcmykcolor#1{\_setcolor{\_formatcmyk{#1}}}
\_def\_setrgbcolor#1{\_setcolor{\_formatrgb{#1}}}
\_def\_setgreycolor#1{\_setcolor{\_formatgrey{#1}}}
\_def\_formatcmyk#1{\_fillstroke{#1 k}{#1 K}}
\_def\_formatrgb#1{\_fillstroke{#1 rg}{#1 RG}}
\_def\_formatgrey#1{\_fillstroke{#1 g}{#1 G}}
\_def\_fillstroke#1#2{#1 #2}
\_public \setcmykcolor \setrgbcolor \setgreycolor ;
```

The \onlyrgb declaration redefines _formatcmyk in order it expands to its conversion to RGB ⟨*pdf-primitive*⟩. This conversion is done by the _cmyktorgb macro. Moreover, \onlyrgb re-defines three basic RGB colors for RGB color space and re-declares \colordef as \rgbcolordef. The \onlycmyk macro does a similar work, it re-defines _formatrgb macro. The Grey color space is unchanged and works in both main settings (RGB or CMYK) without collisions.

colors.opm

```
\_def\_onlyrgb{\_def\Red{\_setrgbcolor{1 0 0}}%
   \_def\Green{\_setrgbcolor{0 1 0}}\_def\Blue{\_setrgbcolor{0 0 1}}%
   \_let\_colordef=\_rgbcolordef
   \_def\_formatrgb##1{\_fillstroke{##1 rg}{##1 RG}}%
   \_def\_formatcmyk##1{\_fillstroke{\_cmyktorgb ##1 ; rg}{\_cmyktorgb ##1 ; RG}}}
\_def\_onlycmyk{\_def\_formatcmyk##1{\_fillstroke{##1 k}{##1 K}}%
   \_def\_formatrgb##1{\_fillstroke{\_rgbtocmyk ##1 ; k}{\_rgbtocmyk ##1 ; K}}}
\_public \onlyrgb \onlycmyk ;
```

The _setcolor macro redefines empty _ensureblack macro (used in output routine for headres and footers) to _ensureblackA which sets Black at the start of its parameter and retunrs to the current color at the end of its parameter. The current color is saved into _currentcolor macro and colorstack is pushed. Finally, the _colorstackpop is initialized by \aftergroup if \localcolor is declared.

You can save current color to your macro by \let\yourmacro=_currentcolor and you can return to this color by the command _setcmykcolor\yourmacro.

colors.opm

```
\_protected\_def \_setcolor #1{\_global\_let\_ensureblack=\_ensureblackA
   \_iflocalcolor \_edef\_currentcolor{#1}\_colorstackpush\_currentcolor
                  \_aftergroup\_colorstackpop
   \_else         \_xdef\_currentcolor{#1}\_colorstackset\_currentcolor \_fi
}
\_def\_pdfblackcolor{0 g 0 G}
\_edef\_currentcolor{\_pdfblackcolor}
\_def\_ensureblackA#1{\_global\_let\_openfnotestack=\_openfnotestackA
   \_colorstackpush\_pdfblackcolor #1\_colorstackpop}
```

The colorstack is initialized here and the basic macros _colorstackpush, _colorstackpop and _colorstackset are defined here.

colors.opm

```
\_mathchardef\_colorstackcnt=0 % Implicit stack usage
```

```
\_def\_colorstackpush#1{\_pdfcolorstack\_colorstackcnt push{#1}}
\_def\_colorstackpop{\_pdfcolorstack\_colorstackcnt pop}
\_def\_colorstackset#1{\_pdfcolorstack\_colorstackcnt set{#1}}
```

We need to open a special color stack for footnotes, because footnotes can follow on next pages and their colors are independent on colors used in the main page-body. The _openfnotestack is defined as _openfnotestackA when the _setcolor is used first. The _fnotestack is initializedin in \everyjob because the initialization is not saved to the format.

colors.opm

```
%\_mathchardef\_fnotestack=\_pdfcolorstackinit page {0 g 0 G} % must be in \everyjob
\_def \_openfnotestackA {\_pdfcolorstack\_fnotestack current}
```

We use lua codes for RGB to CMYK or CMYK to RGB conversions and for addition color components in the \colordef macro. The _rgbtocmyk ⟨*R*⟩ ⟨*G*⟩ ⟨*B*⟩ ; expands to ⟨*C*⟩ ⟨*M*⟩ ⟨*Y*⟩ ⟨*K*⟩ and the _cmyktorgb ⟨*C*⟩ ⟨*M*⟩ ⟨*Y*⟩ ⟨*K*⟩ ; expands to ⟨*R*⟩ ⟨*G*⟩ ⟨*B*⟩. 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.

colors.opm

```
\_def\_rgbtocmyk #1 #2 #3 ;{%
   \_ea \_stripzeros \_detokenize \_ea{\_directlua{
      local kr = math.max(#1,#2,#3)
      if (kr==0) then
         tex.print('0. 0. 0. 1 ;')
      else
         tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
            (kr-#1)/kr, (kr-#2)/kr, (kr-#3)/kr, 1-kr))
      end
}}}
\_def\_cmyktorgb #1 #2 #3 #4 ;{%
   \_ea \_stripzeros \_detokenize \_ea{\_directlua{
      local kr = 1-#4
      tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f ;',
         (1-#1)*kr, (1-#2)*kr, (1-#3)*kr))
}}}
\_def\_colorcrop{\_directlua{
   local m=math.max(color_C, color_M, color_Y, color_K)
   if (m>1) then
      color_C=color_C/m  color_M=color_M/m  color_Y=color_Y/m color_K=color_K/m
   end
}}
\_def\_colordefFin{\_colorcrop \_ea \_stripzeros \_detokenize \_ea{\_directlua{
   tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
      color_C, color_M, color_Y, color_K))
}}}
\_def\_douseK{\_colorcrop \_directlua{
   kr=math.min(color_C, color_M, color_Y)
   if (kr>=1) then
      color_C=0  color_M=0  color_Y=0  color_K=1
   else
      color_C=(color_C-kr)/(1-kr)  color_M=(color_M-kr)/(1-kr)
      color_Y=(color_Y-kr)/(1-kr)  color_K=math.min(color_K+kr,1)
   end
}}
```

We have a problem with the %.3f directive in Lua code. It prints trailed zeros: (0.300 instead desired 0.3) but we want to save PDF file space. The macro _stripzeros removes these trailing zeros at expand processor level. So _stripzeros 0.300 0.400 0.560 ; expands to .3 .4 .56.

colors.opm

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

The \rgbcolordef and \cmykcolordef use common macro _commoncolordef with different first four parameters. The _commoncolordef ⟨*selector*⟩⟨*K*⟩⟨*R*⟩⟨*G*⟩⟨*B*⟩⟨*what-define*⟩{⟨*data*⟩} does the real work. It initializes the Lua variables for summation. It expands ⟨*data*⟩ in the group where color selectors have

special meaning, then it adjusts the resulting string by \replstring and runs it. Example shows how the ⟨*data*⟩ are processed:

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

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

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

colors.opm

```
\_def\_rgbcolordef  {\_commoncolordef \_setrgbcolor  krg}
\_def\_cmykcolordef {\_commoncolordef \_setcmykcolor KRG}
\_def\_commoncolordef#1#2#3#4#5#6{%
  \_begingroup
     \_directlua{color_C=0 color_M=0 color_Y=0 color_K=0}%
     \_def\_setcmykcolor##1{!=#2 ##1 }%
     \_def\_setrgbcolor ##1{!=#3 ##1 }%
     \_def\_setgreycolor##1{!=#4 ##1 }%
     \_let\_useK=\_relax
     \_edef\_tmpb{+#6}%
     \_replstring\_tmpb{+ }{+}\_replstring\_tmpb{- }{-}%
     \_replstring\_tmpb{+}{\_addcolor}\_replstring\_tmpb{-}{\_addcolor-}%
     \_replstring\_tmpb{^!=}{!^}\_replstring\_tmpb{-!}{-1!}%
     \_ifx K#2\_let\_useK=\_douseK \_fi
     \_tmpb
     \_edef\_tmpb{\_colordefFin}%
     \_ifx k#2\_edef\_tmpb{\_ea\_stripK \_tmpb;}\_fi
  \_ea\_endgroup
  \_ea\_def\_ea#5\_ea{\_ea#1\_ea{\_tmpb}}%
}
\_def\_addcolor#1!#2#3{\_cs{addcolor:#2#3}#1}
\_def\_addcolorA #1 #2 #3 #4 #5 {%
   \_def\_tmpa{#1}\_ifx\_tmpa\_empty \_else \_edef\_tmpa{\_tmpa*}\_fi
   \_directlua{color_C=math.max(color_C+\_tmpa#2,0)
               color_M=math.max(color_M+\_tmpa#3,0)
               color_Y=math.max(color_Y+\_tmpa#4,0)
               color_K=math.max(color_K+\_tmpa#5,0)
}}
\_sdef{addcolor:=K}#1 #2 #3 #4 #5 {\_addcolorA #1 #2 #3 #4 #5 }
\_sdef{addcolor:^K}#1 #2 #3 #4 #5 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) #5 }
\_sdef{addcolor:^G}#1 #2 {\_addcolorA #1 0 0 0 #2 }
\_sdef{addcolor:=G}#1 #2 {\_addcolorA #1 0 0 0 (1-#2) }
\_sdef{addcolor:=R}#1 #2 #3 #4 {%
   \_edef\_tmpa{\_noexpand\_addcolorA #1 \_rgbtocmyk #2 #3 #4 ; }\_tmpa
}
\_sdef{addcolor:^R}#1 #2 #3 #4 {\_cs{addcolor:=R}#1 (1-#2) (1-#3) (1-#4) }

\_sdef{addcolor:=k}#1 #2 #3 #4 #5 {%
   \_edef\_tmpa{\_noexpand\_addcolorA #1 \_cmyktorgb #2 #3 #4 #5 ; 0 }\_tmpa
}
\_sdef{addcolor:^k}#1 #2 #3 #4 #5 {\_cs{addcolor:=k}#1 (1-#2) (1-#3) (1-#4) #5 }
\_sdef{addcolor:^g}#1 #2 {\_addcolorA #1 (1-#2) (1-#2) (1-#2) 0 }
\_sdef{addcolor:=g}#1 #2 {\_addcolorA #1 #2 #2 #2 0 }
\_sdef{addcolor:=r}#1 #2 #3 #4 {\_addcolorA #1 #2 #3 #4 0 }
```

```
\_sdef{addcolor:^r}#1 #2 #3 #4 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) 0 }
\_def\_stripK#1 .0;{#1}
\_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

```
\_def \useK{\_useK}
\_def \colordef {\_colordef}
\_public \cmykcolordef \rgbcolordef ;
```

The 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}
\begmulti 4 \typosize[11/14]
\morecolors
\endmulti
```

colors.opm

```
\_def\_morecolors{%
   \_long\_def\_tmp##1\preparecolorset##2##3##4##5{\_tmpa ##5;,,,;}
   \_def\_tmpa##1,##2,##3,##4;{\_ifx,##1,\_else
      \_def\_tmpb{##1}\_replstring\_tmpb{1}{}\_replstring\_tmpb{2}{B}%
      \_replstring\_tmpb{3}{C}\_replstring\_tmpb{4}{D}\_replstring\_tmpb{0}{O}%
      \_ifcsname \_tmpb\_endcsname \_else
          \_sdef{\_tmpb}{\_setrgbcolor{##2 ##3 ##4}}\_showcolor\_fi
      \_ea\_tmpa\_fi
   }
   \_ea\_tmp\_input x11nam.def
}
\_let\_showcolor=\_relax % re-define it if you want to print a color catalog
\_public \morecolors ;
```

2.20 The .ref file

The .ref file has the name \jobname.ref and it saves information about references, TOC lines, etc. All data needed in next TeX run are saved here. OpTeX reads this file at the beginning of the document (using \everyjob) if such file exists. The .ref file looks like:

```
\Xrefversion{⟨ref-version⟩}
\_Xpage{⟨gpageno⟩}{⟨pageno⟩}
\_Xtoc{⟨level⟩}{⟨type⟩}{⟨text⟩}{⟨title⟩}
\_Xlabel{⟨label⟩}{⟨text⟩}
\_Xlabel{⟨label⟩}{⟨text⟩}
...
\_Xpage{⟨gpageno⟩}{⟨pageno⟩}
\_Xlabel{⟨label⟩}{⟨text⟩}
...
```

where ⟨*gpageno*⟩ is internal page number globally numbered from one and ⟨*pageno*⟩ is a page number (\the\pageno) used in pagination (they may be differ). Each page begins with _Xpage. The ⟨*label*⟩ is a label used by user in \label[⟨*label*⟩] and ⟨*text*⟩ is a text which should be referenced (the number of section or table, for example 2.3.14). The ⟨*title*⟩ is a title of the chapter (⟨*level*⟩=1, ⟨*type*⟩=chap), section (⟨*level*⟩=2, ⟨*type*⟩=sec), subsection (⟨*level*⟩=3, ⟨*type*⟩=secc). The _Xpage is written at begining of each page, the _Xtoc is written when chapter or section or subsection title exists on the page and _Xlabel when labeled object prefixed by \label[⟨*label*⟩] exists on the page.

The .ref file is read when the processing of the document starts using \everyjob. It is read, removed and opened to writing immediately. But the .ref file should be missing. If none forward references are needed in the document then .ref file is not created. For example, you only want to test a simple plain TeX macro, you create test.tex file, you do optex test and you don't need to see empty test.ref file in your directory.

ref-file.opm

```
\_codedecl \openref {File for references <2020-02-14>} % preloaded in format
```

The _inputref macro is used in \everyjob. It reads \jobname.ref file if it exists. After the file is read then it is removed and opened to write a new contents to this file.

ref-file.opm

```
\_newwrite\_reffile

\_def\_inputref {%
  \_isfile{\_jobname.ref}\_iftrue
     \_input {\jobname.ref}
     \_gfnotenum=0 \_lfnotenum=0 \_mnotenum=0
     \_openrefA{\_string\_inputref}%
  \_fi
}
```

If the file does not exists then it is not created by default. It means that if you process a document without any forward references then no \jobname.ref file is created because it is unusable. The _wref macro is dummy in such case.

ref-file.opm

```
\_def\_wrefrelax#1#2{}
\_let\_wref=\_wrefrelax
```

If a macro needs to ceate and to use .ref file then such macro must use \openref. When the file is created (using internal _openrefA) then the _wref \⟨*macro*⟩{⟨*data*⟩} is redefined in order to save the line \⟨*macro*⟩⟨*data*⟩ to the .ref file using asynchronous \write primitive. Finally, the _openref destroys itself, because we need not to open the file again.

ref-file.opm

```
\_def\_openref {%
  \_ifx \_wref\_wrefrelax \_openrefA{\_string\openref}\_fi
  \_gdef\_openref{}%
}
\_def\_openrefA #1{%
   \_immediate\_openout\_reffile="\_jobname.ref"\_relax
   \_gdef\_wref ##1##2{\_write\_reffile{\_string##1##2}}%
   \_immediate\_write\_reffile {\_pcent\_pcent\_space OPTeX <\_optexversion> - REF file (#1)}%
   \_immediate\_wref \Xrefversion{{\_REFversion}}%
}
\def\openref{\_openref}
```

We are using convention that the macros used in .ref file are named _X⟨*foo*⟩. If there is a new version of OpTEX with different collection of such macros then we don't want to read the .ref files produced by an old version of OpTEX or by OPmac. So first line of .ref file is in the form

\Xrefversion{⟨*version*⟩}

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

ref-file.opm

```
\_def\_REFversion{3} % actual version of .ref files in OpTeX
\_def\_Xrefversion#1{\_ifnum #1=\_REFversion\_relax \_else \_endinput \_fi}
\_public \Xrefversion ; % we want to ignore .ref files generated by OPmac
```

You cannot define your special .ref macros before .ref file is read because it is read in \everyjob. But you can define such macros using \refdecl{⟨*definitions of your ref macros*⟩}. This command sends to .ref file your ⟨*definitions of your ref macros*⟩ immediately. Next lines in .ref file should include our macros. Example from CTUstyle2:

```
\refdecl{%
   \def\totlist{} \def\toflist{}^^J
   \def\Xtab#1#2#3{\addto\totlist{\totline{#1}{#2}{#3}}}^^J
   \def\Xfig#1#2#3{\addto\toflist{\tofline{#1}{#2}{#3}}}
}
```

We must read ⟨*definition of your ref macros*⟩ when catcode of # is 12 because we needn't to duplicate each # in the .ref file.

ref-file.opm

```
\_def\_refdecl{\bgroup \catcode`\#=12 \_refdeclA}
```

```
\_def\_refdeclA #1{\egroup\_openref
   \_immediate\_write\_reffile {\_pcent\_space \_string \refdecl:}%
   \_immediate\_write\_reffile {\_detokenize{#1}}%
}
\_public \refdecl ;
```

2.21 References

If the references are "forward" (i. e. the `\ref` is used first, the destination is created later) or if the reference text is page number then we must read `.ref` file first in order to get appropriate information. See section 2.20 for more information about `.ref` file concept.

references.opm

```
\_codedecl \ref {References <2020-03-03>} % preloaded in format
```

_Xpage {⟨*gpageno*⟩}{⟨*pageno*⟩} saves the parameter pair into _currpage. Resets `\_lfnotenum`; it is used if footnotes are numbered from one at each page.

references.opm

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

_Xlabel {⟨*label*⟩}{⟨*text*⟩} saves the ⟨*text*⟩ to `\_lab:`⟨*label*⟩ and saves `[pg:`⟨*gpageno*⟩`]{`⟨*pageno*⟩`}` to `\_pgref:`⟨*label*⟩.

references.opm

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

\label[⟨*label*⟩] saves the decalred label to `\_lastlabel` and \wlabel{⟨*text*⟩} uses the `\_lastlabel` and activates `\wref\_Xlabel{`⟨*label*⟩`}{`⟨*text*⟩`}`.

references.opm

```
\_def\_label[#1]{\_isempty{#1}\_iftrue \_global\_let \_lastlabel=\_undefined
  \_else \_isdefined{l0:#1}%
     \_iftrue \_opwarning{duplicated label [#1], ignored}\_else \_xdef\_lastlabel{#1}\_fi
  \_fi \_ignorespaces
}
\_def\_wlabel#1{%
  \_ifx\_lastlabel\_undefined \_else
     \_dest[ref:\_lastlabel]%
     \_printlabel\_lastlabel
     \_edef\_tmp{{\_lastlabel}{#1}}%
     \_ea\_wref \_ea\_Xlabel \_ea{\_tmp}%
     \_sxdef{_lab:\_lastlabel}{#1}\_sxdef{l0:\_lastlabel}{}%
     \_global\_let\_lastlabel=\_undefined
  \_fi
}
\_public \label \wlabel ;
```

\ref[⟨*label*⟩] uses saved `\_lab:`⟨*label*⟩ and prints (linked) ⟨*text*⟩. If the reference is backwarded then we know `\lab:`⟨*label*⟩ without any need to read REF file. On the other hand, if the reference is forwarded, then we doesn't know `\_lab:`⟨*label*⟩ in first run of TEX and we print warning and do `\_openref`.
\pgref[⟨*label*⟩] uses {⟨*gpageno*⟩}{⟨*pageno*⟩} from `\_pgref:`⟨*label*⟩ and prints (linked) ⟨*pageno*⟩ using `\_ilink` macro.

references.opm

```
\_def\_ref[#1]{\_isdefined{_lab:#1}%
  \_iftrue \_ilink[ref:#1]{\_csname _lab:#1\_endcsname}%
  \_else ??\_opwarning{label [#1] unknown. Try to TeX me again}\_openref
  \_fi
}
\_def\_pgref[#1]{\isdefined{_pgref:#1}%
  \_iftrue \_ea\_ea\_ea\_ilink \_csname _pgref:#1\_endcsname
  \_else ??\_opwarning{pg-label [#1] unknown. Try to TeX me again}\_openref
  \_fi
}
\_public \ref \pgref ;
```

Default _printlabel is empty macro (labels are not printed). The \showlabels redefines it as box with zero dimensions and with left lapped [⟨*label*⟩] in blue 10pt `\tt` font shifted up by 1.7ex.

references.opm

```
\_def\_printlabel#1{}
```

```
\_def\_showlabels {%
   \_def\_printlabel##1{\_vbox to0pt{\_vss\_llap{\_labelfont[##1]}\_kern1.7ex}}%
   \_fontdef\_labelfont{\_setfontsize{at10pt}\setfontcolor{blue}\_tt}
}
\_public \showlabels ;
```

2.22 Hyperlinks

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

- `ref:`⟨*label*⟩ – the destination is created when `\label[`⟨*label*⟩`]` is used, see also the section 2.21.
- `toc:`⟨*tocrefnum*⟩ – the destination is created at chap/sec/secc titles, see also the section 2.23.
- `pg:`⟨*gpageno*⟩ – the destination is created at beginning of each page, see also the section 2.17.
- `cite:`⟨*bibnum*⟩ – the destination is created in bibliography reference, see also the section 2.31.1.
- `url:`⟨*url*⟩ – used by `\url` or `\ulink`, see also the end of this section.

The ⟨*tocrefnum*⟩, ⟨*gpageno*⟩ and ⟨*bibnum*⟩ are numbers starting from one and globally incremented by one in whole document. The registers `\tocrefnum`, `\gpageno` and `\bibnum` are used for these numbers.

When a chap/sec/secc title is prefixed by `\label[`⟨*label*⟩`]`, then both types of internal links are created at the same destination place: `toc:`⟨*tocrefnum*⟩ and `ref:`⟨*label*⟩.

hyperlinks.opm

```
\_codedecl \ulink {Hyperlinks <2020-04-22>} % preloaded in format
```

`\dest[`⟨*type*⟩`:`⟨*spec*⟩`]` creates a destination of internal links. The destination is declared in the format ⟨*type*⟩`:`⟨*spec*⟩. If the `\hyperlinks` command in not used, then `\dest` does nothing else it is set to `\_destactive`. The `\_destactive` is implemented by `\_pdfdest` primitive. It creates a box in which the destination is shifted by `\_destheight`. The reason is that the destination is exactly at top border of the PDF viewer but we want to see the line where destination is. The destination box is positioned by different way dependent on current vertical or horizontal mode.

hyperlinks.opm

```
\_def\_destheight{1.4em}
\_def\_destactive[#1:#2]{\_if$#2$\_else\_ifvmode
      \_tmpdim=\_prevdepth \_prevdepth=-1000pt
      \_destbox[#1:#2]\_prevdepth=\_tmpdim
   \_else \_destbox[#1:#2]%
   \_fi\_fi
}
\_def\_destbox[#1]{\_vbox to0pt{\_kern-\_destheight \_pdfdest name{#1} xyz\_vss}}
\_def\_dest[#1]{}
\_public \dest ;
```

`\link[`⟨*type*⟩`:`⟨*spec*⟩`]{`⟨*color*⟩`}{`⟨*text*⟩`}` creates an internal link to `\dest` with the same ⟨*type*⟩`:`⟨*spec*⟩. You can have more links with the same ⟨*type*⟩`:`⟨*spec*⟩ but only one `\dest` in the document. If `\hyperlinks` command is not used, then `\link` only prints ⟨*text*⟩ else it is set to `\_linkactive`. The `\_linkactive` is implemented by `\_pdfstartlink...\_pdfendlink` primitives.
`\ilink[`⟨*type*⟩`:`⟨*spec*⟩`]{`⟨*text*⟩`}` is equivalent to `\_link` but the ⟨*color*⟩ is used from `\hyperlinks` declaration.

hyperlinks.opm

```
\_protected\_def\_linkactive[#1:#2]#3#4{\_leavevmode\_pdfstartlink height.9em depth.3em
      \_pdfborder{#1} goto name{#1:#2}\_relax {#3#4}\_pdfendlink
}
\_protected\_def\_link[#1]#2#3{\_leavevmode{#3}}
\_protected\_def\_ilink[#1]#2{\_leavevmode{#2}}
\_public \ilink \link ;
```

`\ulink[`⟨*url*⟩`]{`⟨*text*⟩`}` creates external link. It prints only te ⟨*text*⟩ by default but the `\hyperlinks` declaration defines it as `\_urlactive[url:`⟨*url*⟩`]{`⟨*text*⟩`}`. The external link is created by the `\_pdfstartlink...\pdfendlink` primitives. The ⟨*url*⟩ is detokenized with `\escapechar=-1` before it is used, so `\%`, `\#` etc. can be used in the ⟨*url*⟩.

hyperlinks.opm

```
\_protected\_def\_urlactive[#1:#2]#3#4{\_leavevmode{\_escapechar=-1
   \_pdfstartlink height.9em depth.3em \_pdfborder{#1}%
      user{/Subtype/Link/A <</Type/Action/S/URI/URI(\_detokenize{#2})>>}\relax
      {#3#4}\pdfendlink}%
}
```

```
\_def\_ulink[#1]#2{\_leavevmode{#2}}
\_def\_urlcolor{}
\_public \ulink ;
```

The `\_pdfstartlink` primitive uses `\_pdfborder{`⟨*type*⟩`}` in its parameter (see `\_linkactive` or `\_urlactive` macros). The `\_pdfborder{`⟨*type*⟩`}` expands to `attr{/C[? ? ?] /Border[0 0 .6]}` if the `\`⟨*type*⟩`border` (i.e. `\refborder`, `\citeborder`, `\tocborder`, `\pgborder`, `\urlborder`, `\fntborder` or `\fnfborder`) is defined. User can define it in order to create colored frames around active links. For example `\def\tocborder{1 0 0}` causes red frames in TOC (not printed, only visible in PDF viewers).

hyperlinks.opm

```
\_def\_pdfborder#1{\_if^#1^\_else \_isdefined{#1border}\_iftrue
   \_if^\_csname #1border\_endcsname^\_else
      attr{/C[\_csname #1border\_endcsname] /Border[0 0 .6]}\_fi
   \_else attr{/Border[0 0 0]}\_fi\_fi
}
```

`\hyperlinks{`⟨*ilink_color*⟩`}{`⟨*ulink_color*⟩`}` activates `\dest`, `\link`, `\ilink`, `\ulink` in order they create links. These macros are redefined here to their "active" version.

hyperlinks.opm

```
\_def\_hyperlinks#1#2{%
   \_let\_dest=\_destactive \_let\_link=\_linkactive
   \_def\_ilink[##1]##2{\_link[##1]{\_localcolor#1}{##2}}%
   \_def\_ulink[##1]##2{\_urlactive[url:##1]{\_localcolor#2}{##2}}%
   \_public \dest \ilink \ulink ;%
}
\_public \hyperlinks ;
```

`\url{`⟨*url*⟩`}` does approximately the same as `\ulink[`⟨*url*⟩`]{`⟨*url*⟩`}`, but more work is done before the `\ulink` is processed. The link-version of ⟨*url*⟩ is saved to `\_tmpa` and the printed version in `\_tmpb`. The printed version is modified in order to set a breakpoints to special places of the ⟨*url*⟩. For example `//` is replaced by `\_urlskip/\_urlskip/\_urlbskip` where `\urlskip` adds a small nobreakable glue between these two slashes and before them and `\_urlbskip` adds a breakable glue after them.
The text version of the ⟨*url*⟩ is printed in `\_urlfont`.

hyperlinks.opm

```
\_def\_url#1{{%
   \_def\_tmpa{#1}\_replstring\_tmpa {\|}{}%
   {\_escapechar=-1 \_ea}\_ea\_edef\_ea\_tmpa\_ea{\detokenize\_ea{\_tmpa}}%
   \_def\_tmpb{#1}\_replstring\_tmpb {\|}{\_urlbskip}%
   \_replstring\_tmpb {//} {{\_urlskip\_urlslashslash\_urlbskip}}%
   \_replstring\_tmpb {/}  {{\_urlskip/\_urlbskip}}%
   \_replstring\_tmpb {.}  {{\_urlskip.\_urlbskip}}%
   \_replstring\_tmpb {?}  {{\_urlskip?\_urlbskip}}%
   \_replstring\_tmpb {=}  {{\urlskip=\urlbskip}}%
   \_ea\_replstring\_ea\_tmpb \_ea{\_string &} {{\urlbskip\char`\& \urlskip}}%
   \_ea\_replstring\_ea\_tmpb \_ea{\_bslash|}  {{\_penalty0}}%
   \_ea\_ulink \_ea[\_tmpa] {\_urlfont\_tmpb\_null}%
}}
\_def\_urlfont{\_tt}
\_def\_urlskip{\_null\_nobreak\_hskip0pt plus0.05em\_relax}
\_def\_urlbskip{\_penalty100 \_hskip0pt plus0.05em\_relax}
\_def\_urlslashslash{/\_urlskip/}

\_public \url ;
```

2.23 Making table of contents

maketoc.opm

```
\_codedecl \maketoc {Macros for maketoc <2020-03-12>} % preloaded in format
```

`\_Xtoc {`⟨*level*⟩`}{`⟨*type*⟩`}{`⟨*number*⟩`}{`⟨*title*⟩`}` (in `.ref` file) reads the specified data and appends them to the `\_toclist` as `\_tocline{`⟨*level*⟩`}{`⟨*type*⟩`}{`⟨*number*⟩`}{`⟨*title*⟩`}{`⟨*gpageno*⟩`}{`⟨*pageno*⟩`}` where:

- ⟨*level*⟩: 0 reserved, 1: chapter, 2: section, 3: subsection
- ⟨*type*⟩: the type of the level, i.e. chap, sec, secc
- ⟨*number*⟩: the number of the chapter/section/subsection in the format 1.2.3
- ⟨*title*⟩: the title text
- ⟨*gpageno*⟩: the page number numbered from 1 independently of pagination
- ⟨*pageno*⟩: the page number used in the pagination

The last two parameters are restored from previous _Xpage{⟨*pageno*⟩}{⟨*gpageno*⟩}, data were saved in the _currpage macro.

maketoc.opm

```
\_def\_toclist{}
\_newifi \_ifischap \_ischapfalse

\def\_Xtoc#1#2#3#4{\_ifnum#1=0 \_ischaptrue\_fi
   \_addto\_toclist{\_tocline{#1}{#2}{#3}{#4}}
   \_ea\_addto\_ea\_toclist\_ea{\_currpage}%
}
```

_tocline{⟨*level*⟩}{⟨*type*⟩}{⟨*number*⟩}{⟨*title*⟩}{⟨*gpageno*⟩}{⟨*pageno*⟩} prints the record to the table of contents. It opens group, reduces _leftskip, _rightskip, runs the \everytocline (user can customise the design of TOC here) and runs _tocl:⟨*level*⟩ {⟨*number*⟩}{⟨*title*⟩}{⟨*pageno*⟩} macro. This macro starts with vertical mode, inserts one record with given ⟨*level*⟩ and it should end by _tocpar which returns to hrizontal 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:

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

maketoc.opm

```
\_newcount \_tocrefnum
\_def\_tocline#1#2#3#4#5#6{%
   \_advance\_tocrefnum by1
   \_bgroup
      \_leftskip=\_iindent \_rightskip=2\_iindent
      \_ifischap \_advance\_leftskip by \_iindent \_fi
      \_def\_pgn{\_ilink[pg:#5]}%
      \_the\_everytocline
      \_ifcsname _tocl:#1\_endcsname
         \_cs{_tocl:#1}{#3}{#4}{#6}\_par
      \_fi
   \_egroup
}
\_public \tocrefnum ;
```

You can re-define default macros for each level of tocline if you want.
Parameters are {⟨*number*⟩}{⟨*title*⟩}{⟨*pageno*⟩}.

maketoc.opm

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

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 recors whith rlapped ⟨*pageno*⟩.
- _pgn{⟨*pageno*⟩} creates ⟨*pageno*⟩ as link to real ⟨*gpage*⟩ saved in #6 of _tocline. This is temporarily defined in the _tocline.

maketoc.opm

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

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

maketoc.opm

```
\_def\_maketoc{\_par \_ifx\_toclist\_empty
```

```
      \_opwarning{\_noexpand\maketoc -- data unavailable, TeX me again}\_openref
    \_else \_begingroup
      \_tocrefnum=0 \_penalty11333
      \_the\_regtoc \_toclist
    \_endgroup \_fi
}
```

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

maketoc.opm

```
\_newtoks \_regtoc  \_newtoks \_regmark  \_newtoks \_regoul

\_def\_regmacro #1#2#3{%
   \_toksapp\_regtoc{#1}\_toksapp\_regmark{#2}\_toksapp\_regoul{#3}%
}
\_public \maketoc \regmacro ;
```

2.24 PDF outlines

2.24.1 Nesting PDF outlines

The problem is that PDF format needs to know the number of direct descendants of each outline if we need to create the tree of structured outlines. But we know only the level of each outline. The required data should be calculated from TOC data. We use two steps over TOC data saved in the `\_toclist` where each record is represented by one `\_tocline`.

First step, the \outlines macro sets `\_tocline` to _outlinesA and calculates the number of direct descendants of each record. Second step, the `\outlines` macro sets `\_tocline` to _outlinesB and it uses prepared data and create outlines.

Each ouline is mapped to the control sequence of the type `\_ol:`⟨*num*⟩ or `\_ol:`⟨*num*⟩`:`⟨*num*⟩ or `\_ol:`⟨*num*⟩`:`⟨*num*⟩`:`⟨*num*⟩ or etc. The first one is reserved for level 0, the second one for level 1 (chapters), third one for level 2 (sections) etc. The number of direct descentants will be stored in these macros after first step is finished. Each new outline of given level increases the ⟨*num*⟩ at given level. When the first step is processed then (above that) the `\_ol:..` sequence of the parent inceases its value too. The `_ol:...` sequences are implemented by `\_ol:\_count0:\_count1:\count2` etc. For example, when section (level 2) is processed in the first step then we do:

```
\advance \count2 by 1
                % increases the mapping pointer of the type
                % \_ol:<\_count0:\_count1:\_count2> of this section
\advance \<_ol:\_count0:\_count1> by 1
                % increases the number of descendants connected
                % to the parent of this section.
```

When second step is processed, then we only read the stored data about the number of descentants. Ad we use it in `count` parameter of `\_pdfoutline` primitive.

For linking, we use the same links as in TOC, i.e. the `toc:\_the\_tocrefnum` labels are used.

\insertoutline

`{`⟨*text*⟩`}` inserts one outline with zero direct descendants. It creates link destination of the type `oul:`⟨*num*⟩ into the document (where \insertoutline is used) and the link itself is created too in the outline.

outlines.opm

```
\_codedecl \outlines {PDF outlines <2020-03-12>} % preloaded in format

\_def\_outlines#1{\_pdfcatalog{/PageMode/UseOutlines}\_openref
   \_ifx\_toclist\_empty
     \_opwarning{\noexpand\outlines -- data unavailable. TeX me again}%
   \_else
     \_ifx\_dest\_destactive \_else
        \opwarning{\_noexpand\outlines doesn't work when \_noexpand\hyperlinks isn't declared}\_fi
     {\_let\_tocline=\_outlinesA
      \_count0=0 \_count1=0 \_count2=0 \_count3=0 \_toclist % calculate numbers o childs
      \_def\_outlinelevel{#1}\_let\_tocline=\_outlinesB
      \_tocrefnum=0 \_count0=0 \_count1=0 \_count2=0 \_count3=0
      \_toclist}% create outlines
```

```
   \_fi
}
\_def\_outlinesA#1#2#3#4#5#6{%
   \_advance\_count#1 by1
   \_ifcase#1\_or
      \_addoneol{_ol:\_the\_count0}\_or
      \_addoneol{_ol:\_the\_count0:\_the\_count1}\_or
      \_addoneol{_ol:\_the\_count0:\_the\_count1:\_the\_count2}\_or
      \_addoneol{_ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}\_fi
}
\_def\_addoneol#1{%
   \_ifcsname #1\_endcsname
           \_tmpnum=\_csname#1\_endcsname\_relax
           \_advance\_tmpnum by1 \_sxdef{#1}{\_the\_tmpnum}%
   \_else \_sxdef{#1}{1}%
   \_fi
}
\_def\_outlinesB#1#2#3#4#5#6{%
   \_advance\_count#1 by1
   \_ifcase#1%
      \_tmpnum=\_trycs{_ol:\_the\_count0}{0}\_or
      \_tmpnum=\_trycs{_ol:\_the\_count0:\_the\_count1}{0}\_relax\_or
      \_tmpnum=\_trycs{_ol:\_the\_count0:\_the\_count1:\_the\_count2}{0}\_relax\_or
      \_tmpnum=\_trycs{_ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}{0}\_relax\_or
      \_tmpnum = 0\_relax\_fi
   \_pdfunidef\_tmp{#4}%
   \_advance\_tocrefnum by1
   \_outlinesC{#1}{toc:\_the\_tocrefnum}{\_ifnum#1<\_outlinelevel\_space\_else-\_fi}{\_tmpnum}{\_tmp}%
}
\_def\_outlinesC#1#2#3#4#5{\_pdfoutline goto name{#2} count #3#4{#5}\_relax}

\_newcount\_oulnum
\_def\_insertoutline#1{\_global\_advance\_oulnum by1
   \_pdfdest name{oul:\_the\_oulnum} xyz\_relax
   \_pdfunidef\_tmp{#1}%
   \_pdfoutline goto name{oul:\_the\_oulnum} count0 {\_tmp}\_relax
}
\_public \outlines \insertoutline ;
```

2.24.2 Strings in PDF outlines

There are only two encodings for PDF strings (used in PDFoutlines, PDFinfo etc.). First one is PDFDocEncoding which is one-byte encoding, but most Czech or Slovak characters are missing here.

The second encoding is PDFunicode encoding wich is implemented in this file. This encoding is TEX-discomfortable, because it looks like

```
\376\377\000C\000v\000i\001\015\000e\000n\000\355\000\040\000j\000e\000\040
\000z\000\341\000t\001\033\001\176
```

This example is real encoding of the string "Cvičení je zátěž". You can see that this is UTF-16 encoding (two bytes per character) with two starting bytes FEFF. Moreover, each byte is encoded by three octal digits preceded by backslash. The only exception is the visible ASCII character encoding: such a character is encoded by its real byte preceded by `\000`.

The command `\pdfunidef\macro{string}` is implemented here using `\directlua`. The input string is preprocessed: detokenized, converted `\word /` to `\word/` (used in logos) preprocessed spaces using `\replstring` and then the `\_pdfunidefB` is repeated on each character. It calls the `\directlua` chunk to print octal numbers in the macro `\_octalprint`.

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

pdfuni-string.opm

```
\_codedecl \pdfunidef {PDFunicode strings for outlines <2020-03-12>} % preloaded in format

\_edef\_octalprint#1#2{\_noexpand\_directlua{% #1=character-code #2=character
    if ('#2'>='A' and '#2'<='Z') or ('#2'>='a' and '#2'<='z') then
      tex.print(string.format('000\_pcent s',"#2"))
    else
      local num=#1\_pcent256
```

```
      tex.print(string.format('\_pcent 03o\_nbb\_pcent03o',(#1-num)/256,num))
    end
}}
\_def\_pdfunidef#1#2{%
   \_begingroup
      \_catcode`\\=12 \_let\\=\_bslash
      \_the\_regoul \_relax % \_regmacro alternatives of logos etc.
      \_escapechar=-1
      \_edef#1{#2\empty}%
      \_escapechar=`\\
      \_ea\_edef \_ea#1\_ea{\_ea\_removeoutmath   #1$\_end$}%  $x$ -> x
      \_ea\_edef \_ea#1\_ea{\_ea\_removeoutbraces #1{\_end}}%  {x} -> x
      \_edef#1{\_detokenize\_ea{#1}}%
      \_replstring#1{ }{{ }}%  text text -> text{ }text
      \_edef\_out{\\376\\377}%
      \_ea\_pdfunidefB#1^%  text -> \_out in octal
      \_ea
   \_endgroup
   \_ea\_def\_ea#1\_ea{\_out}
}
\_def\_pdfunidefB#1{%
   \_ifx^#1\_else
      \_tmpnum=`#1
      \_pdfunidefC{\_luaescapestring{#1}}%
   \_ea\_pdfunidefB \_fi
}
\_def\_pdfunidefC #1{\_edef\_out{\_out \\\_ea\_octalprint\_ea{\_the\_tmpnum}{#1}}}

\_def\_removeoutbraces #1#{#1\_removeoutbracesA}
\_def\_removeoutbracesA #1{\_ifx\_end#1\_else #1\_ea\_removeoutbraces\_fi}
\_def\_removeoutmath #1$#2${#1\_ifx\_end#2\_else #2\_ea\_removeoutmath\_fi}

\_regmacro {}{}{\_let\em=\_empty \_let\rm=\_empty \_let\bf=\_empty
    \_let\it=\_empty \_let\bi=\_empty \_let\tt=\_empty \_let\/=\_empty
    \_let~=\_space
}
\public \pdfunidef ;
```

2.25 Chapters, sections, subsections

sections.opm

```
\_codedecl \chap {Titles, chapters, sections, subsections <2020-03-14>} % preloaded in format
```

We are using scaled fonts for titles _titfont, _chapfont, _secfont and _seccfont. They are scaled from main fonts size of the document, which is declared by first `\typosize[`⟨*fo-size*⟩`/`⟨*b-size*⟩`]` command.

sections.opm

```
\_def \_titfont  {\_scalemain\_boldify\_typoscale[\_magstep4/\_magstep5]}
\_def \_chapfont {\_scalemain\_boldify\_typoscale[\_magstep3/\_magstep3]}
\_def \_secfont  {\_scalemain\_boldify\_typoscale[\_magstep2/\_magstep2]}
\_def \_seccfont {\_scalemain\_boldify\_typoscale[\_magstep1/\_magstep1]}
```

The \tit macro is defined by `\eoldef`, it means that the parameter is separated by end of line. The macros `\chap`, `\sec` and `\secc` use `\eoldef` too.

sections.opm

```
\_eoldef\_tit #1{\_vglue\_titskip
  {\_leftskip=0pt plus1fill \_rightskip=\_leftskip % centering
   \_titfont \_noindent #1\_par}%
   \_nobreak\_bigskip
}
\_public \tit ;
```

You can re-define _printchap, _printsec or _printsecc macros if another design of section titles is needed. These macros gets 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 of such behavior: we don't want to duplicate vertical skip and we don't want to use negative penalty in such cases. Moreover, `\_abovetitle{`⟨*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 of 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 o title is processed. If `\nonum` prefix is used then `\_printrefnum` prints nothing. The macro `\_printrefnum` does more work: it creates destination of hyperlinks (if `\hyperlinks{}{}` is used) and saves references from label (if `\label[`⟨*label*⟩`]` precedes) and saves references for 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 to more pages.
- You can use `\_firstnoindent` in order to the first paragraph after the title is not indented.

sections.opm

```
\_def\_printchap #1{\_vfill\_supereject
   {\_chapfont \_noindent \_mtext{chap} \_printrefnum[@]\_par
    \_nobreak\_smallskip
    \_noindent \_raggedright #1\_nbpar}\_mark{}%
   \_nobreak \_belowtitle{\_bigskip}%
   \_firstnoindent
}
\_def\_printsec#1{\_par
   \_abovetitle{\_penalty-400}\_bigskip
   {\_secfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbpar}\_insertmark{#1}%
   \_nobreak \_belowtitle{\_medskip}%
   \_firstnoindent
}
\_def\_printsecc#1{\_par
   \_abovetitle{\_penalty-200}\_medskip
   {\_seccfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbpar}%
   \_nobreak \_belowtitle{\_medskip}%
   \_firstnoindent
}
```

The `\_sectionlevel` is the level of the printed section:

- `\_sectionlevel=0` – reserved for parts of the book (unused by default)
- `\_sectionlevel=1` – chapters (used in `\chap`)
- `\_sectionlevel=2` – sections (used in `\sec`)
- `\_sectionlevel=3` – subsections (used in `\secc`)
- `\_sectionlevel=4` – subsubsections (unused by default)

sections.opm

```
\_newcount\_sectionlevel
\_def \_secinfo {\_ifcase \_sectionlevel
   part\_or chap\_or sec\_or secc\_or seccc\_fi
}
```

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, figuers and display maths are numbered from one in each section – subsetions doesn't reset these counters. Footnotes declared by `\fnotenumchapters` are numbered in each chapter from one.

The `\_the*` macros `\_thechapnum`, `\_thesecnum`, `\_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 ⟨*num*⟩

and subsections ⟨*num*⟩.⟨*num*⟩. On the other hand, if chapter is used in the document then `\_chapnum>0` and sections have numbers ⟨*num*⟩.⟨*num*⟩ and subsections have numbers ⟨*num*⟩.⟨*num*⟩.⟨*num*⟩.

sections.opm

```
\_newcount \_chapnum  % chapters
\_newcount \_secnum   % sections
\_newcount \_seccnum  % subsections
\_newcount \_tnum     % table numbers
\_newcount \_fnum     % figure numbers
\_newcount \_dnum     % numbered display maths

\_def \_chapx {\_secx  \_secnum=0  \_lfnotenum=0 }
\_def \_secx  {\_seccx \_seccnum=0 \_tnum=0 \_fnum=0 \_dnum=0 \_resetABCDE }
\_def \_seccx {}

\_def \_thechapnum {\_the\_chapnum}
\_def \_thesecnum  {\_othe\_chapnum.\_the\_secnum}
\_def \_theseccnum {\_othe\_chapnum.\_the\_secnum.\_the\_seccnum}
\_def \_thetnum    {\_othe\_chapnum.\_the\_secnum.\_the\_tnum}
\_def \_thefnum    {\_othe\_chapnum.\_the\_secnum.\_the\_fnum}
\_def \_thednum    {(\_the\_dnum)}

\_def\_othe #1.{\_ifnum#1>0 \_the#1.\_fi}
\_def\_incr #1{\_global\_advance#1by1 }
```

The `\notoc` and `\nonum` prefixes are implemented by internal `\_ifnotoc` and `\_ifnonum`. They are reset after each chapter/section/subsection by the `\_resetnonumnotoc` macro.

sections.opm

```
\_newifi \_ifnotoc  \_notocfalse  \_def\_notoc {\_global\_notoctrue}
\_newifi \_ifnonum  \_nonumfalse  \_def\_nonum {\_global\_nonumtrue}
\_def \_resetnonumnotoc{\_global\_notocfalse \_global\_nonumfalse}
\_public \notoc \nonum ;
```

The `\chap`, `\sec` and `\secc` macros are implemented here. The `\_inchap`, `\_insec` and `\_insecc` macros does the real work, First, we read the optional parameter `[`⟨*label*⟩`]`, if it exists.

sections.opm

```
\_optdef\_chap[]{\_trylabel \_inchap}
\_optdef\_sec []{\_trylabel \_insec}
\_optdef\_secc[]{\_trylabel \_insecc}
\_def\_trylabel{\_istoksempty\_opt\_iffalse \_label[\_the\_opt]\_fi}

\_eoldef\_inchap #1{\_par \_sectionlevel=1
   \_def \_savedtitle {#1}% saved to .ref file
   \_ifnonum \_else {\_globaldefs=1 \_incr\_chapnum \_chapx}\_fi
   \_edef \_therefnum {\_ifnonum \_space \_else \_thechapnum \_fi}%
   \_printchap{#1}%
   \_resetnonumnotoc
}
\_eoldef\_insec #1{\_par \_sectionlevel=2
   \_def \_savedtitle {#1}% saved to .ref file
   \_ifnonum \_else {\_globaldefs=1 \_incr\_secnum \_secx}\_fi
   \_edef \_therefnum {\_ifnonum \_space \_else \_thesecnum \_fi}%
   \_printsec{#1}%
   \_resetnonumnotoc
}
\_eoldef\_insecc #1{\_par \_sectionlevel=3
   \_def \_savedtitle {#1}% saved to .ref file
   \_ifnonum \_else {\_globaldefs=1 \_incr\_seccnum \_seccx}\_fi
   \_edef \_therefnum {\_ifnonum \_space \_else \_theseccnum \_fi}%
   \_printsecc{#1}%
   \_resetnonumnotoc
}
\_public \chap \sec \secc ;
```

The `\_printrefnum[`⟨*pre*⟩`@`⟨*post*⟩`]` macro is used in `\_print*` macros.
The `\_wtotoc {`⟨*level*⟩`}{`⟨*info*⟩`}{`⟨*ref-num*⟩`}{`⟨*title-text*⟩`}` macro expands its parameters and does `\_wref`.

Note that the ⟨*tite-text*⟩ is `\detokenized` before `\_wref`, so the problem of "fragile macros" from old LaTeX never occurs.

sections.opm

```
\_def \_printrefnum [#1@#2]{\_leavevmode % we must be in horizontal mode
```

```
  \_ifnonum \_else #1\_therefnum #2\_fi
  \_wlabel \_therefnum  % references, if `\label[<label>]` is declared
  \_ifnotoc \_else \_incr \_tocrefnum
     \_dest[toc:\_the\_tocrefnum]%
     \_wtotoc{\_the\_sectionlevel}{\_secinfo}%
            {\_therefnum}{\_detokenize\_ea{\_savedtitle}}%
  \_fi
}
\_def \_wtotoc #1#2#3#4{\_edef\_tmp{{#1}{#2}{#3}{#4}}\_ea\_wtotocA\_tmp}
\_def \_wtotocA #1#2#3#4{\_wref\_Xtoc{{#1}{#2}{#3}{#4}}}
```

The _abovetitle{⟨*penaltyA*⟩}{⟨*skipA*⟩} and _belowtitle{⟨*skipB*⟩} pair communicates using a special penalty 11333 in vertical mode. The _belowtitle puts the vertical skip (its value is saved in _savedtitleskip) followed by this special penalty. The _abovetitle reads \lastpenalty and if it has this special value then it removes the skip used before and don't use the parameter. The \abovetitle creates ⟨*skipA*⟩ only if whatever previous skip is less or equal than ⟨*skipA*⟩. We must save ⟨*whatever-skip*⟩, remove it, create ⟨*penaltyA*⟩ (if _belowtitle does not preceded) and create ⟨*whatever-skip*⟩ or ⟨*skipA*⟩ depending on what is greater. The amount of ⟨*skipA*⟩ is measured using \setbox0=\vbox.

sections.opm

```
\_newskip \_savedtitleskip
\_newskip \_savedlastskip
\_def\_abovetitle #1#2{\_savedlastskip=\_lastskip % <whatever-skip>
   \_ifdim\_lastskip>0pt \_vskip-\_lastskip \_fi
   \_ifnum\_lastpenalty=11333 \_vskip-\_savedtitleskip \_else #1\_fi
   \_ifdim\_savedlastskip>0pt \_setbox0=\_vbox{#2\_global\_tmpdim=\_lastskip}%
   \_else \_tmpdim=\_maxdimen \_fi
   \_ifdim\_savedlastskip>\_tmpdim \vskip\_savedlastskip \_else #2\_fi
}
\_def\_belowtitle #1{#1\_global\_savedtitleskip=\_lastskip \_penalty11333 }
```

\nbpar sets \interlinepenaty value. \nl is "new line" in text (or titles), but space in toc or headlines or outlines.

sections.opm

```
\_def\_nbpar{{\_interlinepenalty=10000\_endgraf}}

\_protected\def\_nl{\hfil\break}
\_regmacro {\def\_nl{\_unskip\_space}} {\def\_nl{\_unskip\_space}} {\def\_nl{ }}
\_regmacro {\def\nl{\_unskip\_space}}  {\def\nl{\_unskip\_space}}  {\def\nl{ }}

\_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_firtnoindent=\relax. The _wipeepar removes the material from \everypar.

sections.opm

```
\_def \_firstnoindent {\_global\_everypar={\_wipeepar \_setbox7=\_lastbox}}
\_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 user can implement chapter titles for running headers directly by macros, no \mark mechanism is needed. But sections need \marks. And they can be mixed with chapter's running heads, of course.

The _insermark{⟨*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 formating macro with two parameters for processing these data, if you need it.

sections.opm

```
\_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:

```
\addto\_chapx {\_edef\_runningchap {\_thechapnum: \_unexpanded\_ea{\_savedtitle}}}
\def \formathead #1#2{\isempty{#1}\iffalse #1: #2\fi}
\headline = {%
   \ifodd \pageno
       \hfil \ea\formathead\firstmark{}{}%
```

```
    \else
        Chapter: \runningchap \hfil
    \fi
}
```

The \caption/⟨*letter*⟩ uses _⟨*letter*⟩num counter. The group opened by \caption is finalized by first \par from empty line or from \vskip or from \endinsert. The _printcaption⟨*letter*⟩ is called, it starts with printing of the caption.
The \cskip macro inserts nobreakable vertical space between caption and the object.

sections.opm

```
\_def\_caption/#1{\_def\_tmpa{#1}\_nospaceafter \_capA}
\_optdef\_capA []{\_trylabel \_incaption}
\_def\_incaption {\_bgroup
   \_ifcsname _\_tmpa num\_endcsname \_ea\_incr \_csname _\_tmpa num\_endcsname
   \_else \_opwarning{Unknown caption /\_tmpa}\_fi
   \_edef\_thecapnum {\csname _the\_tmpa num\endcsname}%
   \_edef\_thecaptitle{\_mtext{\_tmpa}}%
   \_ifcsname _everycaption\_tmpa\_endcsname
       \_ea\_the \_csname _everycaption\_tmpa\_endcsname \_fi
   \_def\_par{\_nbpar\_egroup}\_let\par=\_par
   \_cs{_printcaption\_tmpa}%
}
\_def \_cskip {\_par\_nobreak\_medskip} % space between caption and the object

\_public \caption \cskip ;
```

The _printcaptiont and _printcaptionf macros start in vertical mode. They switch to horizontal mode and use _wlabel_thecapnum (in order to make reference and hyperlink destination) a they can use:

- _thecaptitle ... expands to the word Table or Figure (depending on the current language).
- _thecapnum ... expands to \the⟨*letter*⟩num (caption number).

sections.opm

```
\_def \_printcaptiont {%
   \_noindent \_wlabel\_thecapnum {\_bf\_thecaptitle~\_thecapnum}\_enspace
   \_narrowlastlinecentered\_iindent
}
\_let \_printcaptionf = \_printcaptiont % caption of figures = caption of tables
```

The default format of \caption text is paragraph in block narrower by _iindent and with the last line is centered. This setting is done by the _narrowlastlinecentered macro.

sections.opm

```
\_def\_narrowlastlinecentered#1{%
   \_leftskip=#1plus1fil
   \_rightskip=#1plus-1fil
   \_parfillskip=0pt plus2fil
}
```

\eqmark is processed in display mode (we add \eqno primitive) or in internal mode when \eqaligno is used (we don't add \eqno).

sections.opm

```
\_optdef\_eqmark []{\_trylabel \_ineqmark}
\_def\_ineqmark{\_incr\_dnum
  \_ifinner\_else\_eqno \_fi
  \_wlabel\_thednum \_thednum
}
\_public \eqmark ;
```

The \numberedpar ⟨*letter*⟩{⟨*name*⟩} is implemented here.

sections.opm

```
\_newcount\_counterA \_newcount\_counterB \_newcount\_counterC
\_newcount\_counterD \_newcount\_counterE

\_def\_resetABCDE {\_counterA=0 \_counterB=0 \_counterC=0 \_counterD=0 \_counterE=0 }

\_def \_theAnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterA}
\_def \_theBnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterB}
\_def \_theCnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterC}
```

```
\_def \_theDnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterD}
\_def \_theEnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterE}

\_def\_numberedpar#1#2{\_ea \_incr \_csname _counter#1\_endcsname
  \_def\_tmpa{#1}\_def\_tmpb{#2}\_numberedparparam}
\_optdef\_numberedparparam[]{%
  \_ea \_printnumberedpar \_csname _the\_tmpa num\_ea\_endcsname\_ea{\_tmpb}}

\_public \numberedpar ;
```

The _printnumberedpar \theXnum {⟨*name*⟩} opens numbered paragraph and prints it. The optional parameter is in _the_opt. You can re-define it if you need another design.

_printnumberedpar needs not to be re-defined if you only want to print Theorems in italic and to insert vertical skips (for example). You can do this by the following code:

```
\def\theorem    {\medskip\bgroup\it \numberedpar A{Theorem}}
\def\endtheorem {\par\egroup\medskip}

\theorem Let $M$ be... \endtheorem
```

sections.opm

```
\_def \_printnumberedpar #1#2{\_par
  \_noindent\_wlabel #1%
  {\bf #2 #1\_istoksempty\_opt\_iffalse \_space \_the\_opt \_fi.}\_space
  \_ignorespaces
}
```

2.26 Lists, items

lists.opm

```
\_codedecl \begitems {Lists: begitems, enditems <2020-04-21>} % preloaded in format
```

_aboveliskip is used above the list of items,
_belowliskip is used below the list of items and
_interliskip is used between items.
_listskipA is used as \listskipamount at level 1 of items.
_listskipB is used as \listskipamount at other levels.
_setlistskip sets the skip dependent on the current level of items

lists.opm

```
\_def\_aboveliskip {\_removelastskip \_penalty-100 \_vskip\_listskipamount}
\_def\_belowliskip {\_penalty-200 \_vskip\_listskipamount}
\_def\_interliskip {}
\_def\_listskipA {\_medskipamount}
\_def\_listskipB {0pt plus.5\smallskipamount}

\_def\_setlistskip {%
  \_ifnum \_ilevel = 1 \_listskipamount = \_listskipA \_relax
  \_else  \_listskipamount = \_listskipB \_relax
  \_fi}
```

The \itemnum is locally reset to zero in each group declared by \begitems. So nested lists are numbered independently. User can set initial value of \itemnum to another value after \beitems if he/she want. Each level of nested lists is indented by new \iindent from left. Default item mark is _printitem.
The \begitems runs _aboveliskip only if we are not near below a title, where a vertical skip is placed already and where the \penalty 11333 is. It activates * and defines it as _startitem.
The \enditems runs _isnextchar_par{}{_noindent} thus the next paragraph is without indentation if there is no empty line between the list and this paragraph (it is similar behavior as after display math).

lists.opm

```
\_newcount\_itemnum  \_itemnum=0
\_newtoks\_printitem

\_def\_begitems{\_par
  \_bgroup
  \_advance \_ilevel by1
  \_setlistskip
  \_ifnum\_lastpenalty<10000 \_aboveliskip \_fi
```

```
  \_itemnum=0 \_adef*{\_startitem}
  \_advance\_leftskip by\_iindent
  \_printitem=\_defaultitem
  \_the\_everylist \_relax
}
\_def\_enditems{\_par\_belowliskip\_egroup \_isnextchar\_par{}{\_noindent}}

\_def\_startitem{\_par \_ifnum\_itemnum>0 \_interliskip \_fi
   \_advance\_itemnum by1
   \_the\_everyitem \_noindent\_llap{\_the\_printitem}\_ignorespaces
}
\_public \begitems \enditems \itemnum ;
```

\novspaces sets \listskipamount to 0pt.

lists.opm

```
\_def\_novspaces {\_removelastskip \_listskipamount=0pt \_relax}
\_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.

lists.opm

```
\_def\_style#1{%
   \_ifcsname _item:#1\_endcsname \_printitem=\ea{\_csname _item:#1\_endcsname}%
   \_else \_printitem=\_defaultitem \_fi
}
\_sdef{_item:o}{\_raise.4ex\_hbox{$\_scriptscriptstyle\_bullet$} }
\_sdef{_item:-}{- }
\_sdef{_item:n}{\_the\_itemnum. }
\_sdef{_item:N}{\_the\_itemnum) }
\_sdef{_item:i}{(\_romannumeral\_itemnum) }
\_sdef{_item:I}{\_uppercase\_ea{\_romannumeral\_itemnum}\_kern.5em}
\_sdef{_item:a}{\_athe\_itemnum) }
\_sdef{_item:A}{\_uppercase\_ea{\_athe\_itemnum}) }
\_sdef{_item:x}{\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
\_sdef{_item:X}{\_raise.2ex\_fullrectangle{1ex}\_kern.5em}
```

_athe{⟨*num*⟩} returns the ⟨*num*⟩s lowercase letter from the alphabet.
_fullrectangle{⟨*dimen*⟩} prints full rectangle with given ⟨*dimen*⟩.

lists.opm

```
\_def\_fullrectangle#1{\_hbox{\_vrule height#1 width#1}}

\_def\_athe#1{\_ifcase#1?\_or a\_or b\_or c\_or d\_or e\_or f\_or g\_or h\_or
   i\_or j\_or k\_or l\_or m\_or n\_or o\_or p\_or q\_or r\_or s\_or t\_or
   u\_or v\_or w\_or x\_or y\_or z\_else ?\_fi
}
\_public \style ;
```

2.27 Verbatim, listings

2.27.1 Inline and “display” verbatim

verbatim.opm

```
\_codedecl \begtt {Verbatim <2020-04-22>} % preloaded in format
```

The internal parameters _ttskip, _ttpenalty, _viline, _vifile and _ttfont for verbatim macros are set.

verbatim.opm

```
\_def\_ttskip{\_medskip}          % space above and below \begtt, \verbinput
\_mathchardef\_ttpenalty=100      % penalty between lines in \begtt, \verbinput
\_newcount\_viline                % last line number in \verbinput
\_newread\_vifile                 % file given by \verinput
\_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 \proteced 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

```
\_def\_code#1{\_unexpanded\_ea{\_csname _code \_endcsname{#1}}}
\_protected\_sdef{_code }#1{{\_escapechar=-1 \_ttfont \_the\_everyintt \_relax
   \_ea\_printinverbatim\_ea{\_detokenize{#1}}}}
\_def\_printinverbatim#1{\_leavevmode\_hbox{#1}}

\_regmacro {}{}{\_let\code=\_detokenize \_let\_code=\_detokenize}
\_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

```
\_newcatcodetable \_verbatimcatcodes
\_def\_setverb{\_begingroup
   \_def\do##1{\_catcode`##1=12 }
   \_dospecials
   \_savecatcodetable\_verbatimcatcodes % all characters are normal
   \_endgroup
}
\_setverb
\_def\_setverb{\_catcodetable\_verbatimcatcodes }%
```

\activettchar⟨*char*⟩ saves original catcode of previously declared ⟨*char*⟩ (if such character was declared) using _savedttchar and _savedttcharc values. Then new such values are stored. The declared charater is activated by _adef as a macro (active character) which opens a group, does _setverb and other settings and reads its parameter until second the same character. This is done by the _readverb macro. Finally it prints scanned ⟨*text*⟩ by _printinverbatim and closes group. Suppose that \activettchar" is used. Then the following work is schematically done:

```
\_def "{\_begingroup \_setverb ... \_readverb}
\_def \_readverb #1"{\_printinverbatim{#1}\_endgroup}
```

Note that the second occurrence of " is not active because _setverb deactivates it.

verbatim.opm

```
\_def\_activettchar#1{%
   \_ifx\_savedttchar\_undefined\_else \_catcode\_savedttchar=\_savedttcharc \_fi
   \_chardef\_savedttchar=`#1
   \_chardef\_savedttcharc=\_catcode`#1
   \_adef{#1}{\_begingroup \_setverb \_adef{ }{\ }\_ttfont \_the\_everyintt\_relax \_readverb}%
   \_def\_readverb ##1#1{\_printinverbatim{##1}\_endgroup}%
}
\_public \activettchar ;
```

\begtt is defined only as public. We don't need private _begtt variant. This macro is defined by \eoldef, so user can put a parameter at the same line where \begtt is. This #1 parameter is used after _everytt parameters settings, so user can change them locally.

The \begtt macro opens group, does _setverb and another preprocessing, sets \endlinechar to ^^J and reads the following text in verbatim mode until \endtt occurs. This scanning is done by _startverb macro which is defined as:

```
\_def\_startverb #1\endtt #2^^J{...}
```

We must to ensure that the backslash in \endtt has category 12 (this is a reason of the \ea chain in real code). The #2 is something between \endtt and end of the same line and it is simply ignored.

The _startverb puts the scanned data to _prepareverbdata. It sets the data to _tmpb without changes by default, but you should re-define it in order to do special changes, if you want. (For example, \hisyntax redefines this macro.) The scanned data have ^^J at each end of line and all spaces are active characters (defined as \␣). Other characters have normal category 11 or 12.

When `\_prepareverbdata` finishes then `\_startverb` runs `\_printverb` loop over each line of the data and does a final work: last skip plus `\noindent` in the next paragraph.

The `\_printverb` macro calls `\_printverbline{`⟨*line*⟩`}` to each scanned line of verbatim text. This macro expect that it strarts in vertical mode and it must do `\par` in order to return the vertical mode. The `\_printverblinenum` is used here: it does nothing when `\_ttline<0` else it prints the line number using `\_llap`.

verbatim.opm

```
\_eoldef \begtt#1{\_par \_wipeepar
   \_vskip\_parskip \_ttskip
   \_begingroup
   \_setverb
   \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
   \_adef{ }{\ }\_adef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
   \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
   \_the\_everytt \_relax #1\_relax \_ttfont
   \_endlinechar=`\^^J
   \_startverb
}
\_ea\_def\_ea\_startverb \_ea#\ea1\_csstring\\endtt#2^^J{%
   \_prepareverbdata\_tmpb{#1^^J}%
   \_ea\_printverb \_tmpb\_end
   \_par
   \_endgroup \_ttskip
   \_isnextchar\_par{}{\_noindent}%
}
\_def\_printverb #1^^J#2{\_ifx\_end#2
      \_bgroup \_adef{ }{}\_def\t{}%
          \_ifcat&#1&\_egroup \_else\_egroup \_printverbline{#1}\_fi
   \_else
      \_printverbline{#1}%
      \_ea \_printverb \_ea #2%
   \_fi
}
\_def\_prepareverbdata#1#2{\_def#1{#2}}
\_def\_printverbline#1{\_penalty \_ttpenalty
   \_indent \_printverblinenum \_kern\_ttshift #1\par}
\_def\_initverblinenum{\_tenrm \_thefontscale[700]\_ea\_let\_ea\_sevenrm\_the\_font}
\_def\_printverblinenum{\_global\_advance\_ttline by1 \_llap{\_sevenrm \_the\_ttline\_kern.9em}}
```

Macro `\verbinput` uses a file read previously or opens the given file. Then it runs the parameter scanning by `\_viscanparameter` and `\_viscanminus`. Finally the `\_doverbinput` is run. At beginning of `\_doverbinput`, we have `\_viline`= number of lines already read using previous `\verbinput`, `\_vinolines`= the number of lines we need to skip and `\_vidolnes`= the number of lines we need to print. Similar preparation is done as in `\begtt` after the group is opened. Then we skip `\_vinolines` lines in a loop a and we read `\_vidolines` lines. The read data is accumulated into `\_tmpb` macro. The next steps are equal to the steps done in `\_startverb` macro: data are processed via `\_prepareverbdata` and printed via `\_printverb` loop.

verbatim.opm

```
\_def\_verbinput #1(#2) #3 {\_par \_def\_tmpa{#3}%
   \_def\_tmpb{#1}%  cmds used in local group
   \_ifx\_vifilename\_tmpa \_else
      \_openin\_vifile={#3}%
      \_global\_viline=0 \_global\_let\_vifilename=\_tmpa
      \_ifeof\_vifile
         \_opwarning{\_noexpand\verbinput - file "#3" is unable to reading}
         \_ea\_ea\_ea\_skiptorelax
      \_fi
   \_fi
   \_viscanparameter #2+\_relax
}
\_def\_skiptorelax#1\_relax{}

\_def \_viscanparameter #1+#2\_relax{%
   \_if$#2$\_viscanminus(#1)\_else \_viscanplus(#1+#2)\_fi
}
\_def\_viscanplus(#1+#2+){%
   \_if$#1$\_tmpnum=\_viline
```

```
  \_else \_ifnum#1<0 \_tmpnum=\_viline \_advance\_tmpnum by-#1
      \_else \_tmpnum=#1
           \_advance\_tmpnum by-1
           \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi % (0+13) = (1+13)
  \_fi \_fi
  \_edef\_vinolines{\_the\_tmpnum}%
  \_if$#2$\_def\_vidolines{0}\_else\_edef\_vidolines{#2}\_fi
  \_doverbinput
}
\_def\_viscanminus(#1-#2){%
  \_if$#1$\_tmpnum=0
     \_else \_tmpnum=#1 \_advance\_tmpnum by-1 \_fi
  \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi  % (0-13) = (1-13)
  \_edef\_vinolines{\_the\_tmpnum}%
  \_if$#2$\_tmpnum=0
     \_else \_tmpnum=#2 \_advance\_tmpnum by-\_vinolines \_fi
  \_edef\_vidolines{\_the\_tmpnum}%
  \_doverbinput
}
\_def\_doverbinput{%
  \_tmpnum=\_vinolines
  \_advance\_tmpnum by-\_viline
  \_ifnum\_tmpnum<0
     \_openin\_vifile=\_vifilename\_space
     \_global\_viline=0
  \_else
     \_edef\_vinolines{\_the\_tmpnum}%
  \_fi
  \_vskip\_parskip \_ttskip \_wipeepar
  \_begingroup
  \_ifnum\_ttline<-1 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
  \_setverb \_adef{ }{\ }\_adef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
  \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
  \_the\_everytt\_relax \_tmpb\_relax \_ttfont
  \_endlinechar=`\^^J \_tmpnum=0
  \_loop \_ifeof\_vifile \_tmpnum=\_vinolines\_space \_fi
       \_ifnum\_tmpnum<\_vinolines\_space
       \_vireadline \_advance\_tmpnum by1 \_repeat      %% skip lines
  \_edef\_ttlinesave{\_ttline=\_the\_ttline}%
  \_ifnum\_ttline=-1 \_ttline=\_viline \_fi
  \_tmpnum=0 \_def\_tmpb{}%
  \_ifnum\_vidolines=0 \_tmpnum=-1 \_fi
  \_ifeof\_vifile \_tmpnum=\_vidolines\_space \_fi
  \_loop \_ifnum\_tmpnum<\_vidolines\_space
           \_vireadline
           \_ifnum\_vidolines=0 \_else\_advance\_tmpnum by1 \_fi
           \_ifeof\_vifile \_tmpnum=\_vidolines\_space \_else \_visaveline \_fi %% save line
           \_repeat
  \_ea\_prepareverbdata \_ea \_tmpb\_ea{\_tmpb^^J}%
  \_ea\_printverb \_tmpb\_end
  \_global\_ttlinesave
  \_par
  \_endgroup
  \_ttskip
  \_isnextchar\_par{}{\_noindent}%
}
\_def\_vireadline{\_read\_vifile to \_tmp \_global\_advance\_viline by1 }
\_def\_visaveline{\_ea\_addto\_ea\_tmpb\_ea{\_tmp}}

\_public \verbinput ;
```

The \visiblesp sets spaces as visible characters ␣. It redefines \␣ primitive, so it is useful for verbatim modes only.

verbatim.opm

```
\_def \_visiblesp{\_ifx\_initunifonts\_relax \_def\ {\_char9251 }%
                 \_else \_def\ {\_char32 }\_fi}

\_public \visiblesp ;
```

2.27.2 Listings with syntax highlighting

The user can write

```
\begtt \hisytnax{C}
...
\endtt
```

and the code is colorized by C syntax. The user can write `\everytt={\hisyntax{C}}` and all verbatim listings are colorized.

The `\hisyntax{`⟨*name*⟩`}` reads the file `hisyntax-`⟨*name*⟩`.opm` where the colorization is declared. The parameter ⟨*name*⟩ is case insensitive and the file name must include it in lowercase letters. For example the file `hisyntax-c.opm` looks like:

hisyntax-c.opm

```
\_codedecl \_hisyntaxc {Syntax highlighting for C sources <2020-04-03>}

\_newtoks \_hisyntaxc  \_newtoks \_hicolorsc

\_global\_hicolorsc={%     colors for C language
   \_hicolor K \Red      % Keywords
   \_hicolor S \Magenta  % Strings
   \_hicolor C \Green    % Comments
   \_hicolor N \Cyan     % Numbers
   \_hicolor P \Blue     % Preprocessor
   \_hicolor O \Blue     % Non-letters
}
\_global\_hisyntaxc={%
   \_the\_hicolorsc
   \_let\c=\_relax \_let\e=\_relax \let\o=\_relax
   \_replfromto {/*}{*/}         {\x C{/*#1*/}}%   /*...*/
   \_replfromto {//}{^^J}        {\z C{//#1}^^J}%  //...
   \_replfromto {\_string#}{^^J} {\z P{\##1}^^J}%  #include ...
   \_replthis   {\_string\"}     {{\_string\"}}%   \" protected inside strings
   \_replfromto {"}{"}           {\x S{"#1"}}%     "..."
   %
   \_edef\_tmpa {()\_string{\_string}+-*/=[]<>,:;\_pcent\_string&\_string^|!?}% non-letters
   \ea \_foreach \_tmpa
      \_do {\replthis{#1}{\n\o#1\n}}
   \_foreach                                                          % keywords
      {auto}{break}{case}{char}{continue}{default}{do}{double}%
      {else}{entry}{enum}{extern}{float}{for}{goto}{if}{int}{long}{register}%
      {return}{short}{sizeof}{static}{struct}{switch}{typedef}{union}%
      {unsigned}{void}{while}
      \_do {\replthis{\n#1\n}{\z K{#1}}}
   \_replthis{.}{\n.\n}                                               % numbers
   \_foreach 0123456789
      \_do {\replfromto{\n#1}{\n}{\c#1##1\e}}
   \_replthis{\e.\c}{.}
   \_replthis{\e.\n}{.\e}
   \_replthis{\n.\c}{\c.}
   \_replthis{e\e\o+\c}{e+}\_replthis{e\e\o-\c}{e-}
   \_replthis{E\e\o+\c}{E+}\_replthis{E\e\o-\c}{E-}
   \def\o#1{\z O{#1}}
   \def\c#1\e{\z N{#1}}
}
```

OpTEX provides `hisyntax-{c,python,tex,html}.opm` files. You can take inspiration from these files and declare more languages.

User can re-declare colors by `\hicolors={...}` This value has precedence before `\_hicolors`⟨*name*⟩ values declared in the `hicolors-`⟨*name*⟩`.opm` file. What exactly to do: copy `\_hicolors`⟨*name*⟩`={...}` from `hicolors-`⟨*name*⟩`.opm` to your document, rename it as `\hicolors={...}` and do you own colors modifications.

Another way to set non-default colors is to declare `\newtoks\hicolors`⟨*name*⟩ (without the `_` prefix) and set the colors palette here. It has precedence before `\_hicolors`⟨*name*⟩ (with the `_` prefix) declared in the `hicolors-`⟨*name*⟩`.opm` file. This is useful when there are more hi-syntax languages used in one document.

Notes for hi-syntax macro writers

The file `hisyntax-`⟨*name*⟩`.opm` is read only once in the TEX group. If there are definitions then they must be declared as global.

The `hisyntax-`⟨*name*⟩`.opm` file must (globally) declare `\_hisyntax`⟨*name*⟩ tokens string where the action over verbatim text is declared typically by `\replfromto` or `\replthis` macros.

The verbatim text is preparared by *pre-processing phase*, then the `\_hisyntax`⟨*name*⟩ is applied and then *post-processing phase* does final corrections. Finally, the verbatim text is printed line by line.

The pre-processing phase does:

- Each space is replaced by `\n␣\n`, so `\n`⟨*word*⟩`\n` should be a pattern to finding whole words (no subwords). The `\n` control sequence is removed in the post-processing phase.
- Each end of line is repersented by `\n^^J\n`.
- The `\_start` control sequence is added before the verbatim text and `\_end` control sequence is appended to the end of the verbatim text. These control sequences are removed in post-processing phase.

There are special macros working only in a group when processing the verbatim text.

- `\n` means noting but it should be used as a boundary of words as mentioned above.
- `\t` means a tabelator. It is prepared as `\n\t\n` because it can be at the boundary of a word.
- `\x` ⟨*letter*⟩`{`⟨*text*⟩`}` can be used as replacing text. Suppose the example

  ```
  \replfromto{/*}{*/}{\x C{/*#1*/}}
  ```

  This replaces all C comments `/*...*/` by `\x C{/*...*/}`. But the C comments may span more lines, i.e. the `^^J` should be inside it.

  The macro `\x` ⟨*letter*⟩`{`⟨*text*⟩`}` is replaced by one or more `\z` ⟨*letter*⟩`{`⟨*text*⟩`}` in post-processing phase where each parameter ⟨*text*⟩ of `\z` keeps inside one line. Inside-line parameters are represented by `\x C{`⟨*text*⟩`}` and they are replaced to `\z C{`⟨*text*⟩`}` without any change. But:

  ```
  \x C{⟨text1⟩^^J⟨text3⟩^^J⟨text3⟩}
  is replaced by
  \z C{⟨text1⟩}^^J\z C{⟨text2⟩}^^J\z C{⟨text3⟩}
  ```

  The `\z` ⟨*letter*⟩`{`⟨*text*⟩`}` is expanded to `\_z:`⟨*letter*⟩`{`⟨*text*⟩`}` and if `\hicolor` ⟨*letter*⟩ ⟨*color*⟩ is declared then `\_z:`⟨*letter*⟩`{`⟨*text*⟩`}` expands to `{`⟨*color*⟩⟨*text*⟩`}`. So, required color is activated at all lines (separatelly) where C comment spans.
- `\y {`⟨*text*⟩`}` is replaced by `\`⟨*text*⟩ in the post processing phase. It should be used for macros without a parameter. You cannot use unprotected macros as replacement text before the post-processing phase, because the post-processing phase is based on expansion whole verbatim text.

hi-syntax.opm

```
3 \_codedecl \hisyntax {Syntax highlithing of verbatim listings <2020-04-04>} % preloaded in format
```

The following macros `\replfromto` and `\replthis` manipulate with the verbatim text which has been read already and stored in the `\_tmpb` macro.

The `\replfromto` `{`⟨*from*⟩`}{`⟨*to*⟩`}{`⟨*what*⟩`}` finds first ⟨*from*⟩ then the first ⟨*to*⟩ following by ⟨*from*⟩ pattern and the ⟨*text*⟩ between them is packed to `#1`. Then ⟨*from*⟩⟨*text*⟩⟨*to*⟩ is replaced by ⟨*what*⟩. The ⟨*what*⟩ parameter can use `#1` which is replaced by the ⟨*text*⟩.

The `\replfromto` continues by finding next ⟨*from*⟩, then, next ⟨*to*⟩ repeatedly over the whole verbatim text. If the verbatim text is ended by opened ⟨*from*⟩ but not closing by ⟨*to*⟩ then ⟨*to*⟩ is appended to the verbatim text automatically and the last part of verbatim text is replaced too.

First two parameters are expanded before usage of `\replfromto`. You can use `\csstring\%` or something else here.

hi-syntax.opm

```
24 \_def\_replfromto #1#2{\_edef\_tmpa{{#1}{#2}}\_ea\_replfromtoE\_tmpa}
25 \_def\_replfromtoE#1#2#3{% #1=from #2=to #3=what to replace
26    \_def\_replfrom##1#1##2{\_addto\_tmpb{##1}%
27       \_ifx\_end##2\_ea\_replstop \_else \_afterfi{\_replto##2}\_fi}%
28    \_def\_replto##1#2##2{%
29       \_ifx\_end##2\_afterfi{\_replfin##1}\_else
30         \_addto\_tmpb{#3}%
31         \_afterfi{\_replfrom##2}\_fi}%
32    \_def\_replfin##1#1\_end{\_addto\_tmpb{#3}\_replstop}%
33    \_edef\_tmpb{\_ea}\_ea\_replfrom\_tmpb#1\_end#2\_end\_end\_relax
34 }
```

```
\_def\_replstop#1\_end\_relax{}
\_def\_finrepl{}
```

The \replthis {⟨*pattern*⟩}{⟨*what*⟩} replaces each ⟨*pattern*⟩ by ⟨*what*⟩. Both parameters of \replthis are expanded first.

hi-syntax.opm

```
\_def\_replthis#1#2{\_edef\_tmpa{{#1}{#2}}\_ea\_replstring\_ea\_tmpb \_tmpa}

\_public \replfromto \replthis ;
```

The patterns ⟨*from*⟩, ⟨*to*⟩ and ⟨*pattern*⟩ are not found when they are hidden in braces {...}. Example:

```
\replfromto{/*}{*/}{\x C{/*#1/*}}
```

replaces all C comments by \x C{...}. The patterns inside {...} are not used by next usage of \replfromto or \replthis macros.

The _xscan macro does replacing \x by \z in the post-processing phase. The \x ⟨*letter*⟩{⟨*text*⟩} expands to _xscan {⟨*letter*⟩}⟨*text*⟩^^J^. If #3 is _end then it signals that something wrong happens, the ⟨*from*⟩ was not terminated by legal ⟨*to*⟩ when \replfromto did work. We must to fix it by the _xscanR macro.

hi-syntax.opm

```
\_def\_xscan#1#2^^J#3{\_ifx\_end#3 \_ea\_xscanR\_fi
   \z{#1}{#2}%
   \_ifx^#3\_else ^^J\_afterfi{\_xscan{#1}#3}\_fi}
\_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

```
\_def\_hicolor #1#2{\_sdef{_z:#1}##1{{#2##1}}}
```

The \hisyntax{⟨*name*⟩} re-defines default _prepareverbdata⟨*macro*⟩⟨*verbtext*⟩ in order to it does more things: It saves ⟨*verbtext*⟩ to _tmpb, appends \n around spaces and ^^J characters in pre-processing phase, it opens hisyntax-⟨*name*⟩.opm file if _hisyntax⟨*name*⟩ is not defined. Then _the_isyntax⟨*name*⟩ is processed. Finally, the post-processing phase is realized by setting appropriate values to \x and \y macros and doing _edef_tmpb{_tmpb}.

hi-syntax.opm

```
\_def\_hisyntax#1{\_def\_prepareverbdata##1##2{%
   \_let\n=\relax \_def\t{\n\noexpand\t\n}\_let\_start=\_relax
   \_adef{ }{\n\ \n}\_edef\_tmpb{\_start^^J##2\_end}%
   \_replthis{^^J}{\n^^J\n}\_replthis{\n\_end}{\_end}%
   \_let\x=\_relax  \_let\y=\_relax \_let\z=\_relax \_let\t=\_relax
   \_endlinechar=`\^^M
   \_lowercase{\_def\_tmpa{#1}}%
   \_ifcsname _hialias:\_tmpa\_endcsname \_edef\_tmpa{\_cs{_hialias:\_tmpa}}\_fi
   \_ifx\_tmpa\_empty \_else
      \_unless \_ifcsname _hisyntax\_tmpa\_endcsname
          \_isfile{hisyntax-\_tmpa.opm}\_iftrue \_opinput {hisyntax-\_tmpa.opm} \_fi\_fi
      \_ifcsname _hisyntax\_tmpa\_endcsname
          \_ifcsname hicolors\_tmpa\endcsname
              \_cs{_hicolors\_tmpa}=\_cs{hicolors\_tmpa}%
          \_else
              \_if^\_the\_hicolors^\_else
                  \_ifcsname _hicolors\_tmpa\_endcsname
                      \_global\_cs{_hicolors\_tmpa}=\_hicolors \_global\_hicolors={}%
          \_fi\_fi\_fi
          \_ea\_the \_csname _hisyntax\_tmpa\_endcsname % \_the\_hisyntax<name>
      \_else\_opwarning{Syntax highlighting "\_tmpa" undeclared (no file hisyntax-\_tmpa.opm)}
   \_fi\_fi
   \_replthis{\_start\n^^J}{}\_replthis{^^J\_end}{^^J}%
   \_def\n{}%
   \_def\x####1####2{\_xscan{####1}####2^^J^}%
   \_def\y####1{\_ea \_noexpand \_csname ####1\_endcsname}%
   \_edef\_tmpb{\_tmpb}%
   \_def\z####1{\_cs{_z:####1}}%
   \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
   \_localcolor
}}
```

```
\_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

```
\_sdef{_hialias:xml}{html}
\_sdef{_hialias:json}{c}
```

2.28 Graphics

graphics.opm

```
\_codedecl \inspic {Graphics <2020-04-12>} % preloaded in format
```

`\inspic` accepts old syntax `\inspic` ⟨*filename*⟩⟨*space*⟩ or new syntax `\inspic{`⟨*filename*⟩`}`. So, we need to define two auxiliary macros `\_inspicA` and `\_inspicB`.

You can include more `\pdfximage` parameters (like `page`⟨*number*⟩) in the `\_picparams` macro.

All `\inspic` macros are surrounded in `\hbox` in order user can write `\moveright\inspic ...` or something similar.

graphics.opm

```
\_def\_inspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inspicB\_inspicA}
\_def\_inspicA #1 {\_inspicB {#1}}
\_def\_inspicB #1{%
   \_pdfximage \_ifdim\_picwidth=0pt  \_else width\_picwidth\_fi
               \_ifdim\_picheight=0pt \_else height\_picheight\_fi
               \_picparams {\_the\_picdir#1}%
   \_pdfrefximage\_pdflastximage\_egroup}

\_def\_picparams{}

\_public \inspic ;
```

Inkscape is able to save a picture to `*.pdf` file and labels for the picture to `*.pdf_tex` file. The second file is in LaTeX format (unfortunately) and it is intended to read immediately it after `*.pdf` in included in order to place labels of this ficture in the same font as document is printed. We need to read this LaTeX file by plain TeX macros when `\inkinspic` is used. These macros are stored in the `\_inkdefs` toknes list and it is used locally in the group. The solution is borrowed from OPmac trick 0032.

graphics.opm

```
\_def\_inkinspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}
\_def\_inkinspicA #1 {\_inkinspicB {#1}}
\_def\_inkinspicB #1{%
  \_ifdim\_picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
  \_the\_inkdefs
  \_opinput {\_the\_picdir #1_tex}% file with labels
  \_egroup}

\_newtoks\_inkdefs  \_inkdefs={%
  \_def\makeatletter#1\makeatother{}%
  \_def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{#2}\_hss}%
  \_def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
  \_def\put(#1,#2)#3{\_nointerlineskip\_vbox to0pt{\_vss\_hbox to0pt{\_kern#1\_picwidth
      \_pdfsave\_hbox to0pt{#3}\_pdfrestore\_hss}\_kern#2\_picwidth}}%
  \_def\begin#1{\_csname _begin#1\_endcsname}%
  \_def\_beginpicture(#1,#2){\_vbox\_bgroup
      \_hbox to\_picwidth{}\_kern#2\_picwidth \_def\end##1{\_egroup}}%
  \_def\_begintabular[#1]#2#3\end#4{%
      \_vtop{\_def\\{\_cr}\_tabiteml{}\_tabitemr{}\_table{#2}{#3}}}%
  \_def\color[#1]#2{\_scancolor #2,}%
  \_def\_scancolor#1,#2,#3,{\_pdfliteral{#1 #2 #3 rg}}%
  \_def\makebox(#1)[#2]#3{\_hbox to0pt{\_csname _mbx:#2\_endcsname{#3}}}%
  \_sdef{_mbx:lb}#1{#1\_hss}\_sdef{_mbx:rb}#1{\_hss#1}\_sdef{_mbx:b}#1{\_hss#1\_hss}%
  \_sdef{_mbx:lt}#1{#1\_hss}\_sdef{_mbx:rt}#1{\_hss#1}\_sdef{_mbx:t}#1{\_hss#1\_hss}%
  \_def\rotatebox#1#2{\_pdfrotate{#1}#2}%
  \_def\lineheight#1{}%
  \_def\setlength#1#2{}%
}
\_public \inkinspic ;
```

`\pdfscale{⟨x-scale⟩}{⟨y-scale⟩}` and `\pdfrotate{⟨degrees⟩}` macros are implemented by `\pdfsetmatrix` primitive. We need to know values of sin, cos function in the `\pdfrotate`. We use Lua code for this.

graphics.opm

```
\_def\_pdfscale#1#2{\_pdfsetmatrix{#1 0 0 #2}}

\_def\_gonfunc#1#2{%
   \_directlua{tex.print(string.format('\_pcent.4f',math.#1(3.14159265*(#2)/180)))}%
}
\_def\_sin{\_gonfunc{sin}}
\_def\_cos{\_gonfunc{cos}}

\_def\_pdfrotate#1{\_pdfsetmatrix{\_cos{#1} \_sin{#1} \_sin{(#1)-180} \_cos{#1}}}

\_public \pdfscale \pdfrotate ;
```

The `\transformbox{⟨transformation⟩}{⟨text⟩}` is copied from OPmac trick 0046.
The `\rotbox{⟨degrees⟩}{⟨text⟩}` is a combination of `\rotsimple` from OPmac trick 0101 and the `\transformbox`. Note, that `\rotbox{-90}` puts the rotated text to the height of the outer box (depth is zero) because code from `\rotsimple` is processed. But `\rotbox{-90.0}` puts the rotated text to the depth of the outer box (height is zero) because `\transformbox` is processed.

graphics.opm

```
\_def\_multiplyMxV #1 #2 #3 #4 {% matrix * (vvalX, vvalY)
   \_tmpdim = #1\_vvalX \_advance\_tmpdim by #3\_vvalY
   \_vvalY  = #4\_vvalY \_advance\_vvalY  by #2\_vvalX
   \_vvalX = \_tmpdim
}
\_def\_multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
   \_vvalX=#1pt \_vvalY=#2pt \_ea\_multiplyMxV \_currmatrix
   \_edef\_tmpb{\_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY}%
   \_vvalX=#3pt \_vvalY=#4pt \_ea\_multiplyMxV \_currmatrix
   \_edef\_currmatrix{\_tmpb\_space
      \_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY\_space}%
}
\_def\_transformbox#1#2{\_hbox{\_setbox0=\_hbox{{#2}}%
   \_dimendef\_vvalX 11 \_dimendef\_vvalY 12 % we use these variables
   \_dimendef\_newHt 13 \_dimendef\_newDp 14 % only in this group
   \_dimendef\_newLt 15 \_dimendef\_newRt 16
   \_preptransform{#1}%
   \_kern-\_newLt \_vrule height\_newHt depth\_newDp width0pt
   \_setbox0=\_hbox{\_box0}\_ht0=0pt \_dp0=0pt
   \_pdfsave#1\_rlap{\_box0}\_pdfrestore \_kern\_newRt}%
}
\_def\_preptransform #1{\_def\_currmatrix{1 0 0 1 }%
   \_def\_pdfsetmatrix##1{\_edef\_tmpb{##1 }\_ea\_multiplyMxM \_tmpb\_unskip}%
   \_let\pdfsetmatrix=\_pdfsetmatrix #1%
   \_setnewHtDp 0pt  \_ht0  \_setnewHtDp 0pt  -\_dp0
   \_setnewHtDp \_wd0 \_ht0  \_setnewHtDp \_wd0 -\_dp0
   \_protected\_def \_pdfsetmatrix {\_pdfextension setmatrix}%
   \_let\pdfsetmatrix=\_pdfsetmatrix
}
\_def\_setnewHtDp #1 #2 {%
   \_vvalX=#1\_relax \_vvalY=#2\_relax \_ea\_multiplyMxV \_currmatrix
   \_ifdim\_vvalX<\_newLt \_newLt=\_vvalX \_fi \_ifdim\_vvalX>\_newRt \_newRt=\_vvalX \_fi
   \_ifdim\_vvalY>\_newHt \_newHt=\_vvalY \_fi \_ifdim-\_vvalY>\_newDp \_newDp=-\_vvalY \_fi
}

\_def\_rotbox#1#2{%
   \_isequal{90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_ht0 \_tmpdim=\_dp0}{\_vfill}{#2}%
   \_else \_isequal{-90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_dp0 \_tmpdim=\_ht0}{}{#2}%
   \_else \_transformbox{\_pdfrotate{#1}}{#2}%
   \_fi \_fi
}
\_def\_rotboxA #1#2#3#4{\_hbox{\_setbox0=\hbox{{#4}}#2%
   \_vbox to\_wd0{#3\_wd0=0pt \_dp0=0pt \_ht0=0pt
                  \_pdfsave\_pdfrotate{#1}\_box0\_pdfrestore\vfil}%
   \_kern\_tmpdim
}}
\_public \transformbox \rotbox ;
```

`\_scantwodimens` scans two objects with the syntactic rule ⟨*dimen*⟩ and returns `{`⟨*number*⟩`}{`⟨*number*⟩`}` in `sp` unit.
`\puttext` ⟨*right*⟩ ⟨*up*⟩`{`⟨*text*⟩`}` puts the ⟨*text*⟩ to desired place: From current point moves ⟨*down*⟩ and ⟨*right*⟩, puts the ⟨*text*⟩ and returns back. The current point is unchanged after this macro ends.
`\putpic` ⟨*right*⟩ ⟨*up*⟩ ⟨*width*⟩ ⟨*height*⟩ `{`⟨*image-file*⟩`}` does `\puttext` with the image scaled to desired ⟨*width*⟩ and ⟨*height*⟩. If ⟨*with*⟩ or ⟨*height*⟩ is zero, natural dimension is used. The `\nospec` is a shortcut to such natural dimension.
`\backgroundpic{`⟨*image-file*⟩`}` puts the image to the background of each page. It it used in the slides style, for example.

graphics.opm

```
\_def\_scantwodimens{%
   \_directlua{tex.print(string.format('{\_pcent d}{\_pcent d}',
               token.scan_dimen(),token.scan_dimen()))}%
}

\_def\_puttext{\_ea\_ea\_ea\_puttextA\_scantwodimens}
\_def\_puttextA#1#2#3{\_setbox0=\_hbox{{#3}}\_dimen1=#1sp \_dimen2=#2sp \_puttextB}
\_def\_puttextB{%
   \_ifvmode
      \_ifdim\_prevdepth>0pt \_vskip-\_prevdepth \_relax \_fi
      \_nointerlineskip
   \_fi
   \_wd0=0pt \_ht0=0pt \_dp0=0pt
   \_vbox to0pt{\_kern-\_dimen2 \_hbox to0pt{\_kern\_dimen1 \_box0\_hss}\_vss}}

\_def\_putpic{\_ea\_ea\_ea\_putpicA\_scantwodimens}
\_def\_putpicA#1#2{\_dimen1=#1sp \_dimen2=#2sp \_ea\_ea\_ea\_putpicB\_scantwodimens}
\_def\_putpicB#1#2#3{\_setbox0=\_hbox{\_picwidth=#1sp \_picheight=#2sp \_inspic{#3}}\_puttextB}

\_newbox\_bgbox
\_def\_backgroundpic#1{%
   \_setbox\_bgbox=\_hbox{\_picwidth=\_pdfpagewidth \_picheight=\_pdfpageheight \_inspic{#1}}%
   \_pgbackground={\_copy\_bgbox}
}
\_def\nospec{0pt}
\_public \puttext \putpic \backgroundpic ;
```

`\_circle{`⟨*x*⟩`}{`⟨*y*⟩`}` creates an ellipse with ⟨*x*⟩ axis and ⟨*y*⟩ axix. The origin is in the center.
`\_oval{`⟨*x*⟩`}{`⟨*y*⟩`}{`⟨*roudness*⟩`}` creates an oval with ⟨*x*⟩, ⟨*y*⟩ size and with given ⟨*roundness*⟩. The real size is bigger by 2⟨*roundness*⟩. The origin is at the left bottom corner.
`\_mv{`⟨*x*⟩`}{`⟨*y*⟩`}{`⟨*curve*⟩`}` moves current point to ⟨*x*⟩, ⟨*y*⟩, crates the ⟨*curve*⟩ and retuns back the current point. All these macros are fully expandable and they can be used in the `\pdfliteral` argument.

graphics.opm

```
\def\_circle#1#2{\_expr{.5*(#1)} 0 m
   \_expr{.5*(#1)} \_expr{.276*(#2)} \_expr{.276*(#1)} \_expr{.5*(#2)} 0 \_expr{.5*(#2)} c
   \_expr{-.276*(#1)} \_expr{.5*(#2)} \_expr{-.5*(#1)} \_expr{.276*(#2)} \_expr{-.5*(#1)} 0 c
   \_expr{-.5*(#1)} \_expr{-.276*(#2)} \_expr{-.276*(#1)} \_expr{-.5*(#2)} 0 \_expr{-.5*(#2)} c
   \_expr{.276*(#1)} \_expr{-.5*(#2)} \_expr{.5*(#1)} \_expr{-.276*(#2)} \_expr{.5*(#1)} 0 c h}

\def\_oval#1#2#3{0 \_expr{-(#3)} m \_expr{#1} \_expr{-(#3)} l
    \_expr{(#1)+.552*(#3)} \_expr{-(#3)} \_expr{(#1)+(#3)} \_expr{-.552*(#3)}
                                                          \_expr{(#1)+(#3)} 0 c
    \_expr{(#1)+(#3)} \_expr{#2} l
    \_expr{(#1)+(#3)} \_expr{(#2)+.552*(#3)} \_expr{(#1)+.552*(#3)} \_expr{(#2)+(#3)}
                                             \_expr{#1} \_expr{(#2)+(#3)} c
    0 \_expr{(#2)+(#3)} l
    \_expr{-.552*(#3)} \_expr{(#2)+(#3)} \_expr{-(#3)} \_expr{(#2)+.552*(#3)}
                                       \_expr{-(#3)} \_expr{#2} c
    \_expr{-(#3)} 0 l
    \_expr{-(#3)} \_expr{-.552*(#3)} \_expr{-.552*(#3)} \_expr{-(#3)}  0 \_expr{-(#3)} c h}

\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 _setflcolor uses the _fillstroke macro to separate filling color and drawing color.

graphics.opm

```
\_newdimen \_lwidth
\_def\_fcolor{\_let\_fcolorvalue}
\_def\_lcolor{\_let\_lcolorvalue}
\_def\_shadow{\_let\_shadowvalue}
\_def\_overlapmargins{\_let\_overlapmarginsvalue}
\_def\_ratio{\isnextchar ={\_ratioA}{\_ratioA=}}
\_def\_ratioA =#1 {\_def\_rationvalue{#1}}
\_def\_touppervalue#1{\_ifx#1n\_let#1=N\_fi}

\_def\_setflcolors#1{% use only in a group
   \_def\_setcolor##1{##1}%
   \_def\_fillstroke##1##2{##1}%
   \_edef#1{\_fcolorvalue}%
   \_def\_fillstroke##1##2{##2}%
   \_edef#1{#1\_space\_lcolorvalue\_space}%
}
\_optdef\_inoval[]{\_vbox\_bgroup
   \_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
   \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt
   \_the\_ovalparams \_relax \_the\_opt \_relax
   \_touppervalue\_overlapmarginsvalue \_touppervalue\_shadowvalue
   \_ifx\_overlapmarginsvalue N%
      \_advance\_hsize by-2\_hhkern \_advance\_hsize by-2\_roundness \_fi
   \_setbox0=\_hbox\_bgroup\_bgroup \_aftergroup\_inovalA \_kern\_hhkern \_let\_next=%
}
\_def\_inovalA{\_isnextchar\_colorstackpop\_inovalB\_inovalC}
\_def\_inovalB#1{#1\_isnextchar\_colorstackpop\_inovalB\_inovalC}
\_def\_inovalC{\_egroup % of \setbox0=\hbox\bgroup
   \_ifdim\_vvkern=0pt \_else \_ht0=\_dimexpr\_ht0+\_vvkern \_relax
                              \_dp0=\_dimexpr\_dp0+\_vvkern \_relax \_fi
   \_ifdim\_hhkern=0pt \_else \_wd0=\_dimexpr\_wd0+\_hhkern \_relax \_fi
   \_ifx\_overlapmarginsvalue N\_dimen0=\_roundness \_dimen1=\_roundness
   \_else                      \_dimen0=-\_hhkern   \_dimen1=-\_vvkern \_fi
   \_setflcolors\_tmp
   \_hbox{\_kern\_dimen0
      \_vbox to0pt{\_kern\_dp0
         \_ifx\_shadowvalue N\_else
            \_edef\_tmpb{{\_bp{\_wd0+\_lwidth}}{\_bp{\_ht0+\_dp0+\_lwidth}}{\_bp{\_roundness}}}%
            \_doshadow\_oval
         \_fi
         \_pdfliteral{q \_bp{\_lwidth} w \_tmp
            \_oval{\_bp{\wd0}}{\_bp{\_ht0+\_dp0}}{\_bp{\_roundness}} B Q}\_vss}%
      \_ht0=\_dimexpr\_ht0+\_dimen1 \_relax \_dp0=\_dimexpr\_dp0+\_dimen1 \_relax
      \_box0
      \_kern\_dimen0}%
   \_egroup % of \vbox\bgroup
}
\_optdef\_incircle[]{\_vbox\_bgroup
   \_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
   \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt
   \_ea\_the \_ea\_circleparams \_space \_relax
   \_ea\_the \_ea\_opt \_space \_relax
   \_touppervalue\_overlapmarginsvalue \_touppervalue\_shadowvalue
   \_setbox0=\_hbox\_bgroup\_bgroup \_aftergroup\_incircleA \_kern\_hhkern \_let\_next=%
}
\_def\_incircleA {\_isnextchar\_colorstackpop\_incircleB\_incircleC}
\_def\_incircleB #1{#1\_isnextchar\_colorstackpop\_incircleB\_incircleC}
\_def\_incircleC {\_egroup % of \setbox0=\hbox\bgroup
   \_wd0=\dimexpr \_wd0+\_hhkern \_relax
   \_ht0=\dimexpr \_ht0+\_vvkern \_relax \_dp0=\dimexpr \_dp0+\_vvkern \_relax
   \_ifdim \_rationvalue\_dimexpr \_ht0+\_dp0 > \_wd0
          \_dimen3=\_dimexpr \_ht0+\_dp0 \_relax  \_dimen2=\_rationvalue\_dimen3
   \_else \_dimen2=\_wd0 \_dimen3=\_expr{1/\_rationvalue}\_dimen2 \fi
   \_setflcolors\_tmp
   \_ifx\_overlapmarginsvalue N\_dimen0=0pt \_dimen1=0pt
   \_else \_dimen0=-\hhkern \_dimen1=-\vvkern \_fi
```

```
   \_hbox{\_kern\dimen0
      \_ifx\_shadowvalue N\_else
         \_edef\_tmpb{{\_bp{\_dimen2+\_lwidth}}{\_bp{\_dimen3+\_lwidth}}{}}%
         \_doshadow\_circlet
      \_fi
      \_pdfliteral{q \_bp{\_lwidth} w \_tmp \_mv{\_bp{.5\_wd0}}{\_bp{(\ht0-\dp0)/2}}
                                     {\_circle{\_bp{\_dimen2}}{\_bp{\_dimen3}} B} Q}%
      \_ifdim\_dimen1=0pt \_else
           \_ht0=\dimexpr \_ht0+\_dimen1 \_relax \_dp0=\dimexpr \_dp0+\_dimen1 \_relax \_fi
      \_box0
      \_kern\_dimen0}
   \_egroup % of \vbox\bgroup
}
\_def\_circlet#1#2#3{\_circle{#1}{#2}}

\_public \inoval \incircle \ratio \lwidth \fcolor \lcolor \shadow \overlapmargins ;
```

A shadow effect is implemented here. The shadow is equal to the silhouette of the given path in gray-transparent color shifted by `\_shadowmoveto` vector and with blurred boundary. A waistline with the width 2*`\_shadowb` around the boundary is blurred. The `\shadowlevels` levels of transparent shapes is used for creating this effect. The `\shadowlevels+1/2` level is equal to the shifted given path.

graphics.opm

```
\_def\_shadowlevels{9}         % number of layers for blurr effect
\_def\_shadowdarknessA{0.025}  % transparency of first shadowlevels/2 layers
\_def\_shadowdarknessB{0.07}   % transparency of second half of layers
\_def\_shadowmoveto{1.8 -2.5}  % vector defines shifting layer (in bp)
\_def\_shadowb{1}              % 2*shadowb = blurring area thickness
```

The `\_pdfpageresources` primitive is used to define transparency. It does not work when used in a box. So, we use it at the begining of the output routine. The modification of the output routine is done using `\_insertshadowresources` only once when the shadow effect is used first.

graphics.opm

```
\_def\_insertshadowresources{%
   \_global\_addto\_begoutput{\_setshadowresources}%
   \_xdef\_setshadowresources{%
      \_pdfpageresources{/ExtGState
      <<
      /op1  << /Type /ExtGState /ca \_shadowdarknessA >>
      /op2  << /Type /ExtGState /ca \_shadowdarknessB >>
      >>
      }%
   }%
   \_global\_let\_insertshadowresources=\_relax
}
```

The `\_doshadow{`⟨*curve*⟩`}` does the shadow effect.

graphics.opm

```
\_def\_doshadow#1{\_vbox{%
    \_insertshadowresources
    \_tmpnum=\_numexpr (\_shadowlevels-1)/2 \_relax
    \_edef\_tmpfin{\_the\_tmpnum}%
    \_ifnum\_tmpfin=0 \_def\_shadowb{0}\_def\_shadowstep{0}%
    \_else \_edef\_shadowstep{\_expr{\_shadowb/\_tmpfin}}\_fi
    \_def\_tmpa##1##2##3{\_def\_tmpb
        {#1{##1+2*\_the\_tmpnum*\_shadowstep}{##2+2*\_the\_tmpnum*\_shadowstep}{##3}}}%
    \_ea \_tmpa \_tmpb
    \_def\_shadowlayer{%
        \_ifnum\_tmpnum=0 /op2 gs \_fi
        \_tmpb\_space f
        \_immediateassignment\_advance\_tmpnum by-1
        \_ifnum-\_tmpfin<\_tmpnum
           \_ifx#1\_oval 1 0 0 1 \_shadowstep\_space \_shadowstep\_space cm \_fi
           \_ea \_shadowlayer \_fi
    }%
    \_pdfliteral{q /op1 gs 0 g 1 0 0 1 \_shadowmoveto\_space cm
       \_ifx#1\_circlet 1 0 0 1 \_expr{\_bp{.5\_wd0}} \_expr{\_bp{(\_ht0-\_dp0)/2}} cm
       \_else  1 0 0 1 -\_shadowb\_space -\_shadowb\_space cm \_fi
       \_shadowlayer Q}
}}
```

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 TEX dimensions in their parameters.

graphics.opm

```
\_def\_clipinpath#1#2#3#4{% #1=x-pos[bp], #2=y-pos[bp], #3=curve, #4=text
   \_hbox{\setbox0=\hbox{{#4}}%
          \_tmpdim=\wd0 \wd0=0pt
          \_pdfliteral{q \_mv{#1}{#2}{#3 W n}}%
          \_box0\_pdfliteral{Q}\_kern\_tmpdim
   }%
}

\_def\_clipinoval {\_ea\_ea\_ea\_clipinovalA\_scantwodimens}
\_def\_clipinovalA #1#2{%
   \_def\_tmp{{#1/65781.76}{#2/65781.76}}%
   \_ea\_ea\_ea\_clipinovalB\_scantwodimens
}
\_def\_clipinovalB{\_ea\_clipinovalC\_tmp}
\_def\_clipinovalC#1#2#3#4{%
   \_ea\_clipinpath{#1-(#3/131563.52)+(\_bp{\_roundness})}{#2-(#4/131563.52)+(\_bp{\_roundness})}%
   {\_oval{#3/65781.76-(\_bp{2\_roundness})}{#4/65781.76-(\_bp{2\_roundness})}{\_bp{\_roundness}}}%
}
\_def\_clipincircle {\_ea\_ea\_ea\_clipincircleA\_scantwodimens}
\_def\_clipincircleA #1#2{%
   \_def\_tmp{{#1/65781.76}{#2/65781.76}}%
   \_ea\_ea\_ea\_clipincircleB\_scantwodimens
}
\_def\_clipincircleB#1#2{%
   \_ea\_clipinpath\_tmp{\_circle{#1/65781.76}{#2/65781.76}}%
}
\_public \clipinoval \clipincircle ;
```

2.29 The \table macro

table.opm

```
\_codedecl \table {Basic macros for OpTeX <2020-04-10>} % preloaded in format
```

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.

table.opm

```
\_let\_tablebox=\_vbox
```

Categories (for example of | character) have to be normal when reading \table parameters.

table.opm

```
\_def\_table{\_tablebox\_bgroup \_catcodetable\_optexcatcodes \_tableA}
\_public \table ;
```

The \tablinespace is implemented by enlarging given \tabstrut by desired dimension (height and depth too) and by setting _lineskip=-2_tablinespace. Normal table rows (where no \hrule is between them) have normal baseline distance.

table.opm

```
\_def\_tableA#1{%
   \_the\_thistable \_global\_thistable={}%
   \_ea\_ifx\_ea^\_the\_tabstrut^\_setbox\_tstrutbox=\_null
   \_else \_setbox\_tstrutbox=\_hbox{\_the\_tabstrut}%
          \_setbox\_tstrutbox=\_hbox{\_vrule width0pt
              height\_dimexpr\_ht\_tstrutbox+\_tablinespace
              depth\_dimexpr\_dp\_tstrutbox+\_tablinespace}%
          \_offinterlineskip
          \_lineskip=-2\_tablinespace
   \_fi
   \_colnum=0 \_def\_tmpa{}\_tabdata={}\_scantabdata#1\_relax
   \_the\_everytable \_tableB
}
\_def\_tableB#1{\_halign\_ea{\_the\_tabdata\_cr#1\_crcr}\_egroup}
\_newbox\_tstrutbox  % strut used in table rows
```

```
\_newtoks\_tabdata   % the \halign declaration line
\_newcount\_colnum   % number of columns
```

The _scantabdata converts \table's ⟨*declaration*⟩ to \halign ⟨*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\_tabitemr\_hfil#\_unsskip\_hfil\_the\_tabitemr\_tabstrutA
   &\_the\_tabitemr#\_unsskip\_hfil\_the\_tabitemr\_vrule\_tabstrutA
ddlinedata: &\_dditem &\_dditem\_vvitem &\_dditem &\_dditem
```

The second result in the _ddlinedata macro is a teplate of one row of the table used by \crli macro.

table.opm

```
\_def\_scantabdata#1{\_let\_next=\_scantabdata
   \_ifx\_relax#1\_let\_next=\_relax
   \_else\_ifx|#1\_addtabvrule
      \_else\_ifx(#1\_def\_next{\_scantabdataE}%
         \_else\_isinlist{123456789}#1\_iftrue \_def\_next{\_scantabdataC#1}%
            \_else \_ea\_ifx\_csname _tabdeclare#1\_endcsname \_relax
                  \_ea\_ifx\_csname _paramtabdeclare#1\_endcsname \_relax
                     \_opwarning{tab-declarator "#1" unknown, ignored}%
                  \_else
                     \_def\_next{\_ea\_scantabdataB\_csname _paramtabdeclare#1\_endcsname}\_fi
               \_else \_def\_next{\_ea\_scantabdataA\_csname _tabdeclare#1\_endcsname}%
   \_fi\_fi\_fi\_fi\_fi \_next
}
\_def\_scantabdataA#1{\_addtabitem \_ea\_addtabdata\_ea{#1\_tabstrutA}\_scantabdata}
\_def\_scantabdataB#1#2{\_addtabitem\_ea\_addtabdata\_ea{#1{#2}\_tabstrutA}\_scantabdata}
\_def\_scantabdataC {\_def\_tmpb{}\_afterassignment\_scantabdataD \_tmpnum=}
\_def\_scantabdataD#1{\_loop \_ifnum\_tmpnum>0 \_advance\_tmpnum by-1 \_addto\_tmpb{#1}\_repeat
   \_ea\_scantabdata\_tmpb}
\_def\_scantabdataE#1){\_addtabdata{#1}\_scantabdata}

\_def\_addtabitem{\_ifnum\_colnum>0 \_addtabdata{&}\_addto\_ddlinedata{&\_dditem}\_fi
    \_advance\_colnum by1 \_let\_tmpa=\_relax}
\_def\_addtabdata#1{\_tabdata\_ea{\_the\_tabdata#1}}
\_def\_addtabvrule{%
    \_ifx\_tmpa\_vrule \_addtabdata{\_kern\_vvkern}%
       \_ifnum\_colnum=0 \_addto\_vvleft{\_vvitem}\_else\_addto\_ddlinedata{\_vvitem}\_fi
    \_else \_ifnum\_colnum=0 \_addto\_vvleft{\_vvitemA}\_else\_addto\_ddlinedata{\_vvitemA}\_fi\_fi
    \_let\_tmpa=\_vrule \_addtabdata{\_vrule}%
}
\_def\_tabstrutA{\_copy\_tstrutbox}
\_def\_vvleft{}
\_def\_ddlinedata{}
```

The default "declaration letters" `c`, `l`, `r` and `p` are declared. by `\def\_tabdeclare`⟨*letter*⟩`{...}` for a non-parametric letter and by `\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. All items are put in group because of `\aftergroup` can be used (from `\localcolors` for example). You can declare more such "declaration letters" if you want.

table.opm

```
\_def\_tabdeclarec{{\_the\_tabiteml\_hfil##\_unsskip\_hfil\_the\_tabitemr}}
\_def\_tabdeclarel{{\_the\_tabiteml\_relax##\_unsskip\_hfil\_the\_tabitemr}}
\_def\_tabdeclarer{{\_the\_tabitemr\_hfil##\_unsskip\_the\_tabitemr}}
\_def\_paramtabdeclarep#1{{\_the\_tabitemr
   \_vtop{\_hsize=#1\_relax \_baselineskip=\_normalbaselineskip
   \_lineskiplimit=0pt \_noindent##\_unsskip\_lower\_dp\_tstrutbox\hbox{}}\_the\_tabitemr}}
```

User puts optional spaces around the table item typically, i. e. he/she writes `& text &` instead `&text&`. The left space is ignored by internal TEX algorithm but the right space must be removed by macros. This is a reason why we reccomend 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

```
\_def\_unsskip{\_ifdim\_lastskip>0pt \_unskip\_fi}
```

The family of `\_cr*` macros `\crl`, `\crll`, `\crli`, `\crlli`, `\crlp` and `\tskip` ⟨*dimen*⟩ is implemented here. The `\_zerotabrule` is used in order to suppress the negative `\lineskip` declared by `\tablinespace`.

table.opm

```
\_def\_crl{\_crcr\_noalign{\_hrule}}
\_def\_crll{\_crcr\_noalign{\_hrule\_kern\_hhkern\_hrule}}
\_def\_zerotabrule {\_noalign{\_hrule height0pt width0pt depth0pt}}

\_def\_crli{\_crcr \_zerotabrule \_omit
   \_gdef\_dditem{\_omit\_tablinefil}\_gdef\_vvitem{\_kern\_vvkern\_vrule}\_gdef\_vvitemA{\_vrule}%
   \_vvleft\_tablinefil\_ddlinedata\_crcr \_zerotabrule}
\_def\_crlli{\_crli\_noalign{\_kern\_hhkern}\_crli}
\_def\_tablinefil{\_leaders\_hrule\_hfil}

\_def\_crlp#1{\_crcr \_zerotabrule \_noalign{\_kern-\_drulewidth}%
   \_omit \_xdef\_crlplist{#1}\_xdef\_crlplist{,\_expandafter}\_expandafter\_crlpA\_crlplist,\_end,%
   \_global\_tmpnum=0 \_gdef\_dditem{\_omit\_crlpD}%
   \_gdef\_vvitem{\_kern\_vvkern\_kern\_drulewidth}\_gdef\_vvitemA{\_kern\_drulewidth}%
   \_vvleft\_crlpD\_ddlinedata \_global\_tmpnum=0 \_crcr \_zerotabrule}
\_def\_crlpA#1,{\_ifx\_end#1\_else \_crlpB#1-\_end,\_expandafter\_crlpA\_fi}
\_def\_crlpB#1#2-#3,{\_ifx\_end#3\_xdef\_crlplist{\_crlplist#1#2,}\_else\_crlpC#1#2-#3,\_fi}
\_def\_crlpC#1-#2-#3,{\_tmpnum=#1\_relax
   \_loop \_xdef\_crlplist{\_crlplist\_the\_tmpnum,}\_ifnum\_tmpnum<#2\_advance\_tmpnum by1 \_repeat}
\_def\_crlpD{\_global\_advance\_tmpnum by1
   \_edef\_tmpa{\_noexpand\_isinlist\_noexpand\_crlplist{,\_the\_tmpnum,}}%
   \_tmpa\_iftrue \_kern-\_drulewidth \_tablinefil \_kern-\_drulewidth\_else\_hfil \_fi}

\_def\_tskip{\_afterassignment\_tskipA \_tmpdim}
\_def\_tskipA{\_gdef\_dditem{}\_gdef\_vvitem{}\_gdef\_vvitemA{}\_gdef\_tabstrutA{}%
    \_vbox to\_tmpdim{}\_ddlinedata \_crcr
    \_zerotabrule \_noalign{\_gdef\_tabstrutA{\_copy\_tstrutbox}}}

\_public \crl \crll \crli \crlli \crlp \tskip ;
```

The `\mspan`[⟨*declaration*⟩]{⟨*text*⟩} macro generates similar `\omit\span\omit\span` sequence as plain TEX macro `\multispan`. Moreover, it uses `\_scantabdata` to convert ⟨*declaration*⟩ from `\table` syntax to `\halign` syntax.

table.opm

```
\_def\_mspan{\_omit \_tabdata={\_tabstrutA}\_let\_tmpa=\_relax \_afterassignment\_mspanA \_mscount=}
\_def\_mspanA[#1]#2{\_loop \_ifnum\_mscount>1 \_cs{_span}\_omit \_advance\_mscount-1 \_repeat
   \_colnum=0 \_def\_tmpa{}\_tabdata={}\_scantabdata#1\_relax
   \_setbox0=\_vbox{\_halign\_expandafter{\_the\_tabdata\_cr#2\_cr}\_global\_setbox8=\_lastbox}%
   \_setbox0=\_hbox{\_unhbox8 \_unskip \_global\_setbox8=\_lastbox}%
   \_unhbox8 \_ignorespaces}
\_public \mspan ;
```

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 overwriten by another following `height` parameter. This principle is used for settings another default rule thickness than 0.4 pt by the macro `\rulewidth`.

table.opm

```
\_newdimen\_drulewidth  \_drulewidth=0.4pt
\_let\_orihrule=\_hrule  \_let\_orivrule=\_vrule
\_def\_rulewidth{\_afterassignment\_rulewidthA \_drulewidth}
\_def\_rulewidthA{\_edef\_hrule{\_orihrule height\_drulewidth}%
                  \_edef\_vrule{\_orivrule width\_drulewidth}%
                  \_let\_rulewidth=\_drulewidth
                  \_public \vrule \hrule \rulewidth;}
\_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 `\vvkern` and `\hhkern` parameters.

table.opm

```
\_long\_def\_frame#1{%
   \_hbox{\_vrule\_vtop{\_vbox{\_hrule\_kern\_vvkern
      \_hbox{\_kern\_hhkern\_relax#1\_kern\_hhkern}%
```

```
206    }\_kern\_vvkern\_hrule}\_vrule}}
207 \_public \frame ;
```

2.30 Balanced multi-columns

multicolumns.opm

```
3 \_codedecl \begmulti {Balanced columns <2020-03-26>} % preloaded in format
```

This code is documented in detail in the "TEXbook naruby", pages 244–246, free available, http://petr.olsak.net/tbn.html, but in Czech. Roughly speaking, macros complete all material between `\begmulti`⟨*num-columns*⟩ and `\endmulti` into one `\vbox 6`. Then the macro measures the amount of free space at the current page using `\pagegoal` and `\pagtotal` and does `\vsplit` of `\vbox 6` to columns with height of such free space. This is done only if we have enought amount of material in `\vbox 6` to fill full page by columns. This is repeated in loop until we have less amount of material in `\vbox 6`. Then we run `\_balancecolumns` which balances the last part of columns. Each part of printed material is distributed to main vertical list as `\hbox{`⟨*columns*⟩`}` and we need not do any change in the output routine.

If you have paragraphs in `\begmulti`... `\endmulti` environment then you may say `\raggedright` inside this environment and you can re-assign `\widowpenalty` and `\clubppenalty` (they are set to 10000 in OpTEX).

multicolumns.opm

```
24 \_def\_multiskip{\_medskip}      % space above and below \begmulti...\endmulti
25
26 \_newcount\_mullines
27
28 \_def\_begmulti #1 {\_par\_bgroup\_wipeepar\_multiskip\_penalty0 \_def\_Ncols{#1}
29    \_setbox6=\_vbox\_bgroup\_penalty0
30    %% \hsize := column width = (\hsize+\colsep) / n - \colsep
31    \_advance\_hsize by\_colsep
32    \_divide\_hsize by\_Ncols  \_advance\_hsize by-\_colsep
33    \_mullines=0
34    \_def\par{\_ifhmode\_endgraf\_global\_advance\_mullines by\_prevgraf\_fi}%
35 }
36 \_def\_endmulti{\_vskip-\_prevdepth\_vfil
37    \_ea\_egroup\_ea\_baselineskip\_the\_baselineskip\_relax
38    \_dimen0=.8\_maxdimen \_tmpnum=\_dimen0 \_divide\_tmpnum by\_baselineskip
39    \_splittopskip=\_baselineskip
40    \_setbox1=\_vsplit6 to0pt
41    %% \dimen1 := the free space on the page
42    \_ifdim\_pagegoal=\_maxdimen \_dimen1=\_vsize \_corrsize{\_dimen1}
43    \_else \_dimen1=\_pagegoal \_advance\_dimen1 by-\_pagetotal \_fi
44    \_ifdim \_dimen1<2\_baselineskip
45      \_vfil\_break \_dimen1=\_vsize \_corrsize{\_dimen1} \_fi
46    \_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
47    \_divide\_dimen0 by\_Ncols \_relax
48    %% split the material to more pages?
49    \_ifdim \_dimen0>\_dimen1 \_splitpart
50    \_else \_balancecolumns \_fi  % only balancing
51    \_multiskip\_egroup
52 }
```

Splitting columns...

multicolumns.opm

```
58 \_def\_makecolumns{\_bgroup % full page, destination height: \dimen1
59    \_vbadness=20000 \_setbox1=\_hbox{}\_tmpnum=0
60    \_loop \_ifnum\_Ncols>\_tmpnum
61       \_advance\_tmpnum by1
62       \_setbox1=\_hbox{\_unhbox1 \_vsplit6 to\_dimen1 \_hss}
63    \_repeat
64    \_hbox{}\_nobreak\_vskip-\_splittopskip \_nointerlineskip
65    \_line{\_unhbox1\_unskip}
66    \_dimen0=\_dimen1 \_divide\_dimen0 by\_baselineskip \_multiply\_dimen0 by\_Ncols
67    \_global\_advance\_mullines by-\_dimen0
68    \_egroup
69 }
70 \_def\_splitpart{%
```

```
   \_makecolumns % full page
   \_vskip 0pt plus 1fil minus\_baselineskip \_break
   \_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
   \_divide\_dimen0 by\_Ncols \_relax
   \_ifx\_balancecolumns\_flushcolumns \_advance\_dimen0 by-.5\_vsize \_fi
   \_dimen1=\_vsize \_corrsize{\_dimen1}\_dimen2=\_dimen1
   \_advance\_dimen2 by-\_baselineskip
   %% split the material to more pages?
   \_ifvoid6 \_else
      \_ifdim \_dimen0>\_dimen2 \_ea\_ea\_ea \_splitpart
      \_else \_balancecolumns % last balancing
   \_fi \_fi
}
```

Final balancing of the columns.

multicolumns.opm

```
\_def\_balancecolumns{\_bgroup \_setbox7=\_copy6 % destination height: \dimen0
   \_ifdim\_dimen0>\_baselineskip \_else \_dimen0=\_baselineskip \_fi
   \_vbadness=20000
   \_def\_tmp{%
      \_setbox1=\_hbox{}\_tmpnum=0
      \_loop \_ifnum\_Ncols>\_tmpnum
         \_advance\_tmpnum by1
         \_setbox1=\_hbox{\_unhbox1
              \_ifvoid6 \_hbox to\_wd6{\_hss}\_else \_vsplit6 to\_dimen0 \_fi\_hss}
      \_repeat
   \_ifvoid6 \_else
      \_advance \_dimen0 by.2\_baselineskip
      \_setbox6=\_copy7
      \_ea \_tmp \_fi}\_tmp
   \_hbox{}\_nobreak\_vskip-\_splittopskip \_nointerlineskip
   \_hbox to\_hsize{\_unhbox1\_unskip}%
   \_egroup
}
\_def\_corrsize #1{%% #1 := #1 + \splittopskip - \topskip
   \_advance #1 by \_splittopskip \_advance #1 by-\_topskip
}
\_public \begmulti \endmulti ;
```

2.31 Citations, bibliography

2.31.1 Macros for citations and bibliography preloaded in the format

cite-bib.opm

```
\_codedecl \cite {Cite, Bibliography <2020-03-09>} % loaded in format
```

Registers used by `\cite`, `\bib` macros are declared here. The `\bibnum` counts the bibliography items from one. The `\bibmark` is used when `\nonumcitations` is set.

cite-bib.opm

```
\_newcount\_bibnum                    % the bibitem counter
\_newtoks\_bibmark                    % the bibmark used if \nonumcitations
\_newcount\_lastcitenum \_lastcitenum=0  % for \shortcitations
\_public \bibnum \bibmark ;
```

`\cite` `[`⟨*label*⟩`,`⟨*label*⟩`,...,`⟨*label*⟩`]` manages ⟨*labes*⟩ using `\_citeA` and prints `[`⟨*bib-marks*⟩`]` using `\_printsavedcites`.
`\nocite` `[`⟨*label*⟩`,`⟨*label*⟩`,...,`⟨*label*⟩`]` only manages ⟨*labels*⟩ but prints nothing.
`\rcite` `[`⟨*label*⟩`,`⟨*label*⟩`,...,`⟨*label*⟩`]` behaves like `\cite` but prints ⟨*bib-marks*⟩ without brackets.
`\ecite` `[`⟨*label*⟩`]{`⟨*text*⟩`}` behaves like `\rcite [`⟨*label*⟩`]` but prints ⟨*text*⟩ instead ⟨*bib-mark*⟩. The ⟨*text*⟩ is hyperlinked like ⟨*bib-marks*⟩ when `\cite` or `\rcite` is used. The emptpty internal macro `\_savedcites` will include the ⟨*bib-marks*⟩ list to be printed. This list is set by `\_citeA` inside group and it is used by `\_printsavedcites` in the same group. Each `\cite`/`\rcite`/`\ecite` macro starts from empty list of ⟨*bib-marks*⟩ because new group is opened.

cite-bib.opm

```
\_def\_cite[#1]{{\_citeA#1,,,[\_printsavedcites]}}
\_def\_nocite[#1]{{\_citeA#1,,,}}
\_def\_rcite[#1]{{\_citeA#1,,,\_printsavedcites}}
```

```
\_def\_ecite[#1]{\_bgroup\_citeA#1,,,\_ea\_eciteB\_savedcites;}
\_def\_eciteB#1,#2;#3{\_if?#1\_relax #3\_else \_ilink[cite:#1]{#3}\_fi\_egroup}
\_def\_savedcites{}

\_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{⟨*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:⟨*label*⟩ and _bibm:⟨*number*⟩. First one includes number and second one includes ⟨*nonumber*⟩. The _lastbibnum macro includes last number of bib-entry used in the document. A designer can use it to set appropriate indentation when printing the list of all bib-entries.

cite-bib.opm

```
\_def\_Xbib#1#2#3{\_sdef{_bib:#1}{\_bibnn{#2}&}%
   \_if^#3^\_else\_sdef{_bim:#2}{#3}\_fi\_def\_lastbibnum{#2}}
```

_citeA ⟨*label*⟩, processes one label from list of labels given in the parameter of \cite, \nocite, \rcite or \ecite macros. It adds the ⟨*label*⟩ to global list _citelist which will be used by \usebib (it must to know what ⟨*labels*⟩ are used in the document in order to pick-up only relevant bib-entries from the database. Because we want to save space and not to save the same ⟨*label*⟩ to _citelist twice, we distinguish four cases:

- ⟨*label*⟩ was not declared by _Xbib and it is first such ⟨*label*⟩ in the document: Then _bib:⟨*label*⟩ is undefined and we save label using _addcitlist, write warning on the terminal and define _bib:⟨*label*⟩ as empty.
- ⟨*label*⟩ was not declared by _Xbib but it was used previously in the document: Then _bib:⟨*label*⟩ is empty and we do nothing (only data to _savedcites are saved).
- ⟨*label*⟩ was declared by _Xbib and it is first such ⟨*label*⟩ in the document: Then _bin:⟨*label*⟩ includes _bibnn{⟨*number*⟩}& and we test this case by \if &_bibnn{⟨*number*⟩}&. This is true when _bibnn{⟨*number*⟩} expands to empty. The ⟨*label*⟩ is saved by _addcitelist and _bib:⟨*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 whole list of ⟨*labels*⟩.

cite-bib.opm

```
\_def\_citeA #1#2,{\_if#1,\_else
   \_if *#1\_addcitelist{*}\_ea\_skiptorelax \_fi
   \_ifcsname _bib:#1#2\_endcsname \_else
      \_addcitelist{#1#2}%
      \_opwarning{The cite [#1#2] unknown. Try to TeX me again}\_openref
      \_addto\_savedcites{?,}\_def\_sortcitesA{}\_lastcitenum=0
      \_ea\_gdef \_csname _bib:#1#2\_endcsname {}%
      \_ea\_skiptorelax \_fi
   \_ea\_ifx \_csname _bib:#1#2\_endcsname \_empty
      \_addto\_savedcites{?,}\_def\_sortcitesA{}\_lastcitenum=0
      \_ea\_skiptorelax \_fi
   \_def\_bibnn##1{}%
   \_if &\_csname _bib:#1#2\_endcsname
      \_def\_bibnn##1##2{##1}%
      \_addcitelist{#1#2}%
      \_sxdef{_bib:#1#2}{\_csname _bib:#1#2\_endcsname}%
   \_fi
   \_edef\_savedcites{\_savedcites \_csname _bib:#1#2\_endcsname,}%
   \_relax
   \_ea\_citeA\_fi
}
\_def\_addcitelist#1{\_global\_addto\_citelist{\_citeI[#1]}}
\_def\_citelist{}
```

The ⟨*bib-marks*⟩ (in numeric or text form) are saved in _savedcites macro separated by commas. The _printsavedcites prints them by normal order or sorted if \sortcitations is specified or condensed if \shordcitations is specified.
The \sortcitations appends the dummy number 300000 and we suppose that normal numbers

of bib-entries are less than this constant. This constant is removed after sorting algorithm. The \shortcitations sets simply _lastcitenum=1. The macros for ⟨*bib-marks*⟩ printing follows (sorry, without detail documentation). They are documented in opmac-d.pdf (but only in Czech).

cite-bib.opm

```
\_def\_printsavedcites{\_sortcitesA
   \_chardef\_tmpb=0 \_ea\_citeB\_savedcites,%
   \_ifnum\_tmpb>0 \_printdashcite{\_the\_tmpb}\_fi
}
\_def\_sortcitesA{}
\_def\_sortcitations{%
  \_def\_sortcitesA{\_edef\_savedcites{300000,\_ea}\_ea\_sortcitesB\_savedcites,%
                    \_def\_tmpa####1300000,{\_def\_savedcites{####1}}\_ea\_tmpa\_savedcites}%
}
\_def\_sortcitesB #1,{\_if $#1$%
  \_else
     \_mathchardef\_tmpa=#1
     \_edef\_savedcites{\_ea}\_ea\_sortcitesC \_savedcites\_end
     \_ea\_sortcitesB
  \_fi
}
\_def\_sortcitesC#1,{\_ifnum\_tmpa<#1\_edef\_tmpa{\_the\_tmpa,#1}\_ea\_sortcitesD
                   \_else\_edef\_savedcites{\_savedcites#1,}\_ea\_sortcitesC\_fi}
\_def\_sortcitesD#1\_end{\_edef\_savedcites{\_savedcites\_tmpa,#1}}

\_def\_citeB#1,{\_if$#1$\_else
   \_if?#1\_relax??%
      \_else
      \_ifnum\_lastcitenum=0   % only comma separated list
         \_printcite{#1}%
      \_else
         \_ifx\_citesep\_empty  % first cite item
            \_lastcitenum=#1\_relax
            \_printcite{#1}%
         \_else               % next cite item
            \_advance\_lastcitenum by1
            \_ifnum\_lastcitenum=#1\_relax % cosecutive cite item
               \_mathchardef\_tmpb=\_lastcitenum
            \_else  % there is a gap between cite items
               \_lastcitenum=#1\_relax
               \_ifnum\_tmpb=0 % previous items were printed
                  \_printcite{#1}%
               \_else
                  \_printdashcite{\_the\_tmpb}\_printcite{#1}\_chardef\_tmpb=0
   \_fi\_fi\_fi\_fi\_fi
   \_ea\_citeB\_fi
}
\_def\_shortcitations{\_lastcitenum=1 }

\_def\_printcite#1{\_citesep\_ilink[cite:#1]{\_citelinkA{#1}}\_def\_citesep{,\_hskip.2em\_relax}}
\_def\_printdashcite#1{\_ifmmode-\_else\_hbox{--}\_fi\_ilink[cite:#1]{\_citelinkA{#1}}}
\_def\_citesep{}

\_def\_nonumcitations{\_lastcitenum=0\_def\_sortcitesA{}\_def\_etalchar##1{$^{##1}$}%
   \_def\_citelinkA##1{\_isdefined{_bim:##1}\_iftrue \_csname _bim:##1\_endcsname
      \_else ##1\_opwarning{\_noexpand\_nonumcitations + empty bibmark. Maybe bad BibTeX style}\_fi}
}
\_def\_citelinkA{}

\_public \nonumcitations \sortcitations \shortcitations ;
```

The \bib [⟨*label*⟩] {⟨*optional bib-mark*⟩} prints one bib-entry without reading any database. The bib-enty follows after this command. This command counts the used \bibs from one by \bibnum counter and saves _Xbib{⟨*label*⟩}{_the_bibnum}{_the_bibmark} into .ref file immediately using _wbib. This is the core of creation of mapping from ⟨*labels*⟩ to ⟨*bib-marks*⟩.

cite-bib.opm

```
\_def\_bib[#1]{\_def\_tmp{\_isnextchar={\_bibA[#1]}{\_bibmark={}\_bibB[#1]}}%
   \_ea\_tmp\_romannumeral-`\.} % ignore optional space
\_def\_bibA[#1]=#2{\_bibmark={#2}\_bibB[#1]}
```

```
\_def\_bibB[#1]{\_par \_bibskip
   \_advance\_bibnum by1
   \_noindent \_def\_tmpb{#1}\_wbib{#1}{\_the\_bibnum}{\_the\_bibmark}%
   \_printlabel{#1}%
   \_printbib \_ignorespaces
}
\_def\_wbib#1#2#3{\_dest[cite:\_the\_bibnum]%
   \_ifx\_wref\_wrefrelax\_else \_immediate\_wref\_Xbib{{#1}{#2}{#3}}\_fi}

\_public \bib ;
```

The _printbib prints the bib-entry itself. You can re-define it if you want different design. The _pritbib 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

```
\_def \_printbib {\_hangindent=\_iindent
   \_ifx\_citelinkA\_empty \_hskip\_iindent \_llap{[\_the\_bibnum] }\_fi
}
\_def \_bibskip {\_ifnum\_bibnum>0 \_smallskip \_fi}
```

The \usebib command is implemted in usebib.opm file which is loaded when the \usebib command is firstly used. The usebib.opm file loads the librarian.tex for scaning the .bib files. See the section 2.31.2, where the file usebib.opm is documented.

cite-bib.opm

```
\_def\_usebib{\_par \_opinput {usebib.opm} \_usebib}
\_def\usebib{\_usebib}
```

The macros above works if all \cite (or similar) commands are used before the \usebib command is used because \usebib prints only such bib-entries their ⟨*labels*⟩ are saved in the _citelist. But if some \cite is used after \usebib, then \usebib sets _addcitelist to _writeXcite, so such \cite saves the information to the .reffile in the format _Xcite{⟨*label*⟩}. Such information are copied to _citelistB during reading .ref file and \usebib concats two lists of ⟨*labels*⟩ from _citelist and _citelistB and uses this concatenated list.

cite-bib.opm

```
\_def\_Xcite#1{\_addto\_citelistB{\_citeI[#1]}}
\_def\_writeXcite#1{\_openref\_immediate\_wref\_Xcite{{#1}}}
\_def\_citelistB{}
```

2.31.2 The \usebib command

The file usebib.opm implements the command \usebib/⟨*sorttype*⟩ (⟨*style*⟩) ⟨*bibfiles*⟩ where ⟨*sorttype*⟩ is one letter c (references ordered by citation order in the text) or s (references ordered by key in the style file), ⟨*style*⟩ is the part of the name bib-⟨*style*⟩.opm of the style file and ⟨*bibfiles*⟩ are one or more .bib file names without suffix separated by comma without space. Example:

```
\usebib/s (simple) mybase,yourbase
```

This command reads the ⟨*bibfiles*⟩ directly and creates the list of bibliographics references (only those declared by by \cite[] or \nocite[] in the text). The formatting of such references is defined in the style file. The usage is mentioned in user documentation too.

The principle "first entry wins" is used. Suppose \usebib/s (simple) local,global. If an entry with the same label is declared in local.bib and in global.bib too then the first wins. So, you can set an exceptions in your local.bib file for your document.

Notes for style writers

The bib-⟨*style*⟩.opm file must define the commands:

- _authorname ... formatting of one name in the authors list. The macro can use the following data: _NameCount (the number of the currently processed author name in the list), 0_namecount (the total number of the authors in the list), _Lastname, _Firstname, _Von, _Junior (the parts of the name). See the documentation of the librarian package for more info.

- `\_editorname` ... the same as `\_authorname`, but for editors list.
- `_print:`⟨*entrytype*⟩ (defined by `\_sdef`) for formatting the entry of ⟨*entrytype*⟩. The ⟨*entrytype*⟩ have to be lowercase. This command can use the command:
- `\_bprinta [`⟨*fieldname*⟩`] {`⟨*if defined*⟩`} {`⟨*if not defined*⟩`}`. The part ⟨*if defined*⟩ is executed if ⟨*fieldname*⟩ is declared in .bib file for the entry which is currently processed. Else the part ⟨*if not defined*⟩ is processed. The part ⟨*if defined*⟩ can include the `*` parameter which is replaced by the value of the ⟨*fieldname*⟩. The part ⟨*if not defined*⟩ can include the `\_bibwarning` command if the ⟨*fieldname*⟩ is mandatory.
- `\_bprintb [`⟨*fieldname*⟩`] {`⟨*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 `####1` parameter for internal `\_bprintb`. But if `\_bprinta` commands are nested then the parameter is not duplicated.
- `\_pbprintc \macro {`⟨*if non-empty*⟩`}`. The ⟨*if non-empty*⟩ part is executed if `\macro` is non-empty. The
- parameter can be used, it is replaced by the `\macro`.
- `\_bprintv [`⟨*field1*⟩`,`⟨*field2*⟩`,...] {`⟨*if defined*⟩`} {`⟨*if not defined*⟩`}`. The part ⟨*if defined*⟩ is executed if ⟨*field1*⟩ or ⟨*filed2*⟩ or ... is defined, else the second part ⟨*if not defined*⟩ is executed. There is one filed name or the list field names separated by commas. The parts cannot include any parameter.

There are two special fieldnames: `!author` and `!editor`. The processed list of authors or editors (by repeatedly calling `\_authorname` or `\_editorname`) are used here insted of raw data.

You can define `_print:BEGIN` and/or `_print:END` which is executed at the begin or end of each ⟨*entrytype*⟩. The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by `\_printbib` macro used in OpTEX (and may be redefined by the author or document designer).

You can declare `\_bimark={something}` in the `_print:END` macro. This bibmark is saved to the `.ref` file (created by OpTEX) and used in the next TEX run as `\cite` marks when `\nonumcitations` is set.

The whole style file is read in the group during `\usebib` command is executed before typesetting the reference list. Each definition or setting is local here.

If you are using non-standard fieldnames in .bib database and bib. style, you has to declare them by `\_CreateField {`⟨*fieldname*⟩`}`.

You can declare `\_SortingOrder` in the manner documented by librarian package.

If your style adds some words or abbreviations you can make them multilingual by saying `\_mtext{`⟨*label*⟩`}` instead such word and `\_mtdef{`⟨*label*⟩`} {`⟨*English*⟩`} {`⟨*Czech*⟩`} {`⟨*Slovak*⟩`}` declaration. The right part is printed by current value of the `\language` regiter. You can add more languages by re-defining the `\_mtdef` command. See the section 2.36.3 for more information.

If you are using `\nonumcitations`, then the `\_bibmark` tokens register have to be prepared in the style file (in `_print:BEGIN`, `_print:END`, in `\_authorname` etc.) This value will be used in the `\cite[]` places in the document.

The example of the style file is in `bib-simple.opm`.

User or author of the `bib.` style can create the hidden field which has a precedence while sorting names. Example:

```
\CreateField {sortedby}
\SpecialSort {sortedby}
```

Suppose that the .bib file includes:

```
...
author   = "Jan Chadima",
sortedby = "Hzzadima Jan",
...
```

Now, this author is sorted between H and I, because the Ch digraph in this name has to be sorted by this rule.

If you need (for example) to place the autocitations before other citations, then you can mark your entries in `.bib` file by `sortedby = "@"`, because this character is sorted before `A`.

2.31.3 The `usebib.opm` macro file loaded when `\usebib` is used

usebib.opm

```
\_codedecl \MakeReference {Reading bib databases <2020-03-13>} % loaded on demand by \usebib
```

Loading the `librarian.tex` macro package. See `texdoc librarian` for more information about it.

usebib.opm

```
\_def\_tmp{}
\_let\_errmessageori=\_errmessage % we needn't \errmessage during \input librarian
\_def\_errmessage#1{\_def\_tmp{error}}
\_let\_newwriteori=\_newwrite % we need not to create \jobname.lbr:
\_def\_newwrite#1{\_csname lb@restoreat\_endcsname \_endinput}
\_def\_tmpb{\_catcode`\_=12 \_input librarian \_catcode`\_=11 }\_tmpb
\_let\_errmessage=\_errmessageori
\_let\_newwrite=\_newwriteori
\_ifx\_tmp\_empty\_else
  \_def\_usebib/#1 (#2) #3 {%
     \_opwarning{eTeX and (pdfTeX or XeTeX or LuaTeX) not detected}%
     \_immediate\_write16{\_space\_space
               But librarian package needs it. \_noexpand\usebib ignored.}%
  }
  \_endinput \_fi
\_private \BibFile \ReadList \SortList \SortingOrder \NameCount \AbbreviateFirstname
   \CreateField \RetrieveFieldInFor \RetrieveFieldIn ;
```

The `\usebib` command.

usebib.opm

```
\_def\_usebib/#1 (#2) #3 {%
  \_ifx\_citelist\_empty
     \_opwarning{No cited items. \_noexpand\usebib ignored}%
  \_else
     \_bgroup \_par
        \_emergencystretch=.3\_hsize
        \_ifx\_bibpart\_undefined \_def\_bibpart{none}\_fi
        \_def\_optexbibstyle{#2}%
        \_setctable\_optexcatcodes
        \_input bib-#2.opm
        \_the \_bibtexhook
        \_let\_citeI=\_relax \_xdef\_citelist{\_citelist\_citelistB}%
        \_global\_let\_addcitelist=\_writeXcite
        \_def\_tmp##1[*]##2\_relax{\_def\_tmp{##2}}\_expandafter\_tmp\_citelist[*]\_relax
        \_ifx\_tmp\_empty\_else % there was \nocite[*] used.
           \_setbox0=\_vbox{\_hsize=\_maxdimen \_def\_citelist{}\_adef@{\_readbibentry}%
           \_input #3.bib
           \_expandafter}\_expandafter\_def\_expandafter\_citelist\_expandafter{\_citelist}%
        \_fi
        \_def\_citeI[##1]{\_csname lb@cite\_endcsname{##1}{\_bibpart}{}{}}\_citelist
        \_BibFile{#3}%
        \_if s#1\_SortList{\_bibpart}\_fi
        \_ReadList{\_bibpart}%
        \_restorectable
     \_egroup
  \_fi
}
\_def\_readbibentry#1#{\_readbibentryA}
\_def\_readbibentryA#1{\_readbibentryB#1,,\_relax!.}
\_def\_readbibentryB#1#2,#3\_relax!.{\_addto\_citelist{\_citeI[#1#2]}}
```

Corrections in librarian macros.

usebib.opm

```
\_tmpnum=\_catcode`\@  \_catcode`\@=11
\_def\lb@checkmissingentries#1,{% we needn't \errmessage here, only \opmacwarning
  \_def\lb@temp{#1}%
  \_unless\_ifx\lb@temp\lb@eoe
    \lb@ifcs{#1}{fields}%
            {}%
            {\_opwarning{\_string\_usebib: entry [#1] isn't found in .bib file(s)}}%
    \_ea\lb@checkmissingentries
  \_fi
}
\_def\lb@readentry#1#2#3,{% space before key have to be ingnored
```

```
  \_def\lb@temp{#2#3}%      we need case sensitive keys
  \_def\lb@next{\_ea\lb@gotoat\lb@gobbletoeoe}%
  \lb@ifcs\lb@temp{requested}%
          {\_let\lb@entrykey\lb@temp
           \lb@ifcs\lb@entrykey{fields}{}%
                {\lb@defcs\lb@entrykey{fields}{}%
                 \_lowercase{\lb@addfield{entrytype}{#1}}%
                 \_let\lb@next\lb@analyzeentry}}{}%
  \lb@next
}
\_let\lb@compareA=\lb@compare
\_let\lb@preparesortA=\lb@preparesort
\_def\lb@compare#1\lb@eoe#2\lb@eoe{% SpecialSort:
  \_ifx\lb@sorttype\lb@namestring
     \_ifx\_sortfield\_undefined \lb@compareA#1\lb@eoe#2\lb@eoe
     \_else
        \_ea\_RetrieveFieldInFor\_ea{\_sortfield}\lb@entrykey\lb@temp
        \_ifx\lb@temp\_empty \_toks1={#1\lb@eoe}\_else \_toks1=\_ea{\lb@temp\lb@eoe}\_fi
        \_ea\_RetrieveFieldInFor\_ea{\_sortfield}\lb@currententry\lb@temp
        \_ifx\lb@temp\_empty \_toks2={#2\lb@eoe}\_else \_toks2=\_ea{\lb@temp\lb@eoe}\_fi
        \_edef\lb@temp{\_noexpand\lb@compareA\_space\_the\_toks1 \_space\_the\_toks2}\lb@temp
     \_fi
  \_else \lb@compareA#1\lb@eoe#2\lb@eoe \_fi
}
\_def\lb@preparesort#1#2\lb@eoe{%
  \_if#1-%
    \_def\lb@sorttype{#2}%
  \_else
    \_def\lb@sorttype{#1#2}%
  \_fi
  \lb@preparesortA#1#2\lb@eoe
}
\_def\_SpecialSort#1{\_def\_sortfield{#1}}
\_def\WriteImmediateInfo#1{}  % the existence of .lbr file bocks new reading of .bib
\_catcode`\@=\_tmpnum
```

Main action per every entry.

usebib.opm

```
\_def\MakeReference{\_par \_bibskip
  \_advance\_bibnum by1
  \_isdefined{_bim:\_the\_bibnum}\_iftrue
     \_edef\_tmpb{\_csname _bim:\_the\_bibnum\_endcsname}%
     \_bibmark=\_ea{\_tmpb}%
  \_else \_bibmark={}\_fi
  \_edef\_tmpb{\EntryKey}%
  \_noindent \_dest[cite:\_the\_bibnum]\_printlabel\EntryKey
  \_printbib
  {%
     \_RetrieveFieldIn{entrytype}\_entrytype
     \_csname _print:BEGIN\_endcsname
     \_isdefined{_print:\_entrytype}\_iftrue
        \_csname _print:\_entrytype\_endcsname
     \_else
        \_ifx\_entrytype\_empty \_else
           \_opwarning{Entrytype @\_entrytype\_space from [\EntryKey] undefined}%
           \_csname _print:misc\_endcsname
     \_fi\_fi
     \_csname _print:END\_endcsname
     \_ifx\_wref\_wrefrelax\_else
         \_immediate\_wref\_Xbib{{\EntryKey}{\_the\_bibnum}{\_the\_bibmark}}\_fi
  }\_par
}
```

The \bprinta, \bprintb, \bprintc, \bprintv commands used in the style files:

usebib.opm

```
\_def\_bprinta {\_bprintb*}
\_def\_bprintb #1[#2#3]{%
   \_def\_bibfieldname{#2#3}%
   \_if!#2\_relax
      \_def\_bibfieldname{#3}%
```

```
      \_RetrieveFieldIn{#3}\_bibfield
      \_ifx\_bibfield\_empty\_else
         \_RetrieveFieldIn{#3number}\_namecount
         \_def\_bibfield{\_csname _Read#3\_ea\_endcsname \_csname _pp:#3\_endcsname}%
      \_fi
   \_else
      \_RetrieveFieldIn{#2#3}\_bibfield
   \_fi
   \_if^#1^%
      \_ifx\_bibfield\_empty \_ea\_ea\_ea \_doemptyfield
      \_else \_ea\_ea\_ea \_dofullfield \_fi
   \_else \_ea \_bprintaA
   \_fi
}
\_def\_dofullfield#1#2{\_def\_dofield##1{#1}\_ea\_dofield\_ea{\_bibfield}}
\_def\_doemptyfield#1#2{\_def\_dofield##1{#2}\_ea\_dofield\_ea{\_bibfield}}
\_let\_Readauthor=\ReadAuthor  \_let\_Readeditor=\ReadEditor
\_def\_bprintaA #1#2{\_ifx\_bibfield\_empty #2\_else\_bprintaB #1**\_eee\_fi}
\_def\_bprintaB #1*#2*#3\_eee{\_if^#3^#1\_else\_ea\_bprintaC\_ea{\_bibfield}{#1}{#2}\_fi}
\_def\_bprintaC #1#2#3{#2#1#3}
\_def\_bprintc#1#2{\_bprintcA#1#2**\_relax}
\_def\_bprintcA#1#2*#3*#4\_relax{\_ifx#1\_empty \_else \_if^#4^#2\_else#2#1#3\_fi\_fi}
\_def\_bprintv [#1]#2#3{\_def\_tmpa{#2}\_def\_tmpb{#3}\_bprintvA #1,,}
\_def\_bprintvA #1,{%
   \_if^#1^\_tmpb\_else
      \_RetrieveFieldIn{#1}\_tmp
      \_ifx \_tmp\_empty
      \_else \_tmpa \_def\_tmpb{}\_def\_tmpa{}%
      \_fi
   \_ea \_bprintvA
   \_fi
}
\_sdef{_pp:author}{\_letNames\_authorname}
\_sdef{_pp:editor}{\_letNames\_editorname}
\_def\_letNames{\_let\_Firstname=\Firstname \_let\_Lastname=\Lastname
   \_let\_Von=\Von \_let\_Junior=\Junior
}
```

Various macros + multilinguas.

usebib.opm

```
\_def\_bibwarning{\_opwarning{Missing field "\_bibfieldname" in [\EntryKey]}}

\_def\_mtdef#1#2#3#4{\_sdef{_mt:#1:en}{#2} \_sdef{_mt:#1:cs}{#3}
  \_if$#4$\_slet{_mt:#1:sk}{_mt:#1:cs}
  \_else  \_sdef{_mt:#1:sk}{#4}
  \_fi
}
```

2.31.4 Usage of the `bib-iso690` style

This is the iso690 bibliographic style used by OpTEX.

See `op-example.bib` for an example of the `.bib` input. You can try it by:

```
\fontfam[LMfonts]
\nocite[*]
\usebib/s (iso690) op-example
\end
```

Common rules in `.bib` files

There are entries of type `@FOO{...}` 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 bad interpreted.

The entry names and field names are case insensitive. If there exist a data field no mentioned here then it is simply ignored. You can use it to store more information (abstract, for example).

There are “standard fields” used in ancient bibTEX (author, title, editor, edition, etc., see http://en.wikipedia.org/wiki/BibTeX). The iso690 style introduces several “non-standard” fields: ednote, numbering, isbn, issn, doi, url, citedate, key, bibmark. They are documented here.

Moreover, there are two optional special fields:

- lang = language of the entry. The hyphenation plus autogenerated phrases and abbreviations will be typeset by this language.
- option = options by which you can control special printing of vairous fields.

There can be only one option field per each entry with (may be) more options separated by spaces. You can declare the global option(s) in your document applied for each entry by `\biboptions={...}`.

The author field

All names in the author list have to be separated by “ `and` ”. Each author can be written by various formats (the `von` part is typically missing):

```
Firstname(s) von Lastname
or
von Lastname, Firstname(s)
or
von Lastname, After, Firstname(s)
```

Only the Lastname part is mandatory. Examples:

```
Petr Olšák
or
Olšák, Petr

Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
or
da Vinci, painter, Leonardo Piero
```

The separator “ `and` ” between authors will be converted to comma during printing, but between semi-final and final author the word “and” (or something different depending on current language) is printed.

The first author is printed in reverse order: “LASTNAME, Firstname(s) von, After” and the others author are printed in normal order: “Firstname(s) von LASTNAME, After”. This feature follows the ISO 690 norm. The Lastname is capitalized using uppercase letters. But if the `\caps` font modifier is defined, then it is used and printed `{\caps\_rm␣Lastname}`.

You can specify the option `aumax:`⟨*number*⟩. The ⟨*number*⟩ denotes the maximum authors to be printed. The rest of authors are ignored and the `et~al.` is appended to the list of printed authors. This text is printed only if the `aumax` value is less than the real number of authors. If you have the same number of authors in the .bib file as you need to print but you want to append `et~al.` then you can use `auetal` option.

There is an `aumin:`⟨*number*⟩ option which denotes the definitive number of printed authors if the author list is not fully printed due to `aumax`. If `aumin` is unused then `aumax` authors is printed in such case.

All authors are printed if `aumax:`⟨*number*⟩ option isn't given. There is no internal limit. But you can set the global options in your document by setting the `\biboptions` tokens list. For example:

```
\biboptions={aumax:7 aumin:1}
% if there is 8 or more authors then only first author is printed.
\entdd

Examples:
\begtt
author = "John Green and Bob Brown and Alice Black",
```

output: GREEN, John, Bob BROWN, and Alice BLACK.

```
author = "John Green and Bob Brown and Alice Black",
option = "aumax:1",
```

output: GREEN, John et al.

```
author = "John Green and Bob Brown and Alice Black",
option = "aumax:2",
```

output: GREEN, John, Bob BROWN et al.

```
author = "John Green and Bob Brown and Alice Black",
option = "aumax:3",
```

output: GREEN, John, Bob BROWN, and Alice BLACK.

```
author = "John Green and Bob Brown and Alice Black",
option = "auetal",
```

output: GREEN, John, Bob BROWN, Alice BLACK et al.

If you need to add a text before or after authors list, you can use the `auprint:{`⟨*value*⟩`}` option. The ⟨*value*⟩ will be printed instead of the authors list. The ⟨*value*⟩ can include `\AU` macro which expands to the authors list. Example:

```
author = "Robert Calbraith",
option = "auprint:{\AU\space [pseudonym of J. K. Rowling]}",
```

output: CALBRAITH Robert [pseudonym of J. K. Rowling].

You can use the `autrim:`⟨*number*⟩ option. All Firstnames of all authors are trimmed (i. e. reduced to initials) iff the number of authors in the author field is greater than or equal to ⟨*number*⟩. There is an exception: `autrim:0` means that no Firstnames are trimmed. This is default behavior. Another example: `autrim:1` means that all Firstnames are trimmed.

```
author = "John Green and Bob Brown and Alice Black",
option = "auetal autrim:1",
```

output: GREEN, J., B. BROWN, A. BLACK et al.

If you need to write a team name or institution instead authors, replace all spaces by `\␣` in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after comma. Example:

```
author = "Czech\ Technical\ University\ in\ Prague,
          Faculty\ of\ Electrical\ Engeneering",
```

output: CZECH TECHNICAL UNIVERSITY IN PRAGUE, Faculty of Electrical Engeneering.

The editor field

The editor field is used for list of the authors of the collection. The analogous rules as in author field are used here. It means that the authors are separated by " `and` ", the Firstnames, Lastnames etc. are interpreted and you can use the options `edmax:`⟨*number*⟩, `edmin:`⟨*number*⟩, `edetal`, `edtrim:`⟨*number*⟩ and `edprint:{`⟨*value*⟩`}` (with `\ED` macro). Example:

```
editor = "Jan Tomek and Petr Karas",
option = "edprint:{\ED, editors.} edtrim:1",
```

Output: J. TOMEK and P. KARAS, editors.

If `edprint` option is not set then `{\ED,␣eds.}` or `{\ED,␣ed.}` is used depending on the entry language and on the singular or plural of the editor(s).

The ednote field

The ednote field is used as the secondary authors and more editional info. The value is read as raw data without any interpretation of Lastname, Firstname etc.

```
ednote = "Illustrations by Robert \upper{Agarwal}, edited by Tom \upper{Nowak}",
```

output: Illustrations by Robert AGARWAL, edited by Tom NOWAK.

The `\upper` command have to be used for Lastnames in ednote field.

The title field

This is the title of the work. It will be printed (in common entry types) by italics. The ISO 690 norm declares, that the title plus optional subtitle are in italics and they are separated by colon. Next, the

optional secondary title have to be printed in upright font. This can be added by `titlepost:{`⟨*value*⟩`}`. Example:

```
title = "The Simple Title of The Work",
or
title = "Main Title: Subtitle",
or
title  = "Main Title: Subtitle",
option = "titlepost:{Secondary title}",
```

The output of the last example: *Main Title: Subtitle*. Secondary title.

The edition field
This field is used only for second or more edition of cited work. Write only the number without the word "edition". The shortcut "ed." (or something else depending on current language) is added automatically. Examples:

```
edition = "Second",
edition = "2nd",
edition = "2$^{\rm nd}$",
edition = "2.",
```

Output of the last example: 2. ed.

```
edition = "2."
lang    = "cs",
```

Output: 2. vyd.

Note, that the example `edition␣=␣"Second"` may cause problems. If you are using language "cs" then the output is bad: Second vyd. But you can use `editionprint:{`⟨*value*⟩`}` option. The the ⟨*value*⟩ is printed instead of edition field and shortcut. The edition field must be set. Example:

```
edition = "whatever",
option  = "editionprint:{Second full revised edition}",
```

Output: Second full revised edition.

You can use `\EDN` macro in `editionprint` value. This macro is expanded to the edition value. Example:

```
edition = "Second",
option  = "editionprint:{\EDN\space full revised edition}",
or
edition = "Second full revised edition",
option  = "editionprint:{\EDN}",
```

The address, publisher, year fields
This is an anachronism from ancient BibTEX (unfortunately no exclusive) that the address field includes only the city of the publisher residence. No more data are here. The publisher field includes the name of the publisher.

```
address = "Berlin",
publisher = "Springer Verlag",
year = 2012,
```

Output: Berlin: Springer Verlag, 2012.

Note, that the year needn't to be inserted into quotes because it is pure numeric.

The letter a, b etc. are appended to the year automatically, if two or more subsequent entries in the bibliography list are not distinct by the first author and year fields. If you needn't this feature, you can use the `noautoletters` option.

You can use "yearprint:⟨*value*⟩" option. If it is set then the ⟨*value*⟩ is used for printing year instead the real field value. The reason: year is sort sensitive, may be you need to print something else than only sorting key. Example:

```
year   = 2000,
option = "yearpint:{© 2000}",
```

Output: © 2000, sorted by: 2000.

```
year   = "2012a",
option = "yearprint:{2012}",
```

Output: 2012, sorted by: 2012a.

The address, publisher and year are typically mandatory fields. If they are missing then the warning occurs. But you can set `unpublished` option. Then this warning is suppressed. There is no difference in the printed output.

The url field

Use it without `\url` macro, but with `http://` prefix. Example:

```
url = "http://petr.olsak.net/opmac.html",
```

The ISO 690 norm recommends to add the text "Available from" (or something else if different current language is used) before URL. It means, that the output of previous example is:

Available from http://petr.olsak.net/opmac.html.

If the `cs` language is the current one than the output is:

Dostupné z: http://petr.olsak.net/opmac.html.

If the `urlalso` option is used, then the added text has the form "Available also from" or "Dostupné také z:" (if `cs` language is current).

The citedate field

This is the citation date. The field must be in the form year/month/day. It means, that the two slashes must be written here. The output depends on the current language. Example:

```
citedate = "2004/05/21",
```

Output when `en` is current: [cit. 2004-05-21].
Output when `cs` is current: [vid. 21. 5. 2004].

The howpublished field

This declares the available medium for cited document if it is not in printed form. Alternatives: online, CD, DVD, etc. Example:

```
howpublished = "online",
```

Output: [online].

The volume, number, pages and numbering fields

The volume is the "big mark" of the journal issue and the number is the "small mark" of the journal issue and pages includes the page range of the cited article in the journal. The volume is prefixed by Vol. , the number by No. and the pages by pp. . But these prefixes depends on the language of the entry.

Example:

```
volume = 31,
number = 3,
pages  = "37--42",
```

Output: Vol. 31, No. 3, pp. 37–42.

```
volume = 31,
number = 3,
pages  = "37--42",
lang   = "cs",
```

Output: ročník 31, č. 3, s. 37–42.

If you disagree with the default prefixes, you can use the numbering field. When it is set then it is used instead of volume, number, pages fields and instead of any mentioned prefixes. The numbering can include macros `\VOL`, `\NO`, `\PP`, which are expanded to the respective values of fields. Example:

```
volume    = 31,
number    = 3,
pages     = "37--42"
numbering = "Issue~\VOL/\NO, pages~\PP",
```

Output: Issue 31/3, pages 37–42

Note: The volume, numbers and pages fields are printed without numbering filed only in the `@ARTICLE` entry. It means, that if you need to visible them in the `@INBOOK`, `@INPROCEEDINGS` etc. entries, then you must to use numbering field.

Common notes about entries

The order of the fields in the entry is irrelevant. We use the printed order in this manual. The exclamation mark (!) denotes the mandatory field. If such field is missing then the warning occurs during processing.

If the `unpublished` option is set then the fields address, publisher, year, isbn and pages are not mandatory. If the `nowarn` option is set then no warnings about missing mandatory fields occurs.

If the field is used but not mentioned in the entry documentation below then it is silently ignored.

• The `@BOOK` entry

This is used for book-like entries.

Fields: author(!), title(!), howpublished, edition, ednote, address(!), publisher(!), year(!), citedate, series, isbn(!), doi, url, note.

The ednote field here means the secondary authors (illustrator, cover design etc.).

• The `@ARTICLE` entry

This is used for articles published in a journal.

Fields: author(!), title(!), journal(!), howpublished, address, publisher, month, year, [numbering or volume, number, pages(!)], citedate, issn, doi, url, note.

If the numbering is used then it is used instead volume, number, pages.

• The `@INBOOK` entry

This is used for the part of a book.

Fields: author(!), title(!), booktitle(!), howpublished, edition, ednote, address(!), publisher(!), year(!), numbering, citedate, series, isbn or issn, doi, url, note.

The author field is used for author(s) of the part, the editor field includes author(s) or editor(s) of whole document. The pages field specifies the page range of the part. The series field can include more information about the part (chapter numbers etc.).

The `@INPROCEEDINGS` and `@CONFERENCE` entries are equivalent to `@INBOOK` entry.

• The `@THESIS` entry

This is used for student's thesis.

Fields: author(!), title(!), howpublished, address(!), school(!), month, year(!), citedate, type(!), ednote, doi, url, note.

The type field must include the text "Master's Thesis" or something similar (depending on the language of the outer document).

There are nearly equivalent entries: `@BACHELORSTHESIS`, `@MASTERSTHESIS` and `@PHDTHESIS`. These entries set the type field to an appropriate value automatically. The type field is optional in such case. If it is used then it has a precedence before default setting.

• The @MISC entry

It is intended for various usage.

Fields: author, title, howpublished, ednote, citedate, doi, url, note.

You can use `\AU`, `\ED`, `\EDN`, `\VOL`, `\NO`, `\PP`, `\ADDR`, `\PUBL`, `\YEAR` macros in ednote field. These macros print authors list, editors list, edition, volume, number, pages, address, publisher and year field values respectively.

The reason of this entry is to give to you the possibility to set the format of entry by your own decision. The most of data are concentrated in ednote field.

• The `@BOOKLET`, `@INCOLLECION`, `@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 orientes bib-marks instead numbers. This style file autogenerates these marks in the form "Lastname of the first author, comma, space, the year" if bibmark

field isn't declared. If you need to set an exception from this common format, then you can use bibmark field.

The OPmac trick http://petr.olsak.net/opmac-tricks-e.html#bibmark describes how to redefine the algorithm for bibmark auto-generating when you need the short form of the type [Au13].

Sorting

If `\usebib/c` is used then entries are sorted by citation order in the text. If `\usebib/s` is used then entries are sorted by "Lastname, Firstname(s)" of the first author and if more entries have this value equal, then the year is used (from older to newer). This feature follows the recommendation of the ISO 690 norm.

If you have the same authors and the same year, you can control the sorting by setting years as 2013, 2013a, 2013b, etc. You can print something different to the list using `yearprint{`⟨*value*⟩`}` option, see the section about address, publisher and year above. The real value of year field (ie. not yearprint value) is also used in the text oriented bib-marks when `\nonumcitations` is set.

If you have some problems with name sorting, you can use the hidden field `key`, which is used for sorting instead of the "Lastname Firstname(s)" of authors. If the `key` field is unset then the "Lastname Firstname(s)" is used for sorting normally. Example:

```
author    = "Světla Čmejrková",
key       = "Czzmejrkova Svetla",
```

This entry is now sorted between C and D.

The norm recommends to place the autocitations to the top of the list of references. You can do this by setting `key␣=␣"@"`, to each entry with your name because the `@` character is sorted before `A`.

Languages

There is the language of the outer document and the languages of each entry. The ISO 690 norm recommends that the technical notes (the prefix before URL, the media type, the "and" conjunction between semifinal and final author) may be printed in the language of the outer document. The data of the entry have to be printed in the entry language (edition ed./vyd., Vol./ročník, No./č. etc.). Finally there are the phrases independent on the language (for example In:). Unfortunately, the bibTEX supposes that the entry data are not fully included in value parts of the fields (see edition, volume etc. fields) so the automaton have to add some text during processing. But what language have to be chosen?

The current value of the `\language` register at the start of the `.bib` processing is decided as the language of the outer document. This language is used for technical notes regardless of the entry language. Each entry can have the lang field with the two-letter mark of the entry language. This language is used for ed./vyd., vol./ročník etc. and it is used for hyphenation too. If the entry language is not set then the outer document language is used.

If the outer document language is known before creating of the `.bib` file, you can store some language-dependent phrases into it. On the other hand, if the main document language is unknown, you can use the `\Mtext` macro to create the text multilingual. Example:

```
howpublished = "\Mtext{blue-ray}"
```

Now, you can set the variants of blue-ray into your macros:

```
\mtdef {blue-ray}  {Blue-ray disc}  {Blue-ray disk} {}
```

Tips for using more languages

This style prefers English, Czech and Slovak languages. However, you can add more languages. Use the shortcuts of language names (`de` and `pl` in the example below). You can define all phrases for your language:

```
\def\mtdefx#1#2#3{\sdef{_mt:#1:de}{#2}\sdef{_mt:#1:pl}{#3}}

                            % German              % Polish
\mtdefx {bib.and}           { und }               { a }
\mtdefx {bib.phdthesis}     {Ph.D. Dissertation}  {Praca doktorska}
...
```

See more about language phrases in the 2.36.3 section.

Summary of non-standard fields

This style uses the following fields unkown by bibTEX:

```
option    ... options separated by spaces
lang      ... the language two-letter code of one entry
ednote    ... editional info (secondary authors etc.) or
              global data in @MISC-like entries
citedate  ... the date of the citation in year/month/day format
numbering ... format for volume, number, pages
isbn      ... ISBN
issn      ... ISSN
doi       ... DOI
url       ... URL
```

Summary of options

```
aumax:<number>       ... maximum number of printed authors
aumin:<number>       ... number of printed authors if aumax exceedes
autrim:<number>      ... full Firstnames iff number of authors are less than this
auprint:{<value>}    ... text instead authors list (\AU macro may be used)
edmax, edmin, edtrim ... similar as above for editors list
edprint:{<value>}    ... text instead editors list (\ED macro may be used)
titlepost:{<value>}  ... text after title
yearprint:{<value>}  ... text instead real year (\YEAR macro may be used)
editionprint:{<value>} . text instead real edition (\EDN macro may be used)
urlalso      ... the ``available also from'' is used instead ``available from''
unpublished  ... the publisher etc. fields are not mandatory
nowarn       ... no mandatory fields
```

Another options in the option field are silently ignored.

2.31.5 Implementation of the bib-iso690 style

bib-iso690.opm

```
% bibliography style (iso690), version <2020-03-10>, loaded on demand by \usebib

\_ifx\_optexbibstyle\_undefined \_errmessage
   {This file can be read by: \_string\usebib/? (iso690) bibfiles command only}
   \_endinput \_fi
```

_maybetod (alias \. in the style file group) does not put second dot.

bib-iso690.opm

```
\_def\_maybedot{\_ifnum\_spacefactor=\_sfcode`\.\_relax\_else.\_fi}
\_tmpnum=\_sfcode`\. \_advance\_tmpnum by-2 \_sfcode`\.=\_tmpnum
\_sfcode`\?=\_tmpnum \_sfcode`\!=\_tmpnum
\_let\.=\_maybedot  % prevents from double periods
```

Option field.

bib-iso690.opm

```
\_CreateField {option}
\_def\_isbiboption#1#2{\_edef\_tmp{\_noexpand\_isbiboptionA{#1}}\_tmp}
\_def\_isbiboptionA#1{\_def\_tmp##1 #1 ##2\_relax{%
     \_if^##2^\_csname iffalse\_ea\_endcsname \_else\_csname iftrue\_ea\_endcsname \_fi}%
   \_ea\_tmp\_biboptionsi #1 \_relax}
\_def\_bibopt[#1]#2#3{\_isbiboption{#1}\_iftrue\_def\_tmp{#2}\_else\_def\_tmp{#3}\_fi\_tmp}
\_def\_biboptionvalue#1#2{\_def\_tmp##1 #1:##2 ##3\_relax{\_def#2{##2}}%
   \_ea\_tmp\_biboptionsi #1: \_relax}

\_def\_readbiboptions{%
   \_RetrieveFieldIn{option}\_biboptionsi
   \_toks1=\_ea{\_biboptionsi}%
   \_edef\_biboptionsi{\_space \_the\_toks1 \_space \_the\_biboptions \_space}%
}
\_newtoks\_biboptions
\_public \biboptions ;
```

Formating of Author/Editor lists.

bib-iso690.opm

```
\_def\_firstauthorformat{%
   \_upper{\_Lastname}\_bprintc\_Firstname{, *}\_bprintc\_Von{ *}\_bprintc\_Junior{, *}%
}
\_def\_otherauthorformat{%
   \_bprintc\_Firstname{* }\_bprintc\_Von{* }\_upper{\_Lastname}\_bprintc\_Junior{, *}%
}
\_def\_commonname{%
  \_ifnum\_NameCount=1
     \_firstauthorformat
     \_ifx\_dobibmark\_undefined \_edef\_dobibmark{\_Lastname}\_fi
  \_else
     \_ifnum0\_namecount=\_NameCount
        \_ifx\_maybeetal\_empty \_bibconjunctionand\_else , \_fi
     \_else , \_fi
     \_otherauthorformat
  \_fi
}
\_def\_authorname{%
   \_ifnum\_NameCount>0\_namecount\_relax\_else \_commonname \_fi
   \_ifnum\_NameCount=0\_namecount\_relax \_maybeetal \_fi
}
\_let\_editorname=\_authorname

\_def\_prepareauedoptions#1{%
   \_def\_mabyetal{}\_csname lb@abbreviatefalse\_endcsname
   \_biboptionvalue{#1max}\_authormax
   \_biboptionvalue{#1min}\_authormin
   \_biboptionvalue{#1pre}\_authorpre
   \_biboptionvalue{#1print}\_authorprint
   \_isbiboption{#1etal}\_iftrue \_def\_maybeetal{\_Mtext{bib.etal}}\_fi
   \_biboptionvalue{#1trim}\_autrim
   \_let\_namecountraw=\_namecount
   \_ifx\_authormax\_empty \_else
      \_ifnum 0\_authormax<0\_namecount
         \_edef\_namecount{\_ifx\_authormin\_empty\_authormax\_else\_authormin\_fi}%
         \_def\_maybeetal{\_Mtext{bib.etal}}%
   \_fi\_fi
   \_ifx\_autrim\_empty \_def\_autrim{10000}\_fi
   \_ifnum\_autrim=0 \_def\_autrim{10000}\_fi
   \_ifnum 0\_namecount<\_autrim\_relax \_else \_AbbreviateFirstname \_fi
}
\_def\_maybeetal{}

\_ifx\upper\_undefined
   \_ifx\caps \_undefined \_def\upper{\_uppercase\_ea}\_else
                          \_def\upper#1{{\caps\_rm #1}}\_fi
\_fi
\_let\_upper=\upper
```

Preparing bib-mark (used when `\nonumcitations` is set).

bib-iso690.opm

```
\_def\_setbibmark{%
   \_ifx\_dobibmark\_undefined \_def\_dobibmark{}\_fi
   \_RetrieveFieldIn{bibmark}\_tmp
   \_ifx\_tmp\_empty \_RetrieveFieldIn{year}\_tmp \_edef\_tmp{\_dobibmark, \_tmp}\_fi
   \_bibmark=\_ea{\_tmp}%
}
% Multilinguals:           English                Czech                Slovak

\_mtdef{bib.and}           {, and }               { a }                {}
\_mtdef{bib.etal}          { et al.}              { a~kol.}            {}
\_mtdef{bib.edition}       { ed.}                 { vyd.}              {}
\_mtdef{bib.bachthesis}    {Bachelor's Thesis}    {Bakalářská práce}   {Bakalárska práca}
\_mtdef{bib.masthesis}     {Master's Thesis}      {Diplomová práce}    {Diplomová práca}
\_mtdef{bib.phdthesis}     {Ph.D. Thesis}         {Disertační práce}   {Dizertačná práca}
\_mtdef{bib.available}     {Available from }      {Dostupné na }       {}
\_mtdef{bib.availablealso} {Available also from } {Dostupné též na }  {Dotupné tiež na }
\_mtdef{bib.citedate}      {cit.~}                {vid.~}              {}
\_mtdef{bib.volume}        {Vol.~}                {ročník~}            {}
```

```
\_mtdef{bib.number}        {No.~}              {č.~}             {}
\_mtdef{bib.prepages}      {pp.~}              {s.~}             {}
\_mtdef{bib.postpages}     {~p.}               {~s.}             {}
\_mtdef{bib.editor}        {,~ed.}             {,~editor}        {}
\_mtdef{bib.editors}       {,~eds.}            {,~editoři}       {,~editori}

\_def\_bibconjunctionand{\_Mtext{bib.and}}
\_def\_preurl{\_Mtext{bib.available}}
\_let\_predoi=\_preurl
\_def\_postedition{\_mtext{bib.edition}}
\_def\_Inclause{In:~}
\_def\_prevolume{\_mtext{bib.volume}}
\_def\_prenumber{\_mtext{bib.number}}
\_def\_prepages{\_mtext{bib.prepages}}
\_def\_posteditor{\_ifnum0\_namecountraw>1 \_Mtext{bib.editors}\_else\_Mtext{bib.editor}\_fi}

\_chardef\_documentlanguage=\_language
\_def\_Mtext#1{\_csname _mt:#1:\_csname _lan:\_the\_documentlanguage\_endcsname\_endcsname}

\_CreateField {lang}
\_def\_setlang#1{\_ifx#1\_empty \_else
      \_ea \_ifx \_csname _#1Patt\_endcsname \_relax
         \_opwarning{The language "#1" used in .bib file is unknown}
      \_else \_language=\_csname _#1Patt\_endcsname
   \_fi\_fi
}
```

Non-standard fieldnames.

bib-iso690.opm

```
\_CreateField {ednote}
\_CreateField {citedate}
\_CreateField {numbering}
\_CreateField {isbn}
\_CreateField {issn}
\_CreateField {doi}
\_CreateField {url}
\_CreateField {bibmark}
```

Sorting.

bib-iso690.opm

```
\_SortingOrder{name,year}{lfvj}
\_SpecialSort {key}
```

Supporting macros.

bib-iso690.opm

```
\_def\_bibwarninga{\_bibwarning}
\_def\_bibwarningb{\_bibwarning}

\_def\_docitedate #1/#2/#3/#4\_relax{[\_Mtext{bib.citedate}%
   \_if^#2^#1\_else
      \_if^#3^#1/#2\_else \_docitedateA{#1}{#2}{#3}%
   \_fi\_fi ]%
}
\_def\_docitedateA#1#2#3{%
   \_ifnum\_documentlanguage=\_csPatt \_docitedateCS{#1}{#2}{#3}%
   \_else \_ifnum\_documentlanguage=\_skPatt \_docitedateSK{#1}{#2}{#3}%
         \_else \_docitedateEN{#1}{#2}{#3}%
   \_fi\_fi
}
\_def\_docitedateEN#1#2#3{#1-#2-#3}
\_def\_docitedateCS#1#2#3{\_hbox{\_tmpnum=#3 \_the\_tmpnum. \_tmpnum=#2 \_the\_tmpnum. #1}}
\_let\_docitedateSK=\_docitedateCS

\_def\_doyear#1{
   \_biboptionvalue{yearprint}\_yearprint
   \_ifx\_yearprint\_empty#1\_else\_def\YEAR{#1}\_yearprint\_fi
}
\_def\_preparenumbering{%
   \_def\VOL{\_RetrieveField{volume}}%
   \_def\NO{\_RetrieveField{number}}%
```

```
    \_def\PP{\_RetrieveField{pages}}%
}
\_def\_prepareednote{%
    \_def\EDN{\_RetrieveField{edition}}%
    \_def\ADDR{\_RetrieveField{address}}%
    \_def\PUBL{\_RetrieveField{publisher}}%
    \_def\YEAR{\_RetrieveField{year}}%
    \_def\AU{\_bprintb[!author]{\_doauthor0{####1}}{}}%
    \_def\ED{\_bprintb[!editor]{\_doeditor0{####1}}{}}%
    \_preparenumbering
}
\_def\_doedition#1{%
    \_biboptionvalue{editionprint}\_editionprint
    \_ifx\_editionprint\_empty#1\_postedition\_else\_def\ED{#1}\_editionprint\_fi
}
\_def\_doauthor#1#2{\_prepareauedoptions{au}\_let\_iseditorlist=\_undefined
    \_if1#1\_def\AU{#2}\_else\_let\_authorprint=\_empty\_fi
    \_ifx\_authorprint\_empty #2\_else \_authorprint\_fi
}
\_def\_doeditor#1#2{\_prepareauedoptions{ed}\_let\_firstauthorformat=\_otherauthorformat
    \_if1#1\_def\ED{#2}\_else\_let\_authorprint=\_empty\_fi
    \_ifx\_authorprint\_empty #2\_posteditor\_else \_authorprint\_fi
}
```

Entry types.

bib-iso690.opm

```
\_sdef{_print:BEGIN}{%
    \_readbiboptions
    \_biboptionvalue{titlepost}\_titlepost
    \_isbiboption{unpublished}\_iftrue \_let\_bibwarninga=\_relax \_let\_bibwarningb=\_relax \_fi
    \_isbiboption{nowarn}\_iftrue \_let\_bibwarning=\_relax \_fi
    \_isbiboption{urlalso}\_iftrue \_def\_preurl{\_Mtext{bib.availablealso}}\_fi
    \_RetrieveFieldIn{lang}\_langentry \_setlang\_langentry
}
\_sdef{_print:END}{%
    \_bprinta [note]       {*.}{}%
    \_setbibmark
}
\_def\_bookgeneric#1{%
    \_bprinta [howpublished]  {[*].\ }{}%
    \_bprintb [edition]    {\_doedition{##1}\.\ }{}%
    \_bprinta [ednote]     {*.\ }{}%
    \_bprinta [address]    {*\_bprintv[publisher]{:}{\_bprintv[year]{,}{.}}\ }{\_bibwarninga}%
    \_bprinta [publisher]  {*\_bprintv[year]{,}{.}\ }{\_bibwarninga}%
    \_bprintb [year]       {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}{}}{.}\ }%
                                                                          {\_bibwarning}%
    \_bprinta [numbering]  {\_preparenumbering*\_bprintv[citedate]{}{\.}\ }{}%
    \_bprinta [citedate]   {\_docitedate*///\_relax.\ }{}%
    #1%
    \_bprinta [series]     {*.\ }{}%
    \_bprinta [isbn]       {ISBN~*.\ }{\_bibwarningb}%
    \_bprinta [issn]       {ISSN~*.\ }{}%
    \_bprintb [doi]        {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
    \_bprintb [url]        {\_preurl\_url{##1}. }{}%
}
\_sdef{_print:book}{%
    \_bprintb [!author]    {\_doauthor1{##1}\.\ }{\_bibwarning}%
    \_bprintb [title]      {{\_em##1}\_bprintc\_titlepost{\.\ *}\_bprintv[howpublished]{}{\.}\ }%
                                                                          {\_bibwarning}%
    \_bookgeneric{}%
}
\_sdef{_print:article}{%
    \_biboptionvalue{journalpost}\_journalpost
    \_bprintb [!author]    {\_doauthor1{##1}\.\ }{\_bibwarning}%
    \_bprinta [title]      {*.\ \_bprintc\_titlepost{*.\ }}{\_bibwarning}%
    \_bprintb [journal]    {{\_em##1}\_bprintc\_journalpost{\.\ *}\_bprintv[howpublished]{}{\.}\ }%
                                                                          {\_bibwarninga}%
    \_bprinta [howpublished]  {[*].\ }{}%
    \_bprinta [address]    {*\_bprintb[publisher]{:}{,}\ }{}%
```

```
    \_bprinta [publisher] {*, }{}%
    \_bprinta [month]     {*, }{}%
    \_bprintb [year]      {\_doyear{##1}\_bprintv[volume,number,pages]{,}{\.}\ }{}%
    \_bprinta [numbering] {\_preparenumbering*\_bprintv[citedate]{}{\.}\ }
                         {\_bprinta [volume] {\_prevolume*\_bprintv[number,pages]{,}{\.}\ }{}%
                          \_bprinta [number] {\_prenumber*\_bprintv[pages]{,}{\.}\ }{}%
                          \_bprintb [pages]  {\_prepages\_hbox{##1}\_bprintv[citedate]{}{\.}\ }%
                                                                                {\_bibwarninga}}%
    \_bprinta [citedate]  {\_docitedate*///\_relax.\ }{}%
    \_bprinta [issn]      {ISSN~*.\ }{}%
    \_bprintb [doi]       {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
    \_bprintb [url]       {\_preurl\_url{##1}. }{}%
}
\_sdef{_print:inbook}{%
    \_let\_bibwarningb=\_relax
    \_bprintb [!author]   {\_doauthor1{##1}\.\ }{\_bibwarning}%
    \_bprinta [title]     {*.\ }{\_bibwarning}%
                         \_Inclause
    \_bprintb [!editor]   {\_doeditor1{##1}\.\ }{}%
    \_bprintb [booktitle] {{\_em##1}\_bprintc\_titlepost{\.\ *}\_bprintv[howpublished]{}{\.}\ }%
                                                                                {\_bibwarning}%
    \_bookgeneric{\_bprintb [pages]  {\_prepages\_hbox{##1}. }{}}%
}
\_slet{_print:inproceedings}{_print:inbook}
\_slet{_print:conference}{_print:inbook}

\_sdef{_print:thesis}{%
    \_bprintb [!author]    {\_doauthor1{##1}\.\ }{\_bibwarning}%
    \_bprintb [title]      {{\_em##1}\_bprintc\_titlepost{\.\ *}\_bprintv[howpublished]{}{\.}\ }%
                                                                                {\_bibwarning}%
    \_bprinta [howpublished]  {[*].\ }{}%
    \_bprinta [address]    {*\_bprintv[school]{:}{\_bprintv[year]{,}{.}}\ }{\_bibwarning}%
    \_bprinta [school]     {*\_bprintv[year]{,}{.}\ }{\_bibwarning}%
    \_bprinta [month]      {*, }{}%
    \_bprintb [year]       {\_doyear{##1}\_bprintv[citedate]{}{.}\ }{\_bibwarninga}%
    \_bprinta [citedate]   {\_docitedate*///\_relax.\ }{}%
    \_bprinta [type]       {*\_bprintv[ednote]{,}{.}\ }%
                          {\_ifx\_thesistype\_undefined\_bibwarning
                           \_else\_thesistype\_bprintv[ednote]{,}{.}\ \_fi}%
    \_bprinta [ednote]     {*.\ }{}%
    \_bprintb [doi]        {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
    \_bprintb [url]        {\_preurl\_url{##1}. }{}%
}
\_sdef{_print:phdthesis}{\_def\_thesistype{\_Mtext{bib.phdthesis}}\_cs{_print:thesis}}
\_sdef{_print:mastersthesis}{\_def\_thesistype{\_Mtext{bib.masthesis}}\_cs{_print:thesis}}
\_sdef{_print:bachelorsthesis}{\_def\_thesistype{\_Mtext{bib.bachthesis}}\_cs{_print:thesis}}

\_sdef{_print:generic}{%
    \_bprintb [!author]    {\_doauthor1{##1}\.\ }{\_bibwarning}%
    \_bprintb [title]      {{\_em##1}\_bprintc\_titlepost{\.\ *}\_bprintv[howpublished]{}{\.}\ }%
                                                                                {\_bibwarning}%
    \_bprinta [howpublished]  {[*].\ }{}%
    \_bprinta [ednote]     {\_prepareednote*\_bprintv[citedate]{}{.}\ }{\_bibwarning}%
    \_bprinta [year]       {}{\_bibwarning}%
    \_bprinta [citedate]   {\_docitedate*///\_relax.\ }{}%
    \_bprintb [doi]        {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
    \_bprintb [url]        {\_preurl\_url{##1}. }{}%
}
\_slet{_print:booklet}{_print:generic}
\_slet{_print:incolleciton}{_print:generic}
\_slet{_print:manual}{_print:generic}
\_slet{_print:proceedings}{_print:generic}
\_slet{_print:techreport}{_print:generic}
\_slet{_print:unpublished}{_print:generic}

\_sdef{_print:misc}{\_let\_bibwarning=\_relax \_cs{_print:generic}}
```

2.32 Sorting and making Index

makeindex.opm

```
\_codedecl \makeindex {Makeindex and sorting <2020-04-26>} % loaded in format
```

\makeindex implements sorting algorithm at TEX macro-language level. You need not any external program.

There are two passes in sorting algorithm. Primary pass does not distinguish between a group o letters (typically non-accented and accented). If the result of comparing two string is equal in primary pass then secondary pass is started. It distinguish between variously accented letters. Czech rules, for example says: not accented before dieresis before acute before circumflex before ring. At less priority: lowercase letters must be before uppercase letters.

The _sortingdata⟨*iso-code*⟩ implements these rules for the language ⟨*iso-code*⟩. The groups between commas are not distinguished in the first pass. The second pass distinguishes all characters mentioned in the _sortingdata⟨*iso-code*⟩ (commas are ignored). The order of letters in the _sortingdata⟨*iso-code*⟩ macro is significant for sorting algorithm. The Czech rules (cs) are implemented here:

makeindex.opm

```
\_def \_sortingdatacs {%
  /,{ },-,&,@,%
  aAäÄáÁ,%
  bB,%
  cC,%
  čČ,%
  dDdĎ,%
  eEéÉěĚ,%
  fF,%
  gG,%
  hH,%
  ^^T^^U^^V,% ch Ch CH
  iIíÍ,%
  jJ,%
  kK,%
  lLÍĹŀĽ,%
  mM,%
  nNňŇ,%
  oOöÖóÓôÔ,%
  pP,%
  qQ,%
  rRŕŘ,%
  řŘ,%
  sS,%
  šŠ,%
  tTťŤ,%
  uUüÜúÚůŮ,%
  vV,%
  wW,%
  xX,%
  yYýÝ,%
  zZ,%
  žŽ,%
  0,1,2,3,4,5,6,7,8,9,'%
}
```

Characters ignored by sorting algorithm are declared in _ignoredchars⟨*iso-code*⟩. The compound characters (two or more characters interpreted as one character in sorting algorithm) is mapped to single invisible characters in _compoundchars⟨*iso-code*⟩. Czech rules declares ch or Ch or CH as a single letter sorted between H and I. See _sortingdatacs above where these declared characters are used.

The characters declared in _ignoredchars are ignored in first pass without additional condition. All characters are taken into account in second pass: ASCII characters with code ‘65 are sorted first if they are not mentioned in the _sortingdata⟨*iso-code*⟩ macro. Others not mentioned characters have undefined behavior during sorting.

makeindex.opm

```
\_def \_ignoredcharscs  {.,;?!:'"|()[]<>=+}
\_def \_compoundcharscs {ch:^^T Ch:^^U CH:^^V} % DZ etc. are sorted normally
```

Slovak sorting rules are the same as Czech. The macro `\_sortingdatacs` includes Slovak letters too. Compound characters are the same. English sorting rules can be defined by `\_sortingdatacs` too because English alphabet is subset of Czech and Slovak alphabets. Only difference: `\_compoundcharsen` is empty in English rules.

You can declare these macros for more languages, if you wish to use `\makeindex` with sorting rules in respect to your language. Note: if you need to map compound characters to a character, don't use `^^I` or `^^M` because these characters have very specific category code. And use space to separate more mappings, like in `\_compoundcharscs` above.

makeindex.opm

```
\_let \_sortingdatask = \_sortingdatacs
\_let \_compoundcharssk = \_compoundcharscs
\_let \_ignoredcharssk = \_ignoredcharscs
\_let \_sortingdataen = \_sortingdatacs
\_def \_compoundcharsen {}
\_let \_ignoredcharsen = \_ignoredcharscs
```

Preparing to primary pass is implemented by the `\_setprimarysorting` macro. It is called from `\makeindex` macro and all processing of sorting is in a group.

makeindex.opm

```
\_def\_setprimarysorting {%
   \_ea\_let \_ea\_sortingdata \_csname _sortingdata\_sortinglang\endcsname
   \_ea\_let \_ea\_compoundchars \_csname _compoundchars\_sortinglang\endcsname
   \_ea\_let \_ea\_ignoredchars \_csname _ignoredchars\_sortinglang\endcsname
   \_ifx \_sortingdata\_relax \_addto\_nold{ sortingdata}%
       \_let \_sortingdata = \_sortingdataen \fi
   \_ifx \_compoundchars\_relax \_addto\_nold{ compoundchars}%
       \_let \_compoundchars = \_compoundcharsen \fi
   \_ifx \_ignoredchars\_relax \_addto\_nold{ ignoredchars}%
       \_let \_ignoredchars = \_ignoredcharsen \fi
   \_ifx \_compoundchars\_empty \_else
      \_edef \_compoundchars {\_detokenize\_ea{\_compoundchars} }\_fi % all must be catcode 12
   \_def \_act ##1{\_ifx##1\_relax \_else
      \_ifx##1,\_advance\_tmpnum by1
      \_else \_lccode`##1=\_tmpnum \_fi
      \_ea\_act \_fi}%
   \_tmpnum=65 \_ea\_act \_sortingdata \_relax
   \_def \_act ##1{\_ifx##1\_relax \_else
      \_lccode`##1=`\^^I
      \_ea\_act \_fi}%
   \_ea\_act \_ignoredchars \_relax
}
```

Preparing to secondary pass is implemented by the `\_setsecondarysorting` macro.

makeindex.opm

```
\_def\_setsecondarysorting {%
   \_def \_act ##1{\_ifx##1\_relax \_else
      \_ifx##1,\_else \_advance\_tmpnum by1 \_lccode`##1=\_tmpnum \_fi
      \_ea\_act \_fi}%
  \_tmpnum=65 \_ea\_act \_sortingdata \_relax
}
```

Strings to be sorted are prepared in `\,`⟨*string*⟩ control sequences (in order to save `\TeX` memory). The `\_preparesorting` `\,`⟨*string*⟩ converts ⟨*string*⟩ to `\_tmpb` with respect to the data initialized in `\_setprimarysorting` or `\_setsecondarysorting`.
The compoud characters are converted to single characters by the `\_docompound` macro.

makeindex.opm

```
\_def \_preparesorting #1{%
   \_edef \_tmpb {\_ea\_ignorefirst\_csstring #1}% \,<string> -> <string>
   \_ea \_docompound \_compoundchars \_relax:{}       % replace compound characters
   \_lowercase \_ea{\_ea\_def \_ea\_tmpb \_ea{\_tmpb}}% convert in respect to \_sortingdata
   \_ea\_replstring \_ea\_tmpb \_ea{\_csstring\^^I}{}% remove ignored characters
}
\_def \_docompound #1:#2 {%
   \_ifx\_relax#1\_else \_replstring\_tmpb {#1}{#2}\_ea\_docompound \_fi
}
\_def \_ignorefirst#1{}
```

Macro _isAleB \,⟨*string1*⟩ \,⟨*string2*⟩ returns the result of comparison of given two strings to _ifAleB control sequence. Usage: \isAleB \,⟨*string1*⟩ \,⟨*string2*⟩ _ifAleB ... _else ... _fi The converted strings (in respect of the data prepared for first pass) must be saved as values of \,⟨*string1*⟩ and \,⟨*string2*⟩ macros. The reason is speed: we don't want to convert them repeatedly in each comparison. The macro _testAleB ⟨*converted string1*⟩&_relax⟨*converted-string2*⟩_relax \,⟨*string1*⟩\,⟨*string2*⟩ does the real work. It reads first character from both converted strings, compares them and if it is equal then calls itself recursively else gives result.

makeindex.opm

```
\_newifi \_ifAleB

\_def\_isAleB #1#2{%
   \_edef\_tmpb {#1&\_relax#2&\_relax}%
   \_ea \_testAleB \_tmpb #1#2%
}
\_def\_testAleB #1#2\_relax #3#4\_relax #5#6{%
  \_if #1#3\_if #1&\_testAleBsecondary #5#6%   goto to the second pass::
          \_else \_testAleB #2\_relax #4\_relax #5#6%
          \_fi
  \_else \_ifnum `#1<`#3 \_AleBtrue \_else \_AleBfalse \_fi
  \_fi
}
\_def\_testAleBsecondary#1#2{%
  \_bgroup
     \_setsecondarysorting
     \_preparesorting#1\_let\_tmpa=\_tmpb \_preparesorting#2%
     \_edef\_tmpb{\_tmpa0\_relax\_tmpb1\_relax}%
     \_ea\_testAleBsecondaryX \_tmpb
  \_egroup
}
\_def\_testAleBsecondaryX #1#2\_relax #3#4\_relax {%
   \_if #1#3\_testAleBsecondaryX #2\_relax #4\_relax
   \_else \_ifnum `#1<`#3 \_global\_AleBtrue \_else \_global \_AleBfalse \_fi
   \_fi
}
```

Merge sort is very effectively implemented by TEX macros. The following code is created by my son Miroslav. The _mergesort macro expects that all items in _iilist are separated by comma when it starts. It ends with sorted items in _iilist without commas. So _dosorting macro must prepare commas between items.

makeindex.opm

```
\_def\_mergesort #1#2,#3{% by Miroslav Olsak
   \_ifx,#1%                      % prazdna-skupina,neco,  (#2=neco #3=pokracovani)
      \_addto\_iilist{#2,}%        % dvojice skupin vyresena
      \_sortreturn{\_fif\_mergesort#3}%   % \mergesort pokracovani
   \_fi
   \_ifx,#3%                      % neco,prazna-skupina,  (#1#2=neco #3=,)
      \_addto\_iilist{#1#2,}%      % dvojice skupin vyresena
      \_sortreturn{\_fif\_mergesort}%      % \mergesort dalsi
   \_fi
   \_ifx\_end#3%                   % neco,konec (#1#2=neco)
      \_ifx\_empty\_iilist               % neco=kompletni setrideny seznam
         \_def\_iilist{#1#2}%
         \_sortreturn{\_fif\_fif\_gobbletoend}%   % koncim
      \_else                      % neco=posledni skupina nebo \end
         \_sortreturn{\_fif\_fif       % spojim \indexbuffer+necoa cele znova
                      \_edef\_iilist{\_ea}\_ea\_mergesort\_iilist#1#2,#3}%
   \_fi\_fi                       % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
   \_isAleB #1#3\_ifAleB          % p1<p2
      \_addto\_iilist{#1}%        % p1 do bufferu
      \_sortreturn{\_fif\_mergesort#2,#3}%         % \mergesort neco1,p2+neco2,
   \_else                        % p1>p2
      \_addto\_iilist{#3}%        % p2 do bufferu
      \_sortreturn{\_fif\_mergesort#1#2,}%         % \mergesort p1+neco1,neco2,
   \_fi
   \_relax % zarazka, na ktere se zastavi \sortreturn
}
\_def\_sortreturn#1#2\_fi\_relax{#1} \_def\_fif{\_fi}
\_def\_gobbletoend #1\_end{}
```

The _dosorting \list macro redefines \list as sorted \list. The \list have to include control sequences in the form \⟨*c*⟩⟨*string*⟩. These control sequences will be sorted in respect to ⟨*strings*⟩ without change of meanings of these control sequences. Their meanings are irrelevant when sorting. The first character ⟨*c*⟩ in \⟨*c*⟩⟨*string*⟩ should be whatever. It does not influence the sorting. OpTEX uses comma at this place for sorting indexes: \,⟨*word1*⟩ \,⟨*word2*⟩ \,⟨*word3*⟩ ....

The actual language (chosen for hyphenation patterns) is used for sorting data. If the _sortinglang macro is defined as ⟨*iso-code*⟩(for example \def\sortinglang{de}) then this has precedence and actual language is not used. Moreover, if you specify _asciisortingtrue then ASCII sorting will be processed and all language sorting data will be ignored.

makeindex.opm

```
\_newifi \_ifasciisorting  \_asciisortingfalse
\_def\_dosorting #1{%
   \begingroup
     \_def\_nold{}%
     \_ifx\_sotringlang\_undefined \_edef\_sortinglang{\_cs{_lan:\_the\_language}}\_fi
      \_ifasciisorting
          \_edef\_sortinglang{ASCII}%
          \_def \_preparesorting##1{\_edef\_tmpb{\_ea\_ignorefirst\_csstring##1}}%
          \_let \_setsecondarysorting=\_relax
     \_else
        \_setprimarysorting
     \_fi
     \_message{OpTeX: Sorting \_string#1 (\_sortinglang) ...^^J}%
     \_ifx\_nold\_empty\_else \_opwarning{Missing\_nold\_space for language (\_sortinglang)}\_fi
     \_def \_act##1{\_preparesorting ##1\_edef##1{\_tmpb}}%
     \_ea\_xargs \_ea\_act #1;%
     \_def \_act##1{\_addto #1{##1,}}%
     \_edef #1{\_ea}\_ea\_xargs \_ea\_act #1;%
     \_edef \_iilist{\_ea}\_ea\_mergesort #1\_end,\_end
   \_ea\_endgroup
   \_ea\_def\_ea#1\ea{\_iilist}%
}
```

The \makeindex prints the index. First, it sorts the _iilist second, it prints the sorted _iilist, each item is printed using _printindexitem.

makeindex.opm

```
\_def\_makeindex{\_par
  \_ifx\_iilist\_empty \_opwarning{index data-buffer is empty. TeX me again}
  \_else
    \_dosorting \_iilist % sorting \_iilist
    \_bgroup
       \_rightskip=0pt plus1fil \_exhyphenpenalty=10000 \_leftskip=\_iindent
       \_ea\_xargs \_ea\_printindexitem \_iilist ;\_par
    \_egroup
  \_fi
}
\_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

```
\_def\_printindexitem #1{%
   \_ifcsname _\_csstring #1\_endcsname
      \_ea\_ea\_ea \_printii \_csname _\_csstring #1\_endcsname &%
   \_else
      \_ea\_ea\_ea\_printii \_ea\_ignorefirst \_csstring #1&%
   \_fi
   \_ea\_printiipages #1&
}
```

_printii ⟨*word*⟩& does more intelligent work because we are working with words in the form ⟨*main-word*⟩/⟨*sub-word*⟩/⟨*sub-sub-word*⟩. The \everyii tokens register is applied before \noindent. User can declare something special here.

The _newiiletter{⟨*letter*⟩} macro is empty by default. It is invoked if first letter of index entries is changed. You can declare a design between index entries here. You can try, for example:

```
\def\_newiiletter#1#2{%
    \bigskip \hbox{\setfontsize{at15pt}\bf\uppercase{#1}}\medskip}
```

makeindex.opm

```
\_def\_printii #1#2&{%
   \_ismacro\_lastii{#1}\_iffalse \_newiiletter{#1}{#2}\_def\_lastii{#1}\_fi
   \_gdef\_currii{#1#2}\_the\_everyii\_noindent
   \_hskip-\_iindent \_ignorespaces\_printiiA#1#2//}
\_def\_printiiA #1/{\_if^#1^\_let\_previi=\_currii \_else
   \_ea\_scanprevii\_previi/&\_edef\_tmpb{\_detokenize{#1}}%
   \_ifx\_tmpa\_tmpb \_iiemdash \_else#1 \_gdef\_previi{}\_fi
   \_expandafter\_printiiA\_fi
}
\_def\_iiemdash{\_kern.1em---\_space}
\_def\_lastii{}
\_def\_newiiletter#1#2{}

\_def\_scanprevii#1/#2&{\_def\_previi{#2}\_edef\_tmpa{\_detokenize{#1}}}
\_def\_previi{} % previous index item
```

_printiipages ⟨*pglist*⟩& gets ⟨*pglist*⟩ in the form ⟨*pg*⟩:⟨*type*⟩,⟨*pg*⟩:⟨*type*⟩,...⟨*pg*⟩:⟨*type*⟩ and it converts them to ⟨*pg*⟩, ⟨*pg*⟩, ⟨*from*⟩--⟨*to*⟩, ⟨*pg*⟩ etc. The same pages must be printed only once and continuous consequences of pages must be compressed to the form ⟨*from*⟩-⟨*to*⟩. Moreover, the consequence is continuous only if all pages have the same ⟨*type*⟩. Empty ⟨*type*⟩ is most common, pages with `b` ⟨*type*⟩ must be printed as bold and with `i` ⟨*type*⟩ as italics. Moreover, the ⟨*pg*⟩ mentioned here are ⟨*gpageno*⟩, but we have to print ⟨*pageno*⟩. The following macros solves these tasks.

makeindex.opm

```
\_def\_printiipages#1&{\_let\_pgtype=\_undefined \_tmpnum=0 \_printpages #1,:,\_par}
\_def\_printpages#1:#2,{%  state automaton for compriming pages
   \_ifx,#1,\_uselastpgnum
   \_else \_def\_tmpa{#2}%
      \_ifx\_pgtype\_tmpa \_else
         \_let\_pgtype=\_tmpa
         \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
         \_tmpnum=#1 \_returnfi \_fi
      \_ifnum\_tmpnum=#1 \_returnfi \_fi
      \_advance\_tmpnum by1
      \_ifnum\_tmpnum=#1 \_ifx\_lastpgnum\_undefined \_usepgdash\_fi
                         \_edef\_lastpgnum{\_the\_tmpnum:{\_pgtype}}%
                         \_returnfi \_fi
      \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
      \_tmpnum=#1
      \_relax
   \_ea\_printpages \_fi
}
\_def\_returnfi #1\_relax{\_fi}
\_def\_uselastpgnum{\_ifx\_lastpgnum\_undefined
   \_else \_ea\_pgprint\_lastpgnum \_let\_lastpgnum=\_undefined \_fi
}
\_def\_usepgcomma{\_ifnum\_tmpnum>0, \_fi} % comma+space between page numbers
\_def\_usepgdash{\_hbox{--}}               % dash in the <from>--<to> form
```

You can re-define _pgprint ⟨*gpageno*⟩:{⟨*iitype*⟩} if you need to implement more ⟨*iitypes*⟩.

makeindex.opm

```
\_def\_pgprint #1:#2{%
   \_ifx ,#2,\_pgprintA{#1}\_returnfi \_fi
   \_ifx b#2{\_bf \_pgprintA{#1}}\_returnfi \_fi
   \_ifx i#2{\_it \_pgprintA{#1}}\_returnfi \_fi
   \_ifx u#2\_pgu{\_pgprintA{#1}}\_returnfi \_fi
  \_pgprintA{#1}\_relax
}
\_def\_pgprintA #1{\_ilink[pg:#1]{\_cs{_pgi:#1}}} % \ilink[pg:<gpageno>]{<pageno>}
\_def\_pgu#1{\_leavevmode\_vtop{\_hbox{#1}\kern.3ex\_hrule}}
```

The \iindex{⟨*word*⟩} puts one ⟨*word*⟩ to the index. It writes _Xindex{⟨*word*⟩}{⟨*iitype*⟩} to the `.ref` file. All othes variants of indexing macros expands internally to `\iindex`.

makeindex.opm

```
\_def\_iindex#1{\_isempty{#1}\_iffalse\_openref{\def~{ }%
   \edef\_act{\_noexpand\_wref\_noexpand\_Xindex{{#1}{\_iitypesaved}}}\_act}\_fi}
```

```
\_public \iindex ;
```

The _Xindex{⟨*word*⟩}{⟨*iitype*⟩} stores \,⟨*word*⟩ to the _iilist if there is first occurrence of the ⟨*word*⟩. The list of pages where ⟨*word*⟩ occurs, is the value of the macro \,⟨*word*⟩, so the ⟨*gpageno*⟩:⟨*iitype*⟩ is appended to this list. Moreower, we need a mapping from ⟨*gpageno*⟩ to ⟨*pageno*⟩, because we print ⟨*pageno*⟩ in the index, but hyperlinks are implemented by ⟨*gpageno*⟩. So, the macro _pgi:⟨*gpageno*⟩ is defined as ⟨*pageno*⟩.

makeindex.opm

```
\_def \_iilist {}
\_def \_Xindex #1#2{\_ea\_XindexA \_csname ,#1\_ea\_endcsname \_currpage {#2}}
\_def \_XindexA #1#2#3#4{% #1=\,<word> #2=<gpageno> #3=<pageno> #4=<iitype>
   \_ifx#1\relax \_global\_addto \_iilist {#1}%
                 \_gdef #1{#2:#4}%
   \else \_global\_addto #1{,#2:#4}%
   \fi
   \sxdef{_pgi:#2}{#3}%
}
```

The implementation of macros \ii, \iid, \iis follows. Note that \ii works in horizontal mode on order to the \write whatsit is not broken from the following word. If you need to keep vertical mode, use \iindex{⟨*word*⟩} directly.
The \iitype {⟨*type*⟩} saves the ⟨*type*⟩ to the _iitypesaved macro. It is used in the \iindex macro.

makeindex.opm

```
\_def\_ii #1 {\_leavevmode\_def\_tmp{#1}\_iiA #1,,\_def\_iitypesaved{}}

\_def\_iiA #1,{\_if$#1$\_else\_def\_tmpa{#1}%
   \_ifx\_tmpa\_iiatsign \_ea\_iiB\_tmp,,\_else\_iindex{#1}\_fi
   \_ea\_iiA\_fi}
\_def\_iiatsign{@}

\_def\_iiB #1,{\_if$#1$\_else \_iiC#1/\_relax \_ea\_iiB\_fi}
\_def\_iiC #1/#2\_relax{\_if$#2$\_else\_iindex{#2#1}\_fi}

\_def\_iid #1 {\_leavevmode\_iindex{#1}#1\_futurelet\_tmp\_iiD\_def\_iitypesaved{}}
\_def\_iiD{\_ifx\_tmp,\_else\_ifx\_tmp.\_else\_space\_fi\_fi}

\_def\_iis #1 #2{{\_def~{ }\_global\_sdef{_,#1}{#2}}\_ignorespaces}

\_def\_iitypesaved{}
\_def\_iitype #1{\_def\_iitypesaved{#1}\_ignorespaces}

\_public \ii \iid \iis \iitype ;
```

2.33 Footnotes and marginal notes

fnotes.opm

```
\_codedecl \fnote {Footnotes, marginal notes OpTeX <2020-03-20>} % loaded in format
```

_gfnotenum is conter which counts footnotes globally in the document. whole document, chapters, pages.
_lfnotenum is counter which counts footnotes at each chapter from one. It is used for local page footnote counters too.
_ifpgfnote says that footnote numbers are counted on each page from one. We need to run \openref in such case.
\fnotenum is a macro which expands to footnote number counted in declared part.
\fnotenumchapters declares footnotes numbered in each chapter from one (default), \fnotenumglobal declares footnotes numbered in whole document from one and \fnotenumpages declares footnotes numbered at each page from one.

fnotes.opm

```
\_newcount\_gfnotenum \_gfnotenum=0
\_newcount\_lfnotenum

\_newifi \_ifpgfnote
\_def \_fnotenumglobal   {\def\_fnotenum{\_the\_gfnotenum}\_pgfnotefalse}
\_def \_fnotenumchapters {\def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse}
```

```
\_def \_fnotenumpages    {\def\_fnotenum{\_trycs{_fn:\_the\_gfnotenum}{?}}\_pgfnotetrue}
\_fnotenumchapters  % default are footnotes counted from one in each chapter
\_def \fnotenum{\_fnotenum}
\_public \fnotenumglobal \fnotenumchapters \fnotenumpages ;
\_let \runningfnotes = \fnotenumglobal % for backward compatibility
```

The _printfnotemark prints the footnote mark. You can re-define this macro if you want another design of footnotes. For example

```
\fnotenumpages
\def \_printfnotemark {\ifcase 0\fnotenum\or
   *\or**\or***\or$^\mathbox{†}$\or$^\mathbox{‡}$\or$^\mathbox{††}$\fi}
```

This code gives footnotes* and ** and*** and† etc. and it supposes that there are no more than 6 footnotes at one page.

If you want to distinguish between footnote marks in the text and in the front of footnote itself, then you can define _printfnotemarkA and _printfnotemarkB.

The \fnotelinks⟨*colorA*⟩⟨*colorB*⟩ implements the hyperlinked footnotes (from text to footnote and backward).

fnotes.opm

```
\_def \_printfnotemark  {$^{\_fnotenum}$}   % default footnote mark
\_def \_printfnotemarkA {\_printfnotemark}  % footnote marks used in text
\_def \_printfnotemarkB {\_printfnotemark}  % footnote marks used in front of footnotes

\_def \_fnotelinks#1#2{% <inText color> <inFootnote color>
   \_def\_printfnotemarkA{\_link[fnt:\the\_gfnotenum]{\_localcolor#1}{\_printfnotemark}%
                         \_dest[fnf:\the\_gfnotenum]}%
   \_def\_printfnotemarkB{\_link[fnf:\the\_gfnotenum]{\_localcolor#2}{\_printfnotemark}%
                         \_dest[fnt:\the\_gfnotenum]}%
}
\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

```
\_def \_Xfnote {\_incr\_lfnotenum \_incr\_gfnotenum
   \_sxdef{_fn:\_the\_gfnotenum}{\_the\_lfnotenum}}
```

The \fnote {⟨*text*⟩} macro is simple, \fnotemark and \fnotetext does the real work.

fnotes.opm

```
\_def\_fnote{\_fnotemark1\_fnotetext}
\_def\_fnotemark#1{{\_advance\_gfnotenum by#1\_advance\_lfnotenum by#1\relax \_printfnotemarkA}}
```

The \fnotetext calls _opfootnote which is equivalent to plain TEX \vfootnote. It creates new data to Insert \footins. The only difference is that we are able to propagate a macro parameter into Insert group before the text is printed (see section 2.17). This propagated macro is _fnset which sets smaller fonts.

Note that \vfootnote and _opfootnote does't read the text as a parameter but during normal horizontal mode. This is reason why catcode changes (for example in-line verbatim) can be used here.

fnotes.opm

```
\_def\_fnotetext{\_incr\_gfnotenum \_incr\_lfnotenum % global increment
   \_ifpgfnote \_openref \_fi
   \_wref \_Xfnote{}%
   \_ifpgfnote \_ifcsname _fn:\_the\_gfnotenum \_endcsname \_else
       \_opwarning{unknown \_noexpand\fnote mark. TeX me again}
   \_fi\_fi
   \_opfootnote\_fnset\_printfnotemarkB
}
\_def\_fnset{\_everypar={}\_scalemain \_typoscale[800/800]}

\_public \fnote \fnotemark \fnotetext ;
```

By default \mnote{⟨*text*⟩} are in right margin at odd pages and they are in left margin at even pages. The \mnote macro saves its position to .ref file as _Xmnote without parameter. We define _mn:⟨*mnotenum*⟩ as \right or \left when the .ref file is read. The \ifnum 0≤0#2 trick returns true if ⟨*pageno*⟩ has

numeric type and false if it is non-numeric type (Roman numeral, for example). We prefer to use ⟨*pageno*⟩, but only if it has numeric type. We use ⟨*gpageno*⟩ in other cases.

fnotes.opm

```
\_newcount\_mnotenum    \_mnotenum=0       % global counter of mnotes
\_def \_Xmnote {\_incr\_mnotenum \_ea \_XmnoteA \_currpage}
\_def \_XmnoteA #1#2{% #1=<gpageno> #2=<pageno>
   \_sxdef{_mn:\_the\_mnotenum}{\_ifodd\_numtype{#2}{#1} \_right \_else \_left \_fi}}
\_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

```
\_def \_fixmnotes #1{\_edef\_mnotesfixed{\_cs{_\_csstring #1}}}
\_public \fixmnotes ;
```

The outer box of marginal note has zero width and zero depth and it is appended after current line using `\vadjust` primitive or it is inverted to vertical mode as a box with `\vskip-\baselineskip` followed.

fnotes.opm

```
\_long\_def\_mnote#1{\_ifvmode {\_mnoteA{#1}}\_nobreak\_vskip-\_baselineskip \_else
   \_lower\_dp\_strutbox\_hbox{}\_vadjust{\_kern-\_dp\_strutbox \_mnoteA{#1}\_kern\_dp\_strutbox}%
   \_fi
}
\_public \mnote ;
```

The `\_mnoteA` macro does the real work. The `\_lrmnote{`⟨*left*⟩`}{`⟨*right*⟩`}` uses only first or only second parameter depending on the left or right marginal note.

fnotes.opm

```
\_long\_def\_mnoteA #1{\_incr\_mnotenum
   \_ifx\_mnotesfixed\_undefined
      \_ifcsname _mn:\_the\_mnotenum \_endcsname
          \_edef\_mnotesfixed{\_cs{_mn:\_the\_mnotenum}}%
      \_else
          \_opwarning{unknown \_noexpand\mnote side. TeX me again}\_openref
          \_def\_mnotesfixed{\_right}%
   \_fi\_fi
   \_hbox toOpt{\_wref\_Xmnote{}\_everypar={}%
      \_lrmnote{\_kern-\_mnotesize \_kern-\_mnoteindent}{\_kern\_hsize \_kern\_mnoteindent}%
      \_vbox toOpt{\_vss \_setbox0=\_vtop{\_hsize=\_mnotesize
             \_lrmnote{\_leftskip=0pt plus 1fill \_rightskip=0pt}
                      {\_rightskip=0pt plus 1fil \_leftskip=0pt}%
             {\_the\_everymnote\_noindent#1\_endgraf}}%
          \_dp0=0pt \_box0 \_kern\_mnoteskip \_global\_mnoteskip=0pt}\_hss}%
}
\_def \_lrmnote#1#2{\_ea\_ifx\_mnotesfixed\_left #1\_else #2\_fi}
```

We don't want to process `\fnote`, `\fnotemark`, `\mnote` in TOC, headlines nor outlines.

fnotes.opm

```
\_regmacro {\_def\fnote#1{}} {\_def\fnote#1{}} {\_def\fnote#1{}}
\_regmacro {\_def\fnotemark#1{}} {\_def\fnotemark#1{}} {\_def\fnotemark#1{}}
\_regmacro {\_def\mnote#1{}} {\_def\mnote#1{}} {\_def\mnote#1{}}
```

2.34 Styles

OpTEX provides three styles: `\report`, `\letter` and `\slides`. Their behavior is documented in user part of the manual in the section 1.7.2 and `\slides` style (for presentations) is documented in `op-slides.pdf` which is an example of the presentation.

2.34.1 `\report` and `\letter` styles

styles.opm

```
\_codedecl \report {Basic styles of OpTeX <2020-03-28>} % preloaded in format
```

We define auxiliary macro first (used by the `\address` macro)
The `{\boxlines` ⟨*line-1*⟩⟨*eol*⟩⟨*line-2*⟩⟨*eol*⟩ . . . ⟨*line-n*⟩⟨*eol*⟩`}` returns to the outer vertical mode a box with ⟨*line-1*⟩, next box with ⟨*line-2*⟩ etc. Each box has its natural width. This is reason why we cannot use paragraph mode where each resulting box has the width `\hsize`. The ⟨*eol*⟩ is set active and `\everypar` starts `\hbox{` and acive ⟨*eol*⟩ closes this `\hbox` by `}`.

styles.opm

```
\_def\_boxlines{%
   \_def\_boxlinesE{\_ifhmode\_egroup\_empty\_fi}%
   \_def\_nl{\_boxlinesE}%
   \_bgroup \_lccode`\~=`\^^M\_lowercase{\_egroup\_let~}\_boxlinesE
   \_everypar{\_setbox0=\_lastbox\_endgraf
      \_hbox\_bgroup \_catcode`\^^M=13 \_let\par=\_nl \_aftergroup\_boxlinesC}%
}
\_def\_boxlinesC{\_futurelet\_next\_boxlinesD}
\_def\_boxlinesD{\_ifx\_next\_empty\_else\_ea\_egroup\_fi}

\_public \boxlines ;
```

The \report and \letter style initialization macros are defined here.
The \letter defines \address and \subject macros.

styles.opm

```
\_def\_report{
   \_typosize[11/13.2]
   \_vsize=\_dimexpr \_topskip + 52\_baselineskip \_relax % added 2020-03-28
   \_let\_titfont=\_chapfont
   \_titskip=3ex
   \_eoldef\_author##1{\_removelastskip\_bigskip
      {\_leftskip=0pt plus1fill \_rightskip=\_leftskip \_it \_noindent ##1\_par}\_nobreak\_bigskip
   }
   \_public \author ;
   \_parindent=1.2em \_iindent=\_parindent \_ttindent=\_parindent
   \_footline={\_global\_footline={\_hss\_rmfixed\_folio\_hss}}
}
\_def\_letter{
   \_def\_address{\_vtop\_bgroup\_boxlines \_parskip=0pt \_let\par=\_egroup}
   \_def\_subject{{\_bf \_mtext{subj}: }}
   \_public \address \subject ;
   \_typosize[11/14]
   \_vsize=\_dimexpr \_topskip + 49\_baselineskip \_relax % added 2020-03-28
   \_parindent=0pt
   \_parskip=\_medskipamount
   \_nopagenumbers
}
\_public \letter \report ;
```

The \slides macro reads macro file slides.opm, see the section 2.34.2.

styles.opm

```
\_def\_slides{\_par
  \_input slides.opm
}
\_public \slides ;
```

2.34.2 \slides style for presentations

slides.opm

```
\_codedecl \slideshow {Slides style for OpTeX <2020-03-19>} % loaded on demand by \slides
```

Default margins and design is declared here. The _ttfont is scaled by mag1.15 in order to balance the ex height of Helvetica (Heros) and LM fonts Typewriter. The \begtt...\endtt verbatim is printed by smaller text.

slides.opm

```
\_margins/1 a5l (14,14,10,3)mm  % landscape A5 format
\_def\_wideformat{\_margins/1 (263,148) (16,16,10,3)mm } % 16:9 format

\_fontfam[Heros]
\_typosize[16/19]
\_famvardef\_ttfont{\_setfontsize{mag1.15}\_tt}
\_def\_urlfont{}
\_everytt={\_typosize[13/16] \advance\hsize by10mm}
\_fontdef\_fixbf{\_bf}

\_nopagenumbers
\_parindent=0pt
\_ttindent=5mm
\_parskip=5pt plus 4pt minus2pt
```

```
\_rightskip=0pt plus 1fil
\_ttindent=10pt
\_def\_ttskip{\_smallskip}

\_onlyrgb   % RGB color space is better for presentations
```

The bottom margin is set to 3 mm. If we use 1 mm, then baseline of `\footline` is 2 mm from the bottom page. This is depth of the `\Grey` rectangle used for page numbers. It is r-lapped to `\hoffset` width because left margin = `\hoffset` = right margin. It is 14 mm for narrow pages or 16 mm for wide pages.

slides.opm

```
\_footlinedist=1mm
\_footline={\_hss \_rlap{%
   \_rlap{\Grey\_kern.2\_hoffset\_vrule height6mm depth2mm width.8\_hoffset}%
                       \_hbox to\_hoffset{\White\_hss\_folio\_kern3mm}}}
```

The `\subtit` is defined analogically like `\tit`.

slides.opm

```
\_eoldef\_subtit#1{\_vskip20pt {\_leftskip=0pt plus1fill \_rightskip=\_leftskip
   \_subtitfont #1\_nbpar}}
```

The `\pshow`⟨*num*⟩ prints the text in invisible (transparent) font when `\layernum`<⟨*num*⟩. The color font feature is used for this transparency.

slides.opm

```
\_famvardef\Transparent{\_setfontcolor{FF000000}\_currvar}
\_def\_use#1#2{\_ifnum\_layernum#1\_relax#2\_fi}
\_def\_pshow#1{\_use{=#1}\Red \_use{<#1}\Transparent \_ignorespaces}
```

The main level list of items is activated here. The `\_item:X` and `\_item:x` are used and are re-defined here. If we are in nested level of items and `\pg+` is used then `\egroups` macro expands to the right number of `\egroup`s in order to close page correctly. The level of nested item lists is saved to the `\_ilevel` register and used when we start again the next text after `\pg+`.

slides.opm

```
\_newcount\_gilevel
\_def\*{*}
\_adef*{\_startitem}
\_sdef{_item:X}{\Blue\_raise.2ex\_fullrectangle{.8ex}\_kern.5em}
\_sdef{_item:x}{\Blue\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
\_style X
\_def\_egroups{\_par\_global\_gilevel=\_ilevel \_egroup}
\_everylist={\_novspaces \_ifcase\_ilevel \_or \_style x \_else \_style - \_fi
   \_addto\_egroups{\_egroup}}
```

The default values of `\pg`, i. e. `\pg;`, `\pg+` and `\pg.` are very simple. They are used when `\showslides` is not specified.

slides.opm

```
\_def\_pg#1{\_cs{_spg:#1}}
\_sdef{_spg:;}{\_vfil\_break \_lfnotenumreset}
\_sdef{_spg:.}{\_end}
\_sdef{_spg:+}{\_par}
```

We need no numbers and no table of contents when using slides. The `\_printsec` macro is redefined in order the title is centered and typeset in `\Blue`.

slides.opm

```
\_def\_titfont{\_typosize[42/60]\_bf \Blue}
\_def\_subtitfont{\_typosize[20/30]\_bf}
\_def\_secfont{\_typosize[25/30]\_bf \Blue}

\_nonum \_notoc \_let\_resetnonumnotoc=\_relax
\_def\_printsec#1{\_par
   \_abovetitle{\_penalty-400}\_bigskip
   {\_secfont \_noindent \_leftskip=0pt plus1fill \_rightskip=\_leftskip
     \_printrefnum[@\_quad]#1\_nbpar}\_insertmark{#1}%
   \_nobreak \_belowtitle{\_medskip}%
}
```

When `\slideshow` is active then each page is opened by `\setbox\_slidepage=\vbox\bgroup` (roughly speaking) and closed by `\egroup`. The material is `\unvbox`ed 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.

slides.opm

```
\_newbox\_slidepage  \_newbox\_slidepageB
\_countdef\_slidelayer=1
\_def\_decr#1{\_global\_advance#1 by-1 }

\_def\_slideshow{\_slidelayer=1 \_slideshowactive \_setbox\_slidepage=\_vbox\_bgroup}

\_def\_slideshowactive{%
   \_sdef{_spg:;}{\_closepage \_global\_slidelayer=1 \_resetpage \_openslide}
   \_sdef{_spg:.}{\_closepage \_end}
   \_sdef{_spg:+}{\_closepage \_incr\_slidelayer \_decr\_pageno \_openslide}
   \_def\bye     {\_closepage \_end}
   \_let\_layers=\_layersactive
   \_def\_destbox[##1:##2]{\_isequal{##1}{ref}\_iffalse \_destboxori[##1:##2]\_fi}%
}
\_def\_openslide{\_setbox\_slidepage=\_vbox\_bgroup \_setilevel
   \_ifvoid\_slidepage \_else \_unvbox\_slidepage \_nointerlineskip\_lastbox \_fi}
\_def\_setilevel{\_loop \_decr\_gilevel \_ifnum\_gilevel<0 \_else \_begitems \_repeat}

\_def\_closepage{\_egroups
   \_ifnum \_maxlayers=0 \_unvcopy\_slidepage \_vfil\_break
   \_else \_begingroup \_setwarnslides \_layernum=0
      \_loop
         \_ifnum\_layernum<\_maxlayers \_advance\_layernum by1
            \_printlayers \_vfil\_break
            \_ifnum\_layernum<\_maxlayers \_incr\_slidelayer \_decr\_pageno \_fi
      \_repeat
      \_global\_maxlayers=0
      \_incr\_layernum \_global\_setbox\_slidepage=\_vbox{\_printlayers}%
      \_endgroup
   \_fi}
\_def\_resetpage{%
   \_global\_setbox\_slidepage=\_box\_voidbox \_global\_setbox\_slidepageB=\_box\_voidbox
   \_lfnotenumreset
}
\_def\_setwarnslides{%
   \_def\pg##1{\_opwarning{\_string\pg##1 \_layersenv}\_def\pg####1{}}%
   \_def\layers##1 {\_opwarning{\_string\layers\_space \_layersenv}\_def\layers####1{}}%
}
\_def\_layersenv{cannot be inside \_string\layers...\_string\endlayers, ignored}

\_def\_printlayers{\_unvcopy\_slidepage \_nointerlineskip\_lastbox
   \_layertext \_endgraf
   \_ifdim\_prevdepth>-1000pt \_kern-\_prevdepth \_kern\_dp\_strutbox \_fi
   \_vskip\_parskip
   \_unvcopy\_slidepageB
}
\_let\_destboxori=\_destbox

\_newcount\_layernum \_newcount\_maxlayers
\_maxlayers=0

\_long\_def\_layersactive #1 #2\endlayers{%
   \_par\_egroup
   \_gdef\_layertext{#2}%
   \_global\_maxlayers=#1
   \_setbox\_slidepageB=\_vbox\_bgroup
}
\_public \subtit \slideshow \pg \wideformat \use \pshow ;
```

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

```
\_def\_layers #1 {\_par\_layernum=\_numexpr#1+1\_relax}
\_let\endlayers=\_relax

```

```
\_def\layers{\_layers}
```

We must to redefine \fnotenumpages because the data from .ref file are less usable for implementing such feature: the footnote should be in more layers repeatedly. But we can suppose that each page starts by \pg; macro, so we can reset the footnote counter by this macro.

slides.opm

```
\_def \_fnotenumpages {\def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse
   \_def\_lfnotenumreset{\_global\_lfnotenum=0 }}
\_let \_lfnotenumreset=\_relax
\_public \fnotenumpages ;
```

2.35 Logos

logos.opm

```
\_codedecl \TeX {Logos TeX, LuaTeX, etc. <2020-02-28>} % preloaded in format
```

Despite plain TEX each macro for logos ends by \ignoreslash. This macro ignores next slash if it is present. You can `use \TeX/ like this` for protecting the space following the logo. This is visually more comfortable. The macros \TeX, \OpTeX, \LuaTeX, \XeTeX are defined.

logos.opm

```
\_protected\_def \_TeX {T\_kern-.1667em\_lower.5ex\_hbox{E}\_kern-.125emX\_ignoreslash}
\_protected\_def \_OpTeX {Op\kern-.1em\_TeX}
\_protected\_def \_LuaTeX {Lua\_TeX}
\_protected\_def \_XeTeX {X\_kern-.125em\_phantom E%
   \_pdfsave\_rlap{\_pdfscale{-1}{1}\_lower.5ex\_hbox{E}}\_pdfrestore \_kern-.1667em \_TeX}

\_def\_ignoreslash {\_futurelet\_next \_ignoreslashA}
\_def\_ignoreslashA {\_ifx\_next/\_ea\_ignoreit\_fi}

\_public \TeX \OpTeX \LuaTeX \XeTeX \ignoreslash ;
```

The _slantcorr macro expands to slant-correction of current font. It is used to shifting A if the \LaTeX logo is in italic.

logos.opm

```
\_protected\_def \_LaTeX{\_tmpdim=.42ex L\_kern-.36em \_kern \_slantcorr % slant correction
  \_raise \_tmpdim \_hbox{\_thefontscale[710]A}%
  \_kern-.15em \_kern-\_slantcorr \_TeX}
\_def\_slantcorr{\_ea\_ignorept \_the\_fontdimen1\_font\_tmpdim}

\_public \LaTeX ;
```

\OPmac, \CS and \csplain logos.

logos.opm

```
\_def\_OPmac{\_leavevmode
   \_lower.2ex\_hbox{\_thefontscale[1400]O}\_kern-.86em P{\_em mac}\_ignoreslash}
\_def\_CS{$\_cal C$\_kern-.1667em\_lower.5ex\_hbox{$\_cal S$}\_ignoreslash}
\_def\_csplain{\_CS plain\_ignoreslash}

\_public \OPmac \CS \csplain ;
```

The expandable versions of logos used in Outlines needs the expandable \ingnslash (instead of the \ignoreslash).

logos.opm

```
\_def\_ignslash#1{\_ifx/#1\_else #1\_fi}
\_regmacro {}{}{% conversion for PDF outlines
   \_def\TeX{TeX\_ignslash}\_def\OpTeX{OpTeX\_ignslash}%
   \_def\LuaTeX{LuaTeX\_ignslash}\_def\XeTeX{XeTeX\_ignslash}%
   \_def\LaTeX{LaTeX\_ignslash}\_def\OPmac{OPmac\_ignslash}%
   \_def\CS{CS}\_def\csplain{csplain\_ignslash}%
}
\_public \ignslash ;
```

2.36 Multilingual support

2.36.1 Lovercase, uppercase codes

All codes in unicode table keep information about pairs lowecase-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 `\catocde`=12 for all codes above 127.

The file `uni-lcuc.opm` does this work. It is not much intereting file, only first few lines from 15928 lines in total is shown here.

uni-lcuc.opm

```
% Preloaded in format. A copy o uni-lcuc.tex fom csplain is here:

% uni-lcuc.tex -- sets \lccodes and \uccodes for Unicode chars, nothing more
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Petr Olsak,  Jul. 2014

\_wterm{Setting lccodes and uccodes for Unicode characters}

\_def\_tmp #1 #2 {\_ifx^#1^\_else
   \_lccode"#1="#1
   \_ifx.#2%
      \_uccode"#1="#1
   \_else
      \_uccode"#2="#2
      \_lccode"#2="#1
      \_uccode"#1="#2
   \_fi
   \_ea \_tmp \fi
}
\_tmp
00AA .
00B5 039C
00BA .
00E0 00C0
00E1 00C1
00E2 00C2
00E3 00C3
00E4 00C4
```

... etc. (see uni-lcuc.opm)

2.36.2 Hyphenations

hyphen-lan.opm

```
\_codedecl \langlist {Initialization of hypenation patterns <2020-03-10>} % preloaded in format
```

The ⟨*iso-code*⟩ means a shortcut of language name (mostly by ISO 639-1). The following control sequences are used for language switching:

- `\_lan:`⟨*number*⟩ expands to ⟨*iso-code*⟩ of the language. The number is internal number of languages used as a value of `\language` register.
- `\_ulan:`⟨*long-lang*⟩ expands to ⟨*iso-code*⟩ too. This is transformation from long name of language (lowercase letters) to ⟨*iso-code*⟩.
- `\_`⟨*iso-code*⟩`Patt` (for example `\_csPatt`) is the language ⟨*number*⟩ declared by `\chardef`.
- `\`⟨*iso-code*⟩`lang` (for example `\enlang`, `\cslang`, `\sklang`, `\delang`, `\pllang`) is language selector. It exists in two states
  - Initialization state: when `\`⟨*iso-code*⟩`lang` is used first then it must load the patterns into memory using Lua code. If it is done then the `\`⟨*iso-code*⟩`lang` re-defines itself to processing state.
  - Processing state: it only sets `\language=\_`⟨*iso-code*⟩`Patt`, i.e it selects the hyphenation patterns. It does a little more language-dependent work, as mentioned below.
- `\_langspecific:`⟨*isocode*⟩ is processed by `\`⟨*iso-code*⟩`lang` and it should include language-specific macros declared by user or macro designer.

The USenglish patters are preloaded first:

hyphen-lan.opm

```
\_chardef\_enPatt=0
\_def\_pattlist{\_enPatt=0}
\_def\_langlist{en(USenglish)}
\_sdef{_lan:0}{en}
\_sdef{_ulan:usenglish}{en}
\_def\enlang{\_uselang{en}\_enPatt23} % \lefthyph=2 \righthyph=3
\_sdef{_langspecific:en}{\_nonfrenchspacing}

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

\preplang ⟨*iso-code*⟩ ⟨*long-lang*⟩ ⟨*number-cs*⟩ ⟨*number*⟩ ⟨*pre-hyph*⟩⟨*post-hyph*⟩ prepares the \⟨*iso-code*⟩lang to its initialization state. Roughly speaking, it does:

```
\chardef\_⟨iso-code⟩Patt = ⟨number⟩
\def\_lan:⟨number⟩ {⟨iso-code⟩}
\def\_ulan:⟨long-lang⟩ {⟨iso-code⟩}
\def\_⟨iso-code⟩lang {%
   \_loadpattrs ⟨long-lang⟩ ⟨number⟩ % loads patterns using Lua code
   \gdef\_⟨iso-code⟩lang {\_uselang{⟨iso-code⟩}\_⟨iso-code⟩Patt ⟨pre-hyph⟩⟨post-hyph⟩}
   \_⟨iso-code⟩lang                  % runs itself in processing state
}
\def\⟨iso-code⟩lang {\_⟨iso-code⟩lang} % public version \⟨iso-code⟩lang
```

You can see that \⟨*iso-code*⟩lang runs _loadpattrs ⟨*long-lang*⟩ ⟨*iso-code*⟩ in initialization state and _uselang in processing state.

hyphen-lan.opm

```
\_def\_preplang #1 #2 #3#4 #5 {%
   \_chardef#3=#4
   \_sdef{_lan:#4}{#1}\_lowercase{\_sdef{_ulan:#2}}{#1}%
   \_def\_next{\_ea\_noexpand\_csname _#1lang\_endcsname}
   \_ea\_edef \_csname _#1lang\_endcsname {%
      \_lowercase{\_noexpand\_loadpattrs #2} #4 % loads patterns
      \_gdef\_next{\_noexpand\_uselang{#1}#3#5}%  re-defines itself
      \_next                                  %  runs itself in processing state
   }
   \_addto\_langlist{ #1(#2)}%
   \_sdef{#1lang}{\_csname _#1lang\_endcsname}% unprefixed \<isocode>lang
}
\_def\_loadpattrs#1 #2 {%
   \_directlua{
      require("luatex-hyphen")
      luatexhyphen.loadlanguage("#1",#2)
   }%
}
```

uselang{⟨*iso-code*⟩}⟨*iso-code*⟩Patt ⟨*pre-hyph*⟩⟨*post-hyph*⟩
sets \language, \lefthyphenmin, \righthyphenmin and runs \frenchspacing. This default language-dependent settings should be re-declared by _langspecific:⟨*iso-code*⟩ which is run finally (it is \relax by default, only _langspecific:en runs \nonfrenchspacing).

hyphen-lan.opm

```
\_def\_uselang#1#2#3#4{\_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\_relax
   \_frenchspacing  % \nonfrenchspacing can be set in \cs{_langspecific:lan}
   \_cs{_langspecific:#1}%
}
```

The \uselanguage {⟨*long-lang*⟩} is defined here (for compatibility with e-plain users).

hyphen-lan.opm

```
\_def\_uselanguage#1{\_lowercase{\_cs{_\_cs{_ulan:#1}lang}}}
\_public \uselanguage ;
```

The numbers for languages are declared as fixed constants (no auto-generated). This concept is inspired from CSplain. There are typical numbers of languages in CSplain: 5=Czech in IL2, 15=Czech in T1 and 115=Czech in Unicode. We keep these constants but we load only Unicode patterns (greater than 100), of course.

```
\_preplang enus USenglishmax \_enusPatt 100 23
\_preplang engb UKenglish    \_engbPatt 101 23
\_preplang it   Italian      \_itPatt 102 22
\_preplang ia   Interlingua  \_iaPatt 103 22
\_preplang id   Indonesian   \_idPatt 104 22

\_preplang cs   Czech      \_csPatt 115 23
\_preplang sk   Slovak     \_skPatt 116 23
\_preplang de   nGerman    \_dePatt 121 22
\_preplang fr   French     \_frPatt 122 22
\_preplang pl   Polish     \_plPatt 123 22
\_preplang cy   Welsh      \_cyPatt 124 23
\_preplang da   Danish     \_daPatt 125 22
\_preplang es   Spanish    \_esPatt 126 22
\_preplang sl   Slovenian  \_slPatt 128 22
\_preplang fi   Finnish    \_fiPatt 129 22
\_preplang hy   Hungarian  \_huPatt 130 22
\_preplang tr   Turkish    \_trPatt 131 22
\_preplang et   Estoniak   \_etPatt 132 23
\_preplang eu   Basque     \_euPatt 133 22
\_preplang ga   Irish      \_gaPatt 134 23
\_preplang nb   Bokmal     \_nbPatt 135 22
\_preplang nn   Nynorsk    \_nnPatt 136 22
\_preplang nl   Dutch      \_nlPatt 137 22
\_preplang pt   Portuguese \_ptPatt 138 23
\_preplang ro   Romanian   \_roPatt 139 22
\_preplang hr   Croatian   \_hrPatt 140 22
\_preplang zh   Pinyin     \_zhPatt 141 11
\_preplang is   Icelandic  \_isPatt 142 22
\_preplang hsb  Uppersorbian \_hsbPatt 143 22
\_preplang af   Afrikaans  \_afPatt 144 12
\_preplang gl   Galician   \_glPatt 145 22
\_preplang kmr  Kurmanji   \_kmrPatt 146 22
\_preplang tk   Turkmen    \_tkPatt 147 22
\_preplang la   Latin      \_laPatt 148 22
\_preplang lac  classicLatin \_lacPatt 149 22
\_preplang lal  liturgicalLatin \_lalPatt 150 22
\_preplang elm  monoGreek  \_elmPatt 201 11
\_preplang elp  Greek      \_elpPatt 202 11
\_preplang grc  ancientGreek \_grcPatt 203 11
\_preplang ca   Catalan    \_caPatt 204 22
\_preplang cop  Coptic     \_copPatt 205 11
\_preplang mn   Mongolian  \_mnPatt 206 22
\_preplang sa   Sanskrit   \_saPatt 207 13
\_preplang ru   Russian    \_ruPatt 208 22
\_preplang uk   Ukrainian  \_ukPatt 209 22
\_preplang hy   Armenian   \_hyPatt 210 12
\_preplang as   Assamese   \_asPatt 211 11
\_preplang hi   Hindi      \_hiPatt 212 11
\_preplang kn   Kannada    \_knPatt 213 11
\_preplang lv   Latvian    \_lvPatt 215 22
\_preplang lt   Lithuanian \_ltPatt 216 22
\_preplang ml   Malayalam  \_mlPatt 217 11
\_preplang mr   Marathi    \_mrPatt 218 11
\_preplang or   Oriya      \_orPatt 219 11
\_preplang pa   Panjabi    \_paPatt 220 11
\_preplang ta   Tamil      \_taPatt 221 11
\_preplang te   Telugu     \_tePatt 222 11
```

The \langlist includes names of all languages which are ready to load and use their hyphenation patterns. This list is printed to terminal and to log at iniTEX state here. It can be used when processing document too.


```
\message{Language hyph.patterns ready to load: \_langlist.
   Use \string\<shortname>lang to initialize language,
   \string\cslang\space for example}

\_public \langlist ;
```

Maybe, you need to do more language specific actions than only to swich hyphenation patterns. For example you need to load a specific font with a specific script used in selected language, you can define a macros for quotation marks depending on the language etc.

The example shows how to declare such language specific things.

```
\def\langset #1 #2{\sdef{_langspecific:#1}{#2}}

\langset fr {... declare French quotation marks}
\langset de {... declare German quotation marks}
\langset gr {... switch to Greek fonts family}
... etc.
```

Note that you need not to set language specific phrases (like `\today`) by this code. Another concept is used for such tasks. See the section 2.36.3 for more details.

2.36.3 Multilingual phrases and quotation marks

languages.opm

```
\_codedecl \_mtext {Languages <2020-03-15>} % preloaded in format
```

Only four words are generated by OpTEX macros: “Chapter”, “Table”, “Figure” and “Subject”. These phrases can be generated depending on the current value of `\language` register, if you use `\_mtext{`⟨*phrase-id*⟩`}`, specially `\_mtext{chap}`, `\_mtext{t}`, `\_mtext{f}` or `\_mtext{subj}`. If your macros generate more words then you can define such words by `\sdef{_mt:`⟨*phrase-id*⟩`:`⟨*lang*⟩`}` where ⟨*phrase-id*⟩ is a label for declared word and ⟨*lang*⟩ is language shortcut (iso code).

languages.opm

```
\def\_mtext#1{\_trycs{_mt:#1:\_trycs{_lan:\_the\_language}{en}}
   {\_csname _mt:#1:en\_endcsname}}

\sdef{_mt:chap:en}{Chapter} \sdef{_mt:chap:cs}{Kapitola} \sdef{_mt:chap:sk}{Kapitola}
\sdef{_mt:t:en}{Table}      \sdef{_mt:t:cs}{Tabulka}     \sdef{_mt:t:sk}{Tabuľka}
\sdef{_mt:f:en}{Figure}     \sdef{_mt:f:cs}{Obrázek}     \sdef{_mt:f:sk}{Obrázok}
\sdef{_mt:subj:en}{Subject} \sdef{_mt:subj:cs}{Věc}      \sdef{_mt:subj:sk}{Vec}
```

Using `\_langw` ⟨*lang*⟩ ⟨*chapter*⟩ ⟨*table*⟩ ⟨*figure*⟩ ⟨*subject*⟩ you can declare these words more effectively:

languages.opm

```
\_def \_langw #1 #2 #3 #4 #5 {%
   \_sdef{_mt:chap:#1}{#2}\_sdef{_mt:t:#1}{#3}\_sdef{_mt:f:#1}{#4}%
   \_sdef{_mt:subj:#1}{#5}%
}

\_langw en  Chapter      Table       Figure       Subject
%-------------------------------------------------------------
\_langw cs  Kapitola     Tabulka     Obrázek      Věc
\_langw de  Kapitel      Tabelle     Obrázek      Subjekt
\_langw es  Capítulo     Tabla       Figura       Sujeto
\_langw fr  Chaptire     Tableau     Figure       Matière
\_langw it  Capitolo     Tabella     Fig.         Soggetto
\_langw pl  Rozdział     Tabela      Ilustracja   Temat
```

... etc. (see languages.opm)

You can add more words as you wish. For example `\today` macro:

languages.opm

```
\_def \_monthw #1 #2 #3 #4 #5 #6 #7 {%
   \_sdef{_mt:m1:#1}{#2}\_sdef{_mt:m2:#1}{#3}\_sdef{_mt:m3:#1}{#4}%
   \_sdef{_mt:m4:#1}{#5}\_sdef{_mt:m5:#1}{#5}\_sdef{_mt:m6:#1}{#5}%
   \_monthwB #1
}
\_def \_monthwB #1 #2 #3 #4 #5 #6 #7 {%
   \_sdef{_mt:m7:#1}{#2}\_sdef{_mt:m8:#1}{#3}\_sdef{_mt:m9:#1}{#4}%
   \_sdef{_mt:m10:#1}{#5}\_sdef{_mt:m11:#1}{#5}\_sdef{_mt:m12:#1}{#5}%
}

\_monthw en  January February March April May June
             July August September October November December
\_monthw cs  ledna února března dubna května června
             července srpna září října listopadu prosince
\_monthw sk  januára februára marca apríla mája júna
```

```
             júla augusta septembra októbra novembra decembra

\_sdef{_mt:today:en}{\_mtext{m\the\month} \the\day, \the\year}
\_sdef{_mt:today:cs}{\the\day.~\_mtext{m\the\month} \the\year}
\_slet{_mt:today:sk}{_mt:today:cs}

\_def\_today{\_mtext{today}}
\_public \today ;
```

Quotes should be tagged by `\"`⟨*text*⟩`"` and `\'`⟨*text*⟩`'` if `\`⟨*iso-code*⟩`quotes` is declared at beginning of the document (for example `\enquotes`). If not, then the control sequences `\"` and `\'` are undefined. Remember, that they are used in another meaning when `\oldaccents` command is used. The macros `\"` and `\'` are not defined as `\protected` because we need their expansion when `\outlines` are created. User can declare quotes by `\quoteschars`⟨*clqq*⟩⟨*crqq*⟩⟨*clq*⟩⟨*crq*⟩, where ⟨*clqq*⟩...⟨*crqq*⟩ are normal quotes and ⟨*clq*⟩...⟨*crq*⟩ are alternative quotes. or use `\altquotes` to swap between meaning of these two types of quotes.
`\enquotes`, `\csquotes`, `\dequotes`, `\frquotes` etc. are defined here.

languages.opm

```
\_def \_enquotes {\_quoteschars ""‘’}
\_def \_csquotes {\_quoteschars „",‘}
\_def \_frquotes {\_quoteschars ""«»}
\_let \_plquotes = \_frquotes
\_let \_esquotes = \_frquotes
\_let \_grquotes = \_frquotes
\_let \_ruquotes = \_frquotes
\_let \_itquotes = \_frquotes
\_let \_skquotes = \_csquotes
\_let \_dequotes = \_csquotes

\_def \_quoteschars #1#2#3#4{\_def\_altquotes{\_quoteschars#3#4#1#2}\_public\altquotes;%
   \_def \"##1"{#1##1#2}\_def \'##1'{#3##1#4}}
```

Sometimes should be usable to leave the markup `"such"` or `'such'` i.e. without the first backslash. Then you can make the characters `"` and `'` active by the `\activequotes` macro and leave quotes without first backslash. First, declare `\`⟨*iso-code*⟩`quotes`, then `\altquotes` (if needed) and finally `\activequotes`.

languages.opm

```
\def\_activequotes{\_ea\_activequotesA\"""\_ea\_activequotesA\'''}
\def\_activequotesA#1#2#3{\_bgroup\_lccode`\~=`#3\_lowercase{\_egroup\_adef#3##1~{#1##1#2}}}

\_public \quoteschars \activequotes \enquotes \csquotes \skquotes \frquotes \plquotes
   \esquotes \grquotes \ruquotes \itquotes \dequotes ;
```

2.37 Other macros

Miscellaneous macros are here.

others.opm

```
\_codedecl \uv {Miscenaleous <2020-04-02>} % preloaded in format
```

`\useOpTeX` and `\useoptex` are declared as `\relax`.

others.opm

```
\_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

```
\_def\_totalpages {\_openref\_ea\_lastpageA\_currpage}
\_def\_lastpage   {\_openref\_ea\_lastpageB\_currpage}
\_def\_lastpageA #1#2{#1}
\_def\_lastpageB #1#2{#2}
\_def\_currpage {{0}{?}}
\_public \lastpage \totalpages ;
```

We need `\uv`, `\clqq`, `\crqq`, `\flqq`, `\frqq`, `\uslang`, `\ehyph` `\chyph`, `\shyph`, for backward compatibility with $\mathcal{CS}$plain. Codes are set according to Unicode, because we are using Czech only in Unicode when LuaTEX is used.

others.opm

```

```

```
% for compatibility with csplain:

\_chardef\clqq=8222  \_chardef\crqq=8220
\_chardef\flqq=171   \_chardef\frqq=187
\_chardef\promile=8240

\_def\uv#1{\clqq#1\crqq}

\_let\uslang=\enlang  \_let\ehyph=\enlang
\_let\chyph=\cslang   \_let\shyph=\sklang
\_let\csUnicode=\csPatt \_let\czUnicode=\csPatt \_let\skUnicode=\skPatt
```

The \letfont was used in $\mathcal{C}\mathcal{S}$plain instead of \fontlet.

others.opm

```
\_let \letfont = \_fontlet
```

Non breaking space in Unicode.

others.opm

```
\let ^^a0=~
```

TikZ needs these control sequences.

others.opm

```
\ea\toksdef \csname toks@\endcsname=0
\ea\let     \csname voidb@x\endcsname=\_voidbox
```

We don't want to read opmac.tex unless \input opmac is specified.

others.opm

```
\def\OPmacversion{OpTeX}
```

Lorem ipsum can be printed by \lipsum[⟨*range*⟩] or \lorem[⟨*range*⟩], for example \lipsum[3] or \lipsum[112-121], max=150. The data are read from LaTeX file lipsum.ltd.tex.

others.opm

```
\_def \_lipsum {%
   {\_long\_def\ProvidesFile##1[##2]##3{\_ifx\_par##3\_relax\_else \_ea##3\_fi}\_tmpnum=0
    \_def\NewLipsumPar{\_advance\_tmpnum by1
       \_afterassignment\_negativermnm \_sxdef{lips:\_the\_tmpnum}}%
    \_opinput {lipsum.ltd.tex}%
    \_global\_let \_lipsum=\_reallipsum
   }\_lipsum
}
\_def\_negativermnm{\_romannumeral-`\.}
\_def\_reallipsum[#1]{\_lipsumA #1\_empty-\_empty\_end}
\_def\_lipsumA #1-#2\_empty#3\_end{\_tmpnum=#1 \_edef\_tmp{\_ifx^#2^#1\_else#2\_fi}%
   \_loop \_csname lips:\_the\_tmpnum\_endcsname \par % \par is better here
          \_ifnum\_tmpnum<\_tmp \_advance\_tmpnum by1 \_repeat
}
\def\lipsum {\_lipsum}
\def\lorem  {\_lipsum}
```

2.38 Printing documentation

The \printdoc ⟨*filename*⟩⟨*space*⟩ and \printdoctail ⟨*filename*⟩⟨*space*⟩ commands are defined after the file doc.opm is load by \input doc.opm.

The \printcoc starts reading of given ⟨*filename*⟩ from the second line. The file is read in *the listing mode*. The \prindoctail starts reading given ⟨*filename*⟩ from the first occurrence of the _encode. The file is read in normal mode (like \input ⟨*filename*⟩).

The *listing mode* prints the lines as listing of a code. This mode is finished when first ␣␣_doc occurs or first _endcode occurs. At least two spaces must precede before such _doc. On the other hand, the _encode must be at the left edge of the line without spaces. If this rule is not met then the listing mode continues.

If the first line or the last line of the listing mode is empty then such lines are not printed. The maximal number of printed lines in the listing mode is \maxlines. Is is set to almost infinity (100000). You can set it to a more sensible value. Such setting is valid only for the first following listing mode.

When the listing mode is finished by _doc then next lines are read in the normal way, but the material between \begtt ... \endtt pair is shifted by three letters left. The reason is that the three spaces of indentation is recommended in the _doc ... _cod pair and this shifting is a compensation of this indentation.

The `\_cod` macro ignores the rest of current line and starts the listing mode again.

When the listing mode is finished by the `\_endcode` then the `\endinput` is applied, the reading of the file opened by `\printdoc` is finished.

You cannot reach the end of the file (without `\_endcode`) in the listing mode.

The listing mode creates all control sequences which are listed in the index as active link to the main documentation point of such control sequence and prints them in blue. Other text is printed in black.

The main documentation point is denoted by `\`\⟨sequence⟩`` in red, for example `\`\foo``. The user documentation point is the first occurrence of `\^`\⟨sequence⟩``, for example `\^`\foo``. There can be more such markups, all of them are hyperlinks to the main documentation point. And main documentation point is hyperlink to the user documentation point, if such point exists. Finally, the `\~`\⟨sequence⟩`` (for example `\~`\foo``) are hyperlinks to the user documentation point.

doc.opm

```
\_codedecl \printdoc {Macros for documentation printing <2020-04-22>}
```

General decalarations.

doc.opm

```
\fontfam[lmfonts]
\hyperlinks \Green \Green
\enlang
\enquotes
```

Maybe, somebody needs `\seccc` or `\secccc`?

doc.opm

```
\eoldef\seccc#1{\medskip \noindent{\bf#1}\par\nobreak\_firstnoindent}
\def\secccc{\medskip\noindent $\bullet$ }
```

`\enddocument` can be redefined.

doc.opm

```
\let\enddocument=\bye
```

Full page of listing causes underfill `\vbox` in output routine. We need to add a small tolerance.

doc.opm

```
\pgbottomskip=0pt plus10pt minus2pt
```

The listing mode is implemented here.

doc.opm

```
\newcount \maxlines   \maxlines=100000

\_eoldef\_cod#1{\_par \_wipeepar
   \_vskip\_parskip \medskip \_ttskip
   \_begingroup
   \_typosize[8/10]
   \_let\_printverbline=\_printcodeline
   \_ttline=\_inputlineno
   \_setverb
   \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
   \_adef{ }{\ }\_adef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
   \_relax \_ttfont
   \_endlinechar=`^^J
   \_def\_tmpb{\_start}%
   \_readverbline
}
\_def\_readverbline #1^^J{%
   \_def\_tmpa{\_empty#1}%
   \_let\_next=\_readverbline
   \_ea\_isinlist\_ea\_tmpa\_ea{\_Doc}\_iftrue \_let\_next=\_processinput \_fi
   \_ea\_isinlist\_ea\_tmpa\_ea{\_Endcode}\_iftrue \_endinput \_let\_next=\_processinput \_fi
   \_ifx\_next\_readverbline \_addto\_tmpb{#1^^J}\_fi
   \_next
}
{\catcode`\ =13 \gdef\_aspace{ }}\def\_asp{\_ea\_noexpand\_aspace}
\_edef\_Doc{\_asp\_asp\_bslash _doc}
\_edef\_Endcode{\noexpand\_empty\_bslash _endcode}
```

The scanner of the control sequences in the listing mode.

doc.opm

```
\def\makecs{\def\tmp{}\futurelet\next\makecsA}
\def\makecsA{\ifcat a\_noexpand\next \_ea\makecsB \else \_ea\makecsF \_fi}
\def\makecsB#1{\addto\tmp{#1}\futurelet\next\makecsA}
```

```
\def\makecsF{\ifx\tmp\empty \csstring\\%
       \else \ifcsname ,\tmp\endcsname \_link[cs:\tmp]{\Blue}{\csstring\\\tmp}%
       \else \let\next=\tmp \_remfirstunderscore\next
             \ifx\next\empty \let\next=\tmp \fi
             \ifcsname ,\next\endcsname \_link[cs:\next]{\Blue}{\csstring\\\tmp}%
       \else \_csstring\\\tmp \fi\fi\fi
}
\def\makecsI{\ifx\tmp\empty \csstring\\\relax
   \else
      \iindex{\tmp}%
      \ifcsname cs:^\tmp\endcsname \else \dest[cs:^\tmp]\sxdef{cs:^\tmp}{}\fi
      \_link[cs:\tmp]{\Blue}{\tt\csstring\\\tmp}%
   \fi
}
\def\tryindex{\futurelet\next\tryindexA}
\def\tryindexA{\if\csstring\\\noexpand\next \_ea\tryindexB \fi}
\def\tryindexB#1{\let\makecsF=\makecsI \makecs}

\_def\_processinput{%
   \_let\_start=\_relax
   \_ea\_replstring\_ea\_tmpb\_ea{\_aspace^^J}{^^J}
   \_addto\_tmpb{\_end}%
   \_isinlist\_tmpb{\_start^^J}\_iftrue \_advance\_ttline by1\_fi
   \_replstring\_tmpb{\_start^^J}{\_start}%
   \_replstring\_tmpb{\_start}{}%
   \_replstring\_tmpb{^^J\_end}{\_end}%
   \_replstring\_tmpb{^^J\_end}{}%
   \_replstring\_tmpb{\_end}{}%
   \_ea\_prepareverbdata\_ea\_tmpb\_ea{\_tmpb^^J}%
   \_replthis{\_csstring\\}{\noexpand\makecs}%
   \_ea\_printverb \_tmpb\_end
   \_par
   \_endgroup \_ttskip
   \_isnextchar\_par{}{\_noindent}%
}
```

The lines in the listing mode have Yellow background.

doc.opm

```
\def\Yellow{\setcmykcolor{0.0 0.0 0.3 0.03}}

\def\_printcodeline#1{\advance \maxlines by-1
   \ifnum \maxlines<0 \ea \endverbprinting \fi
   \penalty \_ttpenalty \kern-4pt
   \noindent\rlap{\Yellow \vrule height8pt depth5pt width\hsize}%
   \printfilename
   \indent \_printverblinenum #1\par}

\def\printfilename{\hbox to0pt{%
   \hskip\hsize\vbox to0pt{\vss\llap{\Brown\docfile}\kern7.5pt}\hss}%
   \let\printfilename=\relax
}
\everytt={\_let\_printverblinenum=\_relax}

\long\def\endverbprinting#1\_end{\fi \global\maxlines=100000
   \noindent\typosize[8/]\dots etc. (see {\tt\Brown\docfile})}
```

`\docfile` is currently documented file.

doc.opm

```
\def\docfile{}
\def\printdoc #1 {\_par \_def\docfile{#1}%
   \everytt={\_ttshift=-15pt \_let\_printverblinenum=\_relax}%
   \_ea\_cod \input #1
   \everytt={\_let\_printverblinenum=\_relax}%
   \def\docfile{}%
}
\def\printdoctail #1 {\bgroup
   \everytt={}\ttline=-1 \_ea\printdoctailA \input #1 \egroup}
```

Index