OpTEX

Format Based on Plain TEX and OPmac[1]

Version 1.08

Petr Olšák, 2020, 2021, 2022

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

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

The main goal of OpTEX is:

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

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

OpTEX offers a markup language for authors of texts (like 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 the user knows TEX basics. They are described in many books. You can see a short document TEX in nutshell too.

---

[1] 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 the command line

```
optex document
```

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

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

A minimal document should be

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

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

A somewhat larger example with common settings should be:

```
\fontfam[Termes]  % selecting Unicode font family Termes (section 1.3.1)
\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 a more complex, but still simple example.

1.2   Page layout

1.2.1   Setting the margins

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

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

Parameters are:

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

---

[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, a3, a3l, b5, letter and user can declare another own format by `\sdef`:

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

The ⟨*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 format's dimensions stay unscaled. Example:

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

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

1.2.2   Concept of the default page

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

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

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

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

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

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

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

1.2.3 Footnotes and marginal notes

The Plain TEX's macro `\footnote` can be used as usual. But a new macro `\fnote{⟨text⟩}` is defined. The footnote mark is added automatically and it is numbered on each chapter from one[2]. The ⟨text⟩ is scaled to 80 %. User can redefine footnote mark or scaling, as shown in the section 2.34.

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

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

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

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

1.3 Fonts

1.3.1 Font families

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

If you write `\fontfam[?]` then all font families registered in 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 on how to register your own font family are appended in the catalog.

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

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

The font modifier without following variant selector does not change the font actually, it only prepares data used by next variant selectors. There is one special variant selector `\currvar` which does not change the selected variant but reloads the font due to (maybe newly specified) font modifier(s).

The context between variants `\rm` ↔ `\it` and `\bf` ↔ `\bi` is kept by the `\em` macro (emphasize text). It switches from current `\rm` to `\it`, from current `\it` to `\rm`, from current `\bf` to `\bi` and from current `\bi` to `\bf`. The italics correction `\/` is inserted automatically, if needed. Example:

```
This is {\em important} text.     % = This is {\it important\/} text.
\it This is {\em important} text. % = This is\/ {\rm important} text.
\bf This is {\em important} text. % = This is {\bi important\/} text.
\bi This is {\em important} text. % = This is\/ {\bf important} text.
```

More about the OpTₑX Font Selection System is written in the technical documentation in the section 2.13. You can mix more font families in your document, you can declare your own variant selectors or modifiers, etc.

1.3.2 Font sizes

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

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

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

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

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

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

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

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

The `\typosize` and `\typoscale` macros initialize the font family by `\rm`. You can re-size only the current font by the command `\thefontsize[`⟨*font-size*⟩`]` or the font can be rescaled by `\thefontscale[`⟨*factor*⟩`]`. These macros don't change math fonts sizes nor baselineskip.

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

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

More information about resizing of fonts is documented in the section 2.12.1.

1.3.3 Typesetting math

See the additional document Typesetting Math with OpTEX for more details about this issue.

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

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

The following math alphabets are available:

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

The last two selectors `\bf` and `\bi` select the sans serif fonts in math regardless of the current text font family. This is a common notation for vectors and matrices. You can re-declare them, see section 2.16.2 where definitions of Unicode math variants of `\bf` and `\bi` selectors are documented.

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

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

1.4 Typical elements of the document

1.4.1 Chapters and sections

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

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

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

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

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

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

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

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

1.4.2 Another numbered objects

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

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

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

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

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

```
\caption/t The dependency of the computer-dependency on the age.
\cskip
\noindent\hfil\table{rl}{
    age    & value \crl\noalign{\smallskip}
     0--1  & unmeasured \cr
     1--6  & observable \cr
     6--12 & significant \cr
    12--20 & extremal \cr
    20--40 & normal \cr
    40--60 & various \cr
    60--$\infty$ & moderate}
```

This example produces:

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

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

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

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

If you wish to make the table or figure as a floating object, you need to use Plain 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
```

The pair `\midinsert`...`\endinsert` prefers to put the enclosed object to the current place. Only if this is unable due to page breaking, it behaves like `\topinsert`...`\endinsert`.

There are five prepared counters `A`, `B`, `C`, `D` and `E`. They are reset in each chapter and section[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 1.4.1.** Let M be... You can add an optional parameter in brackets. For example, `\theorem [(L'Hôpital's rule)] Let $f$, $g$ be...` is printed like **Theorem 1.4.2 (L'Hôpital's rule).** Let f, g be...

---

[3] This feature can be changed, see the section 2.26 in the technical documentation.

1.4.3 References

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

```
\sec[beatle] About Beatles

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

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

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

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

If there are forward referenced objects then users have to run 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 de-
fined with `\numberedpar`) if you want to reference these numbered objects. You can't use
`\theorem`[⟨*label*⟩] because the optional parameter is reserved to another purpose here.

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

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

1.4.4 Hyperlinks, outlines

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

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

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

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

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

```
\def\_tocborder {1 0 0}  % links in table of contents: red frame
\def\_pgborder {0 1 0}   % links to pages: green frame
\def\_citeborder {0 0 1} % links to references: blue frame
```

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

The hyperlinked footnotes can be activated by \fnotelinks ⟨*color-fnt*⟩ ⟨*color-fnf*⟩ where footnote marks in the 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 the \hyperlinks declaration is used then the parameter of \url is treated as an external URL link. An example: \url{http://www.olsak.net} creates http://www.olsak.net. The characters %, \, #, {, and } have to be protected by backslash in the \url argument, the other special characters ~, ^, & can be written as single character[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 characters %, \, #, {, and } must be escaped in the ⟨*url*⟩ parameter.

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

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

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

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

1.4.5 Lists

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

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

---

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

```
\style A % items of type A), B), C), ...
\style x % small rectangle
\style X % big rectangle
```

For example:

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

produces:

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

Another style can be defined by the command `\sdef{_item:⟨style⟩}{⟨text⟩}`. Default item can be set by `\defaultitem={⟨text⟩}`. The list environments can be nested. Each new level of items is indented by next multiple of `\iindent` value which is set to `\parindent` by default. The `\ilevel` register says what level of items is currently processed. Each `\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 is without vertical spaces automatically. More information about the design of lists of items should be found in the section 2.27.

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

1.4.6 Tables

The macro `\table{⟨declaration⟩}{⟨data⟩}` provides similar ⟨declaration⟩ of tables as in 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 result:

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

Apart from `l`, `r`, `c` declarators, you can use the `p{`⟨*size*⟩`}` declarator which declares the column with paragraphs of given width. More precisely, a long text in the table cell is printed as a multiline paragraph with given width. By default, the paragraph is left-right justified. But there are alternatives:

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

You can use (⟨*text*⟩) in the ⟨*declaration*⟩. Then this text is applied in each line of the table. For example `r(\kern10pt)l` adds more 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 from Plain TEX. It marks the end of each row in the table. Moreover OpTEX defines following similar commands:

- `\crl` ... the end of the row with a horizontal line after it.
- `\crll` ... the end of the row with a double horizontal line after it.
- `\crli` ... like `\crl` but the horizontal line doesn't intersect the vertical double lines.
- `\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*⟩`}` immediately after `\cr*` commands but it doesn't interrupt the vertical lines.

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

```
\everytable={}          % code used in \vbox before table processing
\thistable={}           % code used in \vbox, it is removed after using it
\tabiteml={\enspace}    % left material in each column
\tabitemr={\enspace}    % right material in each column
\tabstrut={\strut}      % strut which declares lines distance in the table
\tablinespace=2pt       % additional vert. space before/after horizontal lines
\vvkern=1pt             % space between lines in double vertical line
\hhkern=1pt             % space between lines in double horizontal line
\tabskip=0pt            % space between columns
\tabskipl=0pt \tabskipr=0pt % space before first and after last column
```

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

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

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

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

creates the following result:

Title		
first	second	third
seven	eight	nine

The $\backslash$`vspan`⟨*number*⟩`{`⟨*text*⟩`}` shifts the ⟨*text*⟩ down in order it looks like to be in the center of the ⟨*number*⟩ lines (current line is first). You can use this for creating tables like in the following example:

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

You can use $\backslash$`vspan` with non-integer parameter too if you feel that the result looks better, for example $\backslash$`vspan`2.1`{text}`.

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

The `c`, `l`, `r` and `p` are default "declaration letters" but you can define more such letters by

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

$\backslash$`def\_tabdeclare`⟨*letter*⟩`{`⟨*left*⟩`##`⟨*right*⟩`}`. More about it is in technical documentation in section 2.30.5. See the definition of the $\backslash$`_tabdeclarec` macro, for example.

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

1.4.7 Verbatim

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

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

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

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

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

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

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

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

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

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

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

The in-line verbatim surrounded by a `\verbchar` doesn't work in parameter of macros and macro definitions. (It works in titles declared by `\chap`, `\sec` etc. and in `\fnote`s, because these macros are specially defined in OpTeX). You can use more robust command `\code{⟨text⟩}` in problematic situations, but you have to escape the following characters in the ⟨text⟩: `\`, `#`, `%`, braces (if the braces are unmatched in the ⟨text⟩), and space or `^` (if there are more than one subsequent spaces or `^` in the ⟨text⟩). Examples:

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

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

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

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

The `\ttline` influences the line numbering by the same way as in `\begtt...\endtt` environment. If `\ttline=-1` then real line numbers are printed (this is the default). If `\ttline<-1` then no line numbers are printed.

The `\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 `\verbinput \hisyntax{HTML} (-) file.html` can be used for HTML or XML codes. OpTeX implements C, Python, TeX, HTML and XML syntax highlighting. More languages can be declared, see the section 2.28.2.

If the code is read by `\verbinput` and there are comment lines prefixed by two characters then you can set them by `\commentchars⟨first⟩⟨second⟩`. Such comments are fully interpreted by TeX (i.e. not verbatim). Section 2.28.1 (page 135) says more about this feature.

1.5 Autogenerated lists

1.5.1 Table of contents

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

Typically, we don't want to repeat the name of the section "Table of contents" in the table of contents again. The direct usage of `\chap` or `\sec` isn't recommended here because the table of contents is typically not referenced to itself. You can print the unnumbered and unreferenced title of the section like this:

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

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

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

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

1.5.2 Making the index

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
\begmulti ⟨number of columns⟩
⟨text⟩
\endmulti
```

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

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

Only "pure words" can be propagated to the index by the `\ii` command. It means that there cannot be any macro, TEX primitive, math selector, etc. But there is another possibility to create such a complex index entry. Use "pure equivalent" in the `\ii` parameter and map this equivalent to a real word that is printed in the index. Such mapping is done by `\iis` command. Example:

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

```
\iis chiquadrat {$\chi$-quadrat}
\iis relax {\code{\\relax}}
```

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

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

1.5.3 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 of numbers are generated depending on the used bib-style. For example, the citations look like [Now08] or [Nowak, 2008].

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

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

You can define alternative 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 an old program that 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:
\nocite[*] \usebib/s (simple) op-biblist  % prints all from op-biblist.bib
```

The ⟨*bib-base*⟩ is one or more `*.bib` database source files (separated by spaces and without extension) and the ⟨*style*⟩ is the part of the filename `bib-`⟨*style*⟩`.opm` where the formatting of

the references list is defined. OpTeX supports `simple` or `iso690` styles. The features of the `iso690` style is documented in the section 2.32.5 in detail. The \usebib command is more documented in section 2.32.2.

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

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

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

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

1.6 Graphics

1.6.1 Colors, transparency

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

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

The color selectors work locally in groups like font selectors.

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

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

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

---

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

```
\colordef \myCyan {.3\Green + .5\Blue}   % 30 % green, 50 % blue, 20% white
\colordef \DarkBlue {\Blue + .4\Black}   % Blue mixed with 40 % of black
\colordef \myGreen{\Cyan+\Yellow}        % exact the same as \Green
\colordef \MyColor {.3\Orange+.5\Green+.2\Yellow}
```

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

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

The `\coloron` macro can be defined as follows:

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

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

1.6.2 Images

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

The `\picwidth` is an equivalent register to `\picw`. Moreover, there is an `\picheight` register 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 TEXto print labels in the same font as document font. OpTEX supports this feature by `\inkinspic` {⟨*filename*⟩.pdf} command. It reads and displays both: PDF image and labels generated by Inkscape.

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

1.6.3 PDF transformations

All typesetting elements are transformed by linear transformation given by the current transformation matrix. The `\pdfsetmatrix` {⟨*a*⟩ ⟨*b*⟩ ⟨*c*⟩ ⟨*d*⟩} command makes the internal

---

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

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

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

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

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

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

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

This gives the following result. First: second: third: *text3*
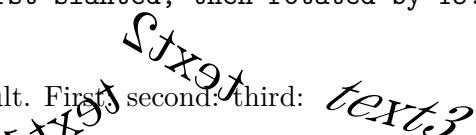
You can see that TEX knows nothing about dimensions of transformed material, it treats it as with a zero dimension object. The `\transformbox{`⟨*transformation*⟩`}{`⟨*text*⟩`}` macro solves the problem. This macro puts the transformed material into a box with relevant dimensions. The ⟨*transfromation*⟩ parameter includes one or more transformation commands `\pdfsetmatrix`, `\pdfscale`, `\pdfrotate` with their parameters. The ⟨*text*⟩ is transformed text.

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

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

1.6.4 Ovals, circles

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

```
\ovalparams={\roundness=2pt        % diameter of circles in the corners
            \fcolor=\Yellow        % color used for filling oval
            \lcolor=\Red           % line color used in the border
            \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 the current point of typesetting and does not change the position of the current point. Assume a coordinate system with origin in the current point. Then `\puttext` ⟨*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-file*⟩`}` puts an image given by ⟨*image-file*⟩ (including extension) 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 specified.

1.7 Others

1.7.1 Using more languages

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

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

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 𝒞𝒮plain, 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 other languages or you want to auto-generate more words in your macros, then you can declare it by `\sdef` or `\_langw` commands as shown in section 2.37.2.

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

If you declare `\`⟨*iso-code*⟩`quotes`, then the control sequences `\"` and `\'` should be used like this: `\"`⟨*quoted text*⟩`"` or `\'`⟨*quoted text*⟩`'` (note that the terminating character is the same but it isn't escaped). This prints language-dependent normal or alternative quotes around ⟨*quoted text*⟩. The language is specified by ⟨*iso-code*⟩. OpTeX declares quotes only for Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English (`\csquotes`, `\dequotes`, ..., `\enquotes`). You can simply define your own quotes as shown in section 2.37.2. The `\"` is used for quotes visually more similar to the `"` character which can be primary quotes or secondary quotes depending on the language rules. Maybe you want to alternate the meaning of these two types of quotes. Use `\`⟨*isocode*⟩`quotes\altquotes` in such case.

1.7.2 Pre-defined styles

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

The `\report` declaration is intended to create reports. It sets default font size to 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 the next pages are numbered (from number 2). Footnotes are numbered from one in the whole document. The macro `\author` ⟨*authors*⟩⟨*end-line*⟩ can be used when `\report` is declared. It prints ⟨*authors*⟩ in italics at the center of the line. You can separate authors by `\nl` to more lines.

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

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

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

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

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

1.7.3 Loading other macro packages

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

```
\load [qrcode, scanbase]
```

does `\input qrcode.opm` and and `\input scanbase.tex`. It saves local information about the fact that these file names (`qrcode`, `scanbase`) were loaded, i.e. next `\load` will skip them.

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

The macro packages intended to OpTeX have the name `*.opm`. The list of packages supported by OpTeX follows. Most of them are directly part of OpTeX:

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

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

1.7.4 Lorem ipsum dolor sit

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

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

1.7.5 Logos

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

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

1.7.6 The last page

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

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

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

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

1.7.7   Use OpTeX

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

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

1.8   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 a list of the items
\enditems        % end of list of the items
\begblock        % start a block of text
\endblock        % end of block of text
\begtt           % start a verbatim text
\endtt           % end verbatim text
\verbchar 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}

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

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

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

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

\magscale[factor]  % resize typesetting, line/page breaking unchanged
\margins/pg format (left, right, top, bottom)unit % margins setting
\report \letter \slides  % style declaration macros
```

1.9 API for macro writers

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

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

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

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

`\addto` `\macro{`⟨*text*⟩`}` adds ⟨*text*⟩ at the end of `\macro` body.
`\adef` ⟨*char*⟩`{`⟨*body*⟩`}` defines ⟨*char*⟩ active character with meaning ⟨*body*⟩.
`\afterfi` `{`⟨*text*⟩`}`⟨*ignored*⟩`\fi` expands to `\fi`⟨*text*⟩.
`\bp` `{`⟨*dimen expression*⟩`}` expands TeX dimension to decimal number in `bp` without unit.
`\_codedecl` ⟨*sequence*⟩ `{`⟨*info*⟩`}` is used at beginning of macro files.
`\colordef` `\macro` `{`⟨*mix of colors*⟩`}` declares `\macro` as color switch.
`\cs` `{`⟨*string*⟩`}` expands `\`⟨*string*⟩.
`\_doc` ... `\_cod` encloses documenation text in the macro code.
`\eoldef` `\macro` `#1{`⟨*body*⟩`}` defines `\macro` with parameter separated to end of line.
`\_endcode` closes the part of macro code in macro files.
`\_endnamespace` closes name space declared by `\_namespace`.
`\eqbox` `[`⟨*label*⟩`]{`⟨*text*⟩`}` creates `\hbox{`⟨*text*⟩`}` with common width across whole document.
`\expr` `{`⟨*expression*⟩`}` expands to result of the ⟨*expression*⟩ with decimal numbers.
`\fontdef` `\f` `{`⟨*font spec.*⟩`}` declares `\f` as font switch.
`\fontlet` `\fa=\fb` ⟨*sizespec.*⟩ declares `\fa` as the same font switch like `\fb` at given ⟨*sizespec.*⟩.
`\foreach` ⟨*list*⟩`\do` ⟨*parameters*⟩`{`⟨*what*⟩`}` is exapandable loop over ⟨*list*⟩.
`\foreachdef` `\macro` ⟨*parameters*⟩`{`⟨*what*⟩`}` declares expandable `\macro` as loop over ⟨*list*⟩.
`\fornum` ⟨*from*⟩`..`⟨*to*⟩`\do` `{`⟨*what*⟩`}` is expanadable loop with numeric variable.
`\incr` ⟨*counter*⟩ increases and `\decr` ⟨*counter*⟩ decreases ⟨*counter*⟩ by one globally.
`\ignoreit` ⟨*one*⟩, `\ignoresecond` ⟨*one*⟩⟨*two*⟩ ignores given parameter.
`\expandafter` `\ignorept` `\the`⟨*dimen*⟩ expands to decimal number ⟨*dimen*⟩ without `pt`.
`\isempty`, `\istoksempty`, `\isequal`, `\ismacro`, `\isdefined`, `\isinlist` `\isfile`, `\isfont` do various tests. Example: `\isinlist\list{`⟨*text*⟩`}\iftrue` does `\iftrue` if ⟨*text*⟩ is in `\list`.
`\isnextchar` ⟨*char*⟩`{`⟨*text1*⟩`}{`⟨*text2*⟩`}` performs ⟨*text1*⟩ if next character is ⟨*char*⟩, else ⟨*text2*⟩.
`\kv` `{`⟨*key*⟩`}` expands to value when key-value parameters are used.
`\loop` ... `\repeat` is classical Plain TeX loop.
`\mathstyles` `{`⟨*math list*⟩`}` enables to create macros dependent on current math style.
`\_namespace` `{`⟨*pkg*⟩`}` declares name space used by package writers.
`\newcount`, `\newdimen` etc. are classical Plain TeX allocators.
`\newif` `\iffoo` declares boolean `\iffoo` as in Plain TeX.
`\_newifi` `\_iffoo` declares boolean `\_iffoo`.
`\opinput` `{`⟨*filename*⟩`}` reads file like `\input` but with standard catcodes.
`\optdef` `\macro` `[`⟨*opt-default*⟩`]` ⟨*parameters*⟩`{`⟨*body*⟩`}` defines `\macro` with [opt.parameter].
`\opwarning` `{`⟨*text*⟩`}` prints ⟨*text*⟩ to the terminal and .log file as warning.
`\private` ⟨*sequence*⟩ ⟨*sequence*⟩ ... ; declares ⟨*sequence*⟩s for private name space.
`\public` ⟨*sequence*⟩ ⟨*sequence*⟩ ... ; declares ⟨*sequence*⟩s for public name space.

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

`\replstring` `\macro{⟨stringA⟩}{⟨stringB⟩}` replaces all ⟨stringA⟩ to ⟨stringB⟩ in `\macro`.

`\sdef` `{⟨string⟩}⟨parameters⟩{⟨body⟩}` behaves like `\def\⟨string⟩⟨parameters⟩{⟨body⟩}`.

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

`\slet` `{⟨stringA⟩}{⟨stringB⟩}` behaves like `\let\⟨stringA⟩=\⟨stringB⟩`

`\sxdef` `{⟨string⟩}⟨parameters⟩{⟨body⟩}` behaves like `\xdef\⟨string⟩⟨parameters⟩{⟨body⟩}`.

`\trycs` `{⟨string⟩}{⟨text⟩}` expands \⟨string⟩ if it is defined else expands ⟨text⟩.

`\useit` ⟨one⟩, `\usesecond` ⟨one⟩⟨two⟩ uses given parameter.

`\wlog` `{⟨text⟩}` writes ⟨text⟩ to .log file.

`\wterm` `{⟨text⟩}` writes ⟨text⟩ to the terminal and .log file.

`\xargs` ⟨what⟩ ⟨token⟩ ⟨token⟩ ... ; repeats ⟨what⟩⟨token⟩ for each ⟨token⟩.

1.10   Compatibility with Plain 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 `\f@@t` are removed and mostly replaced by control sequences prefixed by _ (like `\_this`). Only a basic set of old Plain TeX control sequences like `\p@`, `\z@`, `\dimen@` are provided but not recommended for new macros.

All primitives and common macros have two control sequences with the same meaning: in prefixed and unprefixed form. For example `\hbox` is equal to `\_hbox`. Internal macros of 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 users document too). Users should re-define unprefixed forms of control sequences without worries that something internal will be broken.

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

Plain TeX defines `\newcount`, `\bye` etc. as `\outer` macros. OpTeX doesn't set any macro as `\outer`. Macros like `\TeX`, `\rm` are defined as `\protected`.

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

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

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

The tabbing macros `\settabs` and `\+` (from Plain TeX) are not defined in OpTeX because they are obsolete. But you can use the OpTeX trick 0021 if you really need such feature.

The `\sec` macro is reserved for sections but original Plain TeX declares this control sequence for math secant[9].

1.11   Related documents

- Typesetting math with OpTeX – More details about math typesetting.
- TeX in a Nutshell – Summary about TeX principles, TeX primitive commands etc.
- OpTeX catalog – All fonts collected to `\fontfam` families are shown here.
- OMLS – OpTeX Markup Language Standard.
- OpTeX - tips, tricks, howto – Tips of macro codes for various purposes.

---

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

[9] Use `$\secant(x)$` to get sec(x).

Chapter 2

Technical documentation

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

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

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

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

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

A knowledge about 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 an active link that points (usually) to the main documentation point. The main documentation point can be an active link that points to a previous text where the control sequence was mentioned. Such occurrences are active links to the main documentation point.

2.1   The main initialization file

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

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

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

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

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

```
22  % OpTeX version
23
24  \def\optexversion{1.08 Aug 2022}
25  \def\fmtname{OpTeX}
26  \let\fmtversion=\optexversion
```

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

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

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

```
39  % Basic .opm syntax:
40
41  \let\_endcode =\endinput
42  \def \_codedecl #1#2{\immediate\write-1{#2}}%  information about .opm file
43  \long\def\_doc#1\_cod#2 {} % skip documentation
```

Individual *.opm macro files are read.

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

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

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

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

```
96   \_everyjob = {%
97       \_message{This is OpTeX (Olsak's Plain TeX), version <\optexversion>^^J}%
98       \_directlua{lua.bytecode[1]()}% load OpTeX's Lua code
99       \_mathsbon % replaces \int_a^b to \int _a^b
100      \_inputref % inputs \jobname.ref if exists
101  }
102
103  \dump % You can redefine \dump if additional macros are needed. Example:
104        % \let\dump=\relax \input optex.ini \input mymacros \_dump
```

2.2 Concept of namespaces of control sequences

2.2.1 Prefixing internal control sequences

All control sequences used in OpTeX are used and defined with `_` prefix. The user can be sure that when he/she does `\def\foo` then neither internal macros of OpTeX nor TeX primitives will be damaged. For example `\def\if{...}` will not damage macros because OpTeX's macros are using `\_if` instead of `\if`.

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

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

We never change catcode of `_`, so internal macros can be redefined by user without problems if it is desired. We don't need something like `\makeatletter` from LaTeX.

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

  `\public` ⟨*list of control sequences*⟩ ;

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

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

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

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

2.2.2 Namespace of control sequences for users

Users can (re)define or (re)declare any control sequence with a name without any `_`. This does not make any problem in internal OpTeX macros.[1]

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

- Control sequences which begin with `_` are reserved for TeX primitives, OpTeX internal macros and packages internal macros.
- Multiletter control sequences in the form `\`⟨*word*⟩`_` or `\`⟨*word*⟩`_`⟨*one-letter*⟩, where ⟨*word*⟩ is a sequence of letters, are inaccessible, because they are interpreted as `\`⟨*word*⟩ followed by `_` or as `\`⟨*word*⟩ followed by `_`⟨*one-letter*⟩. This is important for writing math, for example:

---

[1] The token `\par` is in user name space too from OpTeX 1.04+ and LuaTeX 1.14, see also the end of section 2.38.

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

This feature is implemented using Lua code at input processor level, see the section 2.15 for more details. You can deactivate this feature by \mathsboff. After this, you can still write $∫_a^b$ (Unicode) or $\int _a^b$ without problems but \int_a^b yields to undefined control sequence \int_a. You can activate this feature again by \mathsbon. The effect will take shape from next line read from input file.

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

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

2.2.3 Macro files syntax

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

The code in macro files starts by _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 protects from reading the file twice. We suppose, that \sequence is defined in the macro file.

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

2.2.4 Name spaces for package writers

Package writer should use internal names in the form _⟨pgk⟩_⟨sequence⟩, where ⟨pkg⟩ is a package label. For example: _qr_utfstring from qrcode.opm package.

The package writer does not need to write repeatedly _pkg_foo _pkg_bar etc. again and again in the macro file.[2] When the _namespace {⟨pkg⟩} is declared at the beginning of the macro file then all occurrences of \.foo will be replaced by _⟨pkg⟩_foo at the input processor level. The macro writer can write (and backward can read his/her code) simply with \.foo, \.bar control sequences and _⟨pkg⟩_foo, _⟨pkg⟩_bar control sequences are processed internally. The scope of the _namespace command ends at the _endnamespace command or when another _namespace is used. This command checks if the same package label is not declared by the _namespace twice.

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

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

If the package writer needs to declare a control sequence by \newif, then there is an exception of the rule described above. Use _newifi_if⟨pkg⟩_bar, for example _newifi_ifqr_incorner. Then the control sequences _qr_incornertrue and _qr_incornerfalse can be used (or the sequences \.incornertrue and \.incornerfalse when _namespace{qr} is used).

2.2.5 Summary about rules for external macro files published for OpTeX

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

---

[2] We have not adopted the idea from expl3 language:)

- Don't use control sequences from the user namespace in the macro bodies if there is no explicit and documented reason to do this.
- Don't declare control sequences in the user namespace if there are no explicit and documented reasons to do this.
- Use control sequences from OpTEX and primitive namespace in read-only mode, if there is not an explicit and documented reason to redefine them.
- Use _⟨*pkg*⟩_⟨*name*⟩ for your internal macros or \.⟨*name*⟩ if the _namespace{⟨*pkg*⟩} is declared. See section 2.2.4.
- Use \load (or better: _load) for loading more external macros if you need them. Don't use _input explicitly in such cases. The reason is: the external macro file is not loaded twice if another macro or the user needs it explicitly too.
- Use _codedecl as your first command in the macro file and _endcode to close the text of macros.
- Use _doc ... _cod pairs for documenting the code pieces.
- You can write more documentation after the _endcode command.
- The OpTEX catcodes are set when \load your package (i.e. plain TEX catcodes plus catcode of _ is 11). If a catcode is changed during loading your package then it is forgot because \load returns to catcodes used before loading package. If you want to offer a catcode changing for users then insert it to a macro which can be used after loading.

If the macro file accepts these recommendations then it should be named by ⟨*filename*⟩.opm where ⟨*filename*⟩ differs from file names used directly in OpTEX and from other published macros. This extension .opm has precedence before .tex when the \load macro is used.

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

2.2.6   The implementation of the namespaces

```
 3 \_codedecl \public {Prefixing and code syntax <2021-08-16>} % preloaded in format
```

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

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

\ea is useful shortcut for \expandafter. We recommend to use always the private form of _ea because there is high probability that \ea will be redefined by the user.
\public ⟨*sequence*⟩ ⟨*sequence*⟩ ... ; does \let \⟨*sequence*⟩ = _⟨*sequence*⟩ for all sequences.
\private ⟨*sequence*⟩ ⟨*sequence*⟩ ...; does \let _⟨*sequence*⟩ = \⟨*sequence*⟩ for all sequences.
_checkexists ⟨*where*⟩ ⟨*prefix*⟩⟨*sequence*⟩ prints error if the control sequence propagated to a new name space by \public etc. macros is not declared.
\xargs ⟨*what*⟩ ⟨*sequence*⟩ ⟨*sequence*⟩ ... ; does ⟨*what*⟩⟨*sequence*⟩ for each sequences.

```
38 \_let\_ea =\_expandafter  % usefull shortcut
39
40 \_long\_def \_xargs #1#2{\_ifx #2;\_else \_ea#1\_ea#2\_ea\_xargs \_ea #1\_fi}
41
42 \_def \_pkglabel{}
43 \_def \_public {\_xargs \_publicA}
44 \_def \_publicA #1{%
45     \_checkexists \public _#1%
46     \_ea\_let \_ea#1\_csname  _\_csstring #1\_endcsname
47 }
48 \_def \_private {\_xargs \_privateA}
49 \_def \_privateA #1{%
50     \_checkexists \private {}#1%
51     \_ea\_let \_csname  _\_csstring #1\_endcsname =#1%
52 }
53 \_def\_checkexists #1#2#3{\_unless \_ifcsname #2\_csstring#3\_endcsname
```

```
54     \_errmessage {\_string#1: \_bslash#2\_csstring#3 must be declared}\_fi
55 }
56 \_public \public \private \xargs \ea ;
```

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

The _endcode is defined as \endinput in the optex.ini file. \wterm {⟨*text*⟩} prints the ⟨*text*⟩ to the terminal and to the .log file, \wlog {⟨*text*⟩} prints the ⟨*text*⟩ only to the .log file (as in plain TEX)

prefixed.opm
```
68 \_def \_codedecl #1#2{%
69     \_ifx #1\_undefined \_wlog{#2}%
70     \_else \_ea \_endinput \_fi
71 }
72 \_def \_wterm {\_immediate \_write16 }
73 \_def \_wlog {\_immediate\_write-1 } % write on log file (only)
74
75 \_public \wterm \wlog ;
```

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

prefixed.opm
```
82 \_private \optexversion \fmtname ;
```

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

prefixed.opm
```
92 \_def \_pkglabel{}
93 \_def\_namespace #1{%
94     \_ifcsname namesp:#1\_endcsname \_errmessage
95         {The name space "#1" is used already, it cannot be used twice}%
96         \_endinput
97     \_else \_resetnamespace{#1}\_fi
98 }
99 \_def\_resetnamespace #1{%
100     \_ea \_gdef \_csname namesp:#1\_endcsname {}%
101     \_gdef \_pkglabel{_#1}%
102     \_directlua{
103         callback.add_to_callback("process_input_buffer",
104             function (str)
105                 return string.gsub(str, "\_nbb[.]([a-zA-Z])", "\_nbb _#1\_pcent 1")
106             end, "_namespace")
107     }%
108 }
109 \_def\_endnamespace {%
110     \_directlua{ callback.remove_from_callback("process_input_buffer", "_namespace") }%
111     \_gdef \_pkglabel{}%
112 }
113
114 \_def \_nspublic {\_xargs \_nspublicA}
115 \_def \_nspublicA #1{%
116     \_checkexists \nspublic {\_pkglabel _}#1%
117     \_unless\_ifx #1\_undefined
118         \_opwarning{\_ea\_ignoreit\_pkglabel\_space redefines the meaning of \_string#1}\_fi
119     \_ea\_let \_ea#1\_csname \_pkglabel _\_csstring #1\_endcsname
120 }
121 \_def \_nsprivate {\_xargs \_nsprivateA}
122 \_def \_nsprivateA #1{%
123     \_checkexists \nsprivate {}#1%
124     \_ea\_let \_csname \_pkglabel _\_csstring #1\_endcsname =#1%
125 }
```

2.3 pdfTEX initialization

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

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

   4

   5  \_let\_pdfpagewidth        \pagewidth
   6  \_let\_pdfpageheight       \pageheight
   7  \_let\_pdfadjustspacing    \adjustspacing
   8  \_let\_pdfprotrudechars    \protrudechars
   9  \_let\_pdfnoligatures      \ignoreligaturesinfont
  10  \_let\_pdffontexpand       \expandglyphsinfont
  11  \_let\_pdfcopyfont         \copyfont
  12  \_let\_pdfxform            \saveboxresource
  13  \_let\_pdflastxform        \lastsavedboxresourceindex
  14  \_let\_pdfrefxform         \useboxresource
  15  \_let\_pdfximage           \saveimageresource
  16  \_let\_pdflastximage       \lastsavedimageresourceindex
  17  \_let\_pdflastximagepages  \lastsavedimageresourcepages
  18  \_let\_pdfrefximage        \useimageresource
  19  \_let\_pdfsavepos          \savepos
  20  \_let\_pdflastxpos         \lastxpos
  21  \_let\_pdflastypos         \lastypos
  22  \_let\_pdfoutput           \outputmode
  23  \_let\_pdfdraftmode        \draftmode
  24  \_let\_pdfpxdimen          \pxdimen
  25  \_let\_pdfinsertht         \insertht
  26  \_let\_pdfnormaldeviate    \normaldeviate
  27  \_let\_pdfuniformdeviate   \uniformdeviate
  28  \_let\_pdfsetrandomseed    \setrandomseed
  29  \_let\_pdfrandomseed       \randomseed
  30  \_let\_pdfprimitive        \primitive
  31  \_let\_ifpdfprimitive      \ifprimitive
  32  \_let\_ifpdfabsnum         \ifabsnum
  33  \_let\_ifpdfabsdim         \ifabsdim

  34

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

  42

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

  44

  45  \_protected\_def \_pdftexversion     {\_numexpr 140\_relax}
  46          \_def \_pdftexrevision    {7}
  47  \_protected\_def \_pdflastlink      {\_numexpr\_pdffeedback lastlink\_relax}
  48  \_protected\_def \_pdfretval        {\_numexpr\_pdffeedback retval\_relax}
  49  \_protected\_def \_pdflastobj       {\_numexpr\_pdffeedback lastobj\_relax}
  50  \_protected\_def \_pdflastannot     {\_numexpr\_pdffeedback lastannot\_relax}
  51          \_def \_pdfxformname      {\_pdffeedback xformname}
  52          \_def \_pdfcreationdate   {\_pdffeedback creationdate}
  53          \_def \_pdffontname       {\_pdffeedback fontname}
  54          \_def \_pdffontobjnum     {\_pdffeedback fontobjnum}
  55          \_def \_pdffontsize       {\_pdffeedback fontsize}
  56          \_def \_pdfpageref        {\_pdffeedback pageref}
  57          \_def \_pdfcolorstackinit {\_pdffeedback colorstackinit}
  58  \_protected\_def \_pdfliteral       {\_pdfextension literal}
  59  \_protected\_def \_pdfcolorstack    {\_pdfextension colorstack}
  60  \_protected\_def \_pdfsetmatrix     {\_pdfextension setmatrix}
  61  \_protected\_def \_pdfsave          {\_pdfextension save\_relax}
  62  \_protected\_def \_pdfrestore       {\_pdfextension restore\_relax}
  63  \_protected\_def \_pdfobj           {\_pdfextension obj }
  64  \_protected\_def \_pdfrefobj        {\_pdfextension refobj }
  65  \_protected\_def \_pdfannot         {\_pdfextension annot }
  66  \_protected\_def \_pdfstartlink     {\_pdfextension startlink }
  67  \_protected\_def \_pdfendlink       {\_pdfextension endlink\_relax}
  68  \_protected\_def \_pdfoutline       {\_pdfextension outline }
  69  \_protected\_def \_pdfdest          {\_pdfextension dest }
  70  \_protected\_def \_pdfthread        {\_pdfextension thread }
  71  \_protected\_def \_pdfstartthread   {\_pdfextension startthread }
```

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

2.4   Basic macros

We define first bundle of basic macros.

```
3  \_codedecl \sdef {Basic macros for OpTeX <2021-07-20>} % preloaded in format
```

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

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

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

```
23 \_long\_def \_ignoreit #1{}
24 \_long\_def \_useit #1{#1}
25 \_long\_def \_ignoresecond #1#2{#1}
26 \_long\_def \_usesecond #1#2{#2}
27 \_public \ignoreit \useit \ignoresecond \usesecond ;
```

\bslash is "normal backslash" with category code 12. \nbb is double backslash and \pcent is normal %. They can be used in Lua codes, for example.

```
36 \_edef \_bslash {\_csstring\\}
37 \_edef \_nbb {\_bslash\_bslash}
38 \_edef \_pcent{\_csstring\%}
39 \_public \bslash \nbb \pcent ;
```

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

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

```
64 \_def \_adef #1{\_catcode`#1=13 \_begingroup \_lccode`\~=`#1\_lowercase{\_endgroup\_def~}}
65 \_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*⟩.

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

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

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

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

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

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

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

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

```
108 \_def\_loggingall{\_tracingcommands=3 \_tracingstats=2 \_tracingpages=1
109    \_tracingoutput=1 \_tracingmacros=3 % \_tracinglostchars=2 is already set
110    \_tracingparagraphs=1 \_tracingrestores=1 \_tracingscantokens=1
111    \_tracingifs=1 \_tracinggroups=1 \_tracingassigns=1 }
112 \_def\_tracingall{\_tracingonline=1 \_loggingall}
113 \_public \loggingall \tracingall ;
```

_byehook is used in the \bye macro. Write a warning if the user did not load a Unicode Font. Write a "rerun" warning if the .ref file was newly created or it was changed (compared to the previous TeX run).

```
122 \_def\_byehook{%
123    \_ifx\_initunifonts\_relax \_relax\_else \_opwarning{Unicode font was not loaded}\_fi
124    \_immediate\_closeout\_reffile
125    \_edef\_tmp{\_mdfive{\_jobname.ref}}%
126    \_ifx\_tmp\_prevrefhash\_else \_opwarning{Try to rerun,
127       \_jobname.ref file was \_ifx\_prevrefhash\_empty created\_else changed\_fi}\_fi
128 }
```

2.5 Allocators for TeX registers

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

Unlike in PlainTeX, the mentioned allocators are not \outer.

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

Inserts are allocated from 254 to 201 using \newinsert.

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

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

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

```
3 \_codedecl \newdimen {Allocators for registers <2022-06-10>} % preloaded in format
```

The limits are set first.

```
 9 \_chardef\_maicount = 65535     % Max Allocation Index for counts registers in LuaTeX
10 \_let\_maidimen  = \_maicount
11 \_let\_maiskip   = \_maicount
12 \_let\_maimuskip = \_maicount
13 \_let\_maibox    = \_maicount
14 \_let\_maitoks   = \_maicount
15 \_chardef\_mairead  = 15
16 \_chardef\_maiwrite = 15
17 \_chardef\_maifam   = 255
18 \_chardef\_mailanguage = 16380 % In fact 16383, but we reserve next numbers for dummy patterns
```

Each allocation macro needs its own counter.

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

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:).

```
43  \_def\_allocator #1#2#3{%
44      \_incr{\_cs{_#2alloc}}%
45      \_ifnum\_cs{_#2alloc}>\_cs{_mai#2}%
46          \_errmessage{No room for a new \_ea\_string\_csname #2\_endcsname}%
47      \_else
48          \_global#3#1=\_cs{_#2alloc}%
49          \_wlog{\_string#1=\_ea\_string\_csname #2\_endcsname\_the\_cs{_#2alloc}}%
50      \_fi
51  }
```

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

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

The \newinsert macro is defined differently than others.

```
78  \_newcount\_insertalloc    \_insertalloc=255
79  \_chardef\_insertmin = 201
80
81  \_def\_newinsert #1{%
82      \_decr\_insertalloc
83      \_ifnum\_insertalloc <\_insertmin
84          \_errmessage {No room for a new \_string\insert}%
85      \_else
86          \_global\_chardef#1=\_insertalloc
87          \_wlog {\_string#1=\_string\_insert\_the\_insertalloc}%
88      \_fi
89  }
90  \_public \newinsert ;
```

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

```
98  \_newcount \_marksalloc \_marksalloc=0 % start at 1, 0 is \mark
99  \_chardef\_maimarks=\_maicount
100 \_def\_newmarks #1{\_allocator #1{marks}\_chardef}
101
102 \_newcount \_attributealloc  \_attributealloc=0
103 \_chardef\_maiattribute=\_numexpr\_maicount -1\_relax
104 \_attributedef\_noattr \_maicount
105 \_def\_newattribute #1{\_allocator #1{attribute}\_attributedef}
```

```
106
107  \_newcount \_catcodetablealloc  \_catcodetablealloc=10
108  \_chardef\_maicatcodetable=32767
109  \_def\_newcatcodetable #1{\_allocator #1{catcodetable}\_chardef}
110
111  \_public \newmarks \newattribute \newcatcodetable ;
```

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

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

A few registers are initialized like in plainTEX. We absolutely don't support the `@`category dance, so `\z@skip \z@`, `\p@` etc. are not defined in OpTEX. If you need such control sequences then you can initialize them by `\load[plain-at]`.

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

```
132  \_newdimen\_maxdimen \_maxdimen=16383.99999pt % the largest legal <dimen>
133  \_newdimen\_zo \_zo=0pt
134  \_newskip\_hideskip \_hideskip=-1000pt plus 1fill % negative but can grow
135  \_newskip\_centering \_centering=0pt plus 1000pt minus 1000pt
136  \_newskip\_zoskip \_zoskip=0pt plus0pt minus0pt
137  \_newbox\_voidbox % permanently void box register
138
139  \_public \maxdimen \hideskip \centering \voidbox ;
```

2.6 If-macros, loops, is-macros

```
3  \_codedecl \newif {Special if-macros, is-macros and loops <2022-05-04>} % preloaded in format
```

2.6.1 Classical \newif

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

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

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

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

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

```
40  \_long\_def \_afterfi#1#2\_fi{\_fi#1}
41  \_long\_def \afterfi#1#2\fi{\_fi#1}
```

2.6.2 Loops

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

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

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

<div style="text-align:right">if-macros.opm</div>

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

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

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

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

Note that `\foreach` ⟨*list*⟩`\do` {⟨*what*⟩} is equivalent to `\foreach` ⟨*list*⟩`\do` `#1{`⟨*what*⟩`}`.

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

<div style="text-align:right">if-macros.opm</div>

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

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

The test in the `\_fornumB` says: if (⟨*to*⟩ < ⟨*current number*⟩ AND ⟨*step*⟩ is positive) or if (⟨*to*⟩ > ⟨*current number*⟩ AND ⟨*step*⟩ is negative) then close loop by `\_getforstack`. Sorry, the condition is writen by somewhat cryptoid TeX language.

<div style="text-align:right">if-macros.opm</div>

```
113  \_def\_fornum#1..#2\_do{\_fornumstep 1:#1..#2\_do}
114  \_long\_def\_fornumstep#1:#2..#3\_do#4{\_putforstack
115      \_immediateassigned{%
116          \_gdef\_fbody##1{#4}%
117          \_global\_frnum=\_numexpr#2\_relax
118      }%
119      \_ea\_fornumB\_ea{\_the\_numexpr#3\_ea}\_ea{\_the\_numexpr#1}%
120  }
121  \_def\_fornumB #1#2{\_ifnum#1\_ifnum#2>0<\_else>\_fi \_frnum \_getforstack
122      \_else \_afterfi{\_ea\_fbody\_ea{\_the\_frnum}%
123          \_immediateassignment\_global\_advance\_frnum by#2
124          \_fornumB{#1}{#2}}\_fi
125  }
126  \_def\fornum#1..#2\do{\_fornumstep 1:#1..#2\_do}
127  \_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
145  \_newcount\_forlevel
146  \_def\_putforstack{\_immediateassigned{%
147     \_ifnum\_forlevel>0
148        \_sxdef{_frnum:\_the\_forlevel\_ea}{\_the\_frnum}%
149        \_global\_slet{_fbody:\_the\_forlevel}{_fbody}%
150     \_fi
151     \_incr\_forlevel
152  }}
153  \_def\_getforstack{\_immediateassigned{%
154     \_decr\_forlevel
155     \_ifnum\_forlevel>0
156        \_global\_slet{_fbody}{_fbody:\_the\_forlevel}%
157        \_global\_frnum=\_cs{_frnum:\_the\_forlevel}\_space
158     \_fi
159  }}
160  \_ifx\_immediateassignment\_undefined % for compatibility with older LuaTeX
161     \_let\_immediateassigned=\_useit \_let\_immediateassignment=\_empty
162  \_fi
```

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

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

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

```
                                                                              if-macros.opm
177  \_def\_foreachdef#1#2#{\_toks0{#2}%
178     \_long\_edef#1##1{\_ea\_noexpand\_csname _body:\_csstring#1\_endcsname
179                       ##1\_the\_toks0 \_noexpand\_finbody}%
180     \_foreachdefA#1{#2}}
181  \_long\_def\_foreachdefA#1#2#3{%
182     \_long\_sdef{_body:\_csstring#1}#2{\_testparam##1..\_iftrue #3\_cs{_body:\_csstring#1\_ea}\_fi}}
183
184  \_public \foreachdef ;
```

2.6.3 Is-macros

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

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

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

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

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

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

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

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

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

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

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

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

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

\isinlist \list{$\langle text\rangle$}\iftrue is true if the $\langle text\rangle$ is included the macro body of the \list. The category codes are relevant here. The macro is expandable.

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

\isfile $\{\langle filename\rangle\}$\iftrue is true if the file $\langle filename\rangle$ exists and are readable by TeX.

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

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

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

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

```
298  \_long\_def\_isnextchar#1#2#3{\_begingroup\_toks0={\_endgroup#2}\_toks1={\_endgroup#3}%
299     \_let\_tmp= #1\_futurelet\_next\_isnextcharA
300  }
301  \_def\_isnextcharA{\_the\_toks\_ifx\_tmp\_next0\_else1\_fi\_space}
302
303  \_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 this control sequence then the appropriate register exists steadily and build-in algorithms are using it without change. But user cannot access its value in this case. OpTeX declares two control sequences for each register: prefixed (private) and unprefixed (public). OpTeX macros use only prefixed variants of control sequences. The user should use the unprefixed variant with the same meaning and set or read the values of registers using the unprefixed variant. If the user re-defines the unprefixed control sequence of a register then OpTeX macros still work without change.

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

2.7.1   Primitive registers

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

```
10  \_parindent=20pt      % indentation of paragraphs
11  \_pretolerance=100    % parameters used in paragraph breaking algorithm
12  \_tolerance=200
13  \_hbadness=1000
14  \_vbadness=1000
15  \_doublehyphendemerits=10000
16  \_finalhyphendemerits=5000
17  \_adjdemerits=10000
18  \_uchyph=1
19  \_defaulthyphenchar=`\-
20  \_defaultskewchar=-1
21  \_hfuzz=0.1pt
22  \_vfuzz=0.1pt
23  \_overfullrule=5pt
24  \_linepenalty=10      % penalty between lines inside the paragraph
25  \_hyphenpenalty=50    % when a word is bro-ken
26  \_exhyphenpenalty=50 % when the hyphenmark is used explicitly
27  \_binoppenalty=700    % between binary operators in math
28  \_relpenalty=500      % between relations in math
29  \_brokenpenalty=100  % after lines if they end by a broken word.
30  \_displaywidowpenalty=50   % before last line of paragraph if display math follows
31  \_predisplaypenalty=10000  % above display math
32  \_postdisplaypenalty=0     % below display math
33  \_delimiterfactor=901 % parameter for scaling delimiters
34  \_delimitershortfall=5pt
35  \_nulldelimiterspace=1.2pt
36  %\_scriptspace=0.5pt % \Umathspaceafterscript used in \_setmathdimens, \_setunimathdimens instead
37  \_maxdepth=4pt
38  \_splitmaxdepth=\_maxdimen
39  \_boxmaxdepth=\_maxdimen
40  \_parskip=0pt plus 1pt
41  \_abovedisplayskip=12pt plus 3pt minus 9pt
42  \_abovedisplayshortskip=0pt plus 3pt
43  \_belowdisplayskip=12pt plus 3pt minus 9pt
44  \_belowdisplayshortskip=7pt plus 3pt minus 4pt
45  \_parfillskip=0pt plus 1fil
46  \_thinmuskip=3mu
47  \_medmuskip=4mu plus 2mu minus 4mu
48  \_thickmuskip=5mu plus 5mu
```

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

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

2.7.2 Plain TₑX registers

Allocate registers that are used just like in plain TₑX.
\smallskipamount, \medskipamount, \bigskipamount, \normalbaselineskip, \normallineskip, \normallineskiplimit, \jot, \interdisplaylinepenalty, \interfootnotelinepenalty.

```
67 % We also define special registers that function like parameters:
68 \_newskip\_smallskipamount \_smallskipamount=3pt plus 1pt minus 1pt
69 \_newskip\_medskipamount \_medskipamount=6pt plus 2pt minus 2pt
70 \_newskip\_bigskipamount \_bigskipamount=12pt plus 4pt minus 4pt
71 \_newskip\_normalbaselineskip \_normalbaselineskip=12pt
72 \_newskip\_normallineskip \_normallineskip=1pt
73 \_newdimen\_normallineskiplimit \_normallineskiplimit=0pt
74 \_newdimen\_jot \_jot=3pt
75 \_newcount\_interdisplaylinepenalty \_interdisplaylinepenalty=100
76 \_newcount\_interfootnotelinepenalty \_interfootnotelinepenalty=100
77
78 \_public \smallskipamount \medskipamount \bigskipamount
79   \normalbaselineskip \normallineskip \normallineskiplimit
80   \jot \interdisplaylinepenalty \interfootnotelinepenalty ;
```

Plain TₑX macros for setting parameters. \normalbaselines, \frenchspacing, \nonfrenchspacing.

```
87 \_def\_normalbaselines{\_lineskip=\_normallineskip
88   \_baselineskip=\_normalbaselineskip \_lineskiplimit=\_normallineskiplimit}
89
90 \_def\_frenchspacing{\_sfcode`\.=1000 \_sfcode`\?=1000 \_sfcode`\!=1000
91   \_sfcode`\:=1000 \_sfcode`\;=1000 \_sfcode`\,=1000 }
92 \_def\_nonfrenchspacing{\_sfcode`\.=3000 \_sfcode`\?=3000 \_sfcode`\!=3000
93   \_sfcode`\:=2000 \_sfcode`\;=1500 \_sfcode`\,=1250 }
94
95 \_public \normalbaselines \frenchspacing \nonfrenchspacing ;
```

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

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

The following primitive registers have different values than in plain TₑX. We prohibit orphans, set more information for tracing boxes, set page origin to the upper left corner of the paper (no at 1 in, 1 in coordinates) and set default page dimensions as A4, not letter.

```
118 \_emergencystretch=20pt % we want to use third pass of paragraph building algorithm
119                         % we don't need compatibility with old documents
120
121 \_clubpenalty=10000    % after first line of paragraph
122 \_widowpenalty=10000   % before last line of paragraph
123
124 \_showboxbreadth=150   % for tracing boxes
125 \_showboxdepth=7
126 \_errorcontextlines=15
127 \_tracinglostchars=2   % missing character warnings on terminal too
128
129 \_outputmode=1   % PDF output
130 \_pdfvorigin=0pt % origin is exactly at upper left corner
131 \_pdfhorigin=0pt
132 \_hoffset=25mm   % margins are 2.5cm, no 1in
133 \_voffset=25mm
134 \_hsize=160mm    % 210mm (from A4 size) - 2*25mm (default margins)
135 \_vsize=244mm    % 297mm (from A4 size) - 2*25mm (default margins) -3mm baseline correction
136 \_pagewidth=210 true mm
137 \_pageheight=297 true mm
```

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

46

```
144  \_def\_plaintexsetting{%
145     \_emergencystretch=0pt
146     \_clubpenalty=150
147     \_widowpenalty=150
148     \_pdfvorigin=1in
149     \_pdfhorigin=1in
150     \_hoffset=0pt
151     \_voffset=0pt
152     \_hsize=6.5in
153     \_vsize=8.9in
154     \_pagewidth=8.5 true in
155     \_pageheight=11 true in
156     \_nonfrenchspacing
157  }
158  \_public \plaintexsetting ;
```

2.7.4 OpTeX parameters

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

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

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

```
184  \_newtoks\_picdir
185  \_public \picdir ;
```

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

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

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

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

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

```
228  \_newtoks\_everytt
229  \_newtoks\_everyintt
230  \_public \everytt \everyintt ;
```

47

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

```
244  \_newcount\_ttline    \_ttline=-1  % last line number in \begtt...\endtt
245  \_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 the right. Despite the `\ttindent`, it does not shift the line numbers, only the text.
The `\iindent` gives default indentations used in the table of contents, captions, lists, bib references,
It is strongly recommended to re-set this value if you set `\parindent` to another value than plain TeX default 20pt. A well-typeset document should have the same dimension for all indentations, so you should say `\ttindent=\parindent` and `\iindent=\parindent`.

```
265  \_newdimen\_ttindent \_ttindent=\_parindent % indentation in verbatim
266  \_newdimen\_ttshift
267  \_newdimen\_iindent  \_iindent=\_parindent
268  \_public \ttindent \ttshift \iindent ;
```

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

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

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

```
291  \_newtoks\_hicolors
292  \_public \hicolors ;
```

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

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

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

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

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

```
330  \_newdimen\_colsep \_colsep=20pt  % space between columns
331  \_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.

```
339 \_newtoks \_everytocline
340 \_public \everytocline ;
```

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

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

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

```
354 \_newtoks\_bibtexhook
355 \_newtoks\_biboptions
356 \_newtoks\_bibpart
357 \_public \bibtexhook \biboptions \bibpart ;
```

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

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

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

```
372 \_newtoks\_everyii
373 \_public \everyii ;
```

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

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

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

```
384 \_newtoks\_everymnote
385 \_newdimen\_mnotesize   \_mnotesize=20mm   % the width of the mnote paragraph
386 \_newdimen\_mnoteindent \_mnoteindent=10pt % distance between mnote and text
387 \_public \everymnote \mnotesize \mnoteindent ;
```

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

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

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

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

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

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

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

`\tsize` is virtual unit of the width of paragraph-like table items when `\table pxto`⟨*size*⟩ is used.

```
421 \_newtoks\_everytable \_newtoks\_thistable
422 \_newtoks\_tabiteml \_newtoks\_tabitemr \_newtoks\_tabstrut
423 \_newdimen\_tablinespace \_newdimen\_vvkern \_newdimen\_hhkern \_newdimen\_tsize
424 \_newskip\_tabskipl  \_newskip\_tabskipr
425 \_everytable={}        % code used after settings in \vbox before table processing
426 \_thistable={}         % code used when \vbox starts, is is removed after using it
427 \_tabstrut={\_strut}
428 \_tabiteml={\_enspace} % left material in each column
429 \_tabitemr={\_enspace} % right material in each column
430 \_tablinespace=2pt     % additional vertical space before/after horizontal rules
431 \_vvkern=1pt           % space between double vertical line and used in \frame
432 \_hhkern=1pt           % space between double horizontal line and used in \frame
433 \_tabskipl=0pt\_relax  % \tabskip used before first column
434 \_tabskipr=0pt\_relax  % \tabskip used after the last column
435 \_public \everytable \thistable \tabiteml \tabitemr \tabstrut \tablinespace
436         \vvkern \hhkern \tsize \tabskipl \tabskipr ;
```

49

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

```
445 \_newtoks  \_eqlines  \_eqlines={\_openup\_jot}
446 \_newtoks  \_eqstyle  \_eqstyle={\_strut\_displaystyle}
447 \_newdimen \_eqspace  \_eqspace=20pt
448 \_public \eqlines \eqstyle \eqspace ;
```

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

```
455 \_newtoks \_lmfil   \_lmfil={\_hfil}
456 \_public \lmfil ;
```

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

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

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

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

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

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

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

```
500 \_newskip \_pgbottomskip  \_pgbottomskip=0pt \_relax
501 \_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={\rmfixed \firstmark \hfil}}
```

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

```
515 \_newtoks \_nextpages
516 \_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}
```

```
528 \_newtoks \_pgbackground   \_pgbackground={} % for page background
529 \_public \pgbackground  ;
```

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

50

```
537  \_newtoks \_ovalparams
538  \_newtoks \_circleparams
539  %\_ovalparams={\_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
540  %              \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt }
541  %\_circleparams={\_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
542  %               \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt}
543
544  \_newdimen \_roundness    \_roundness=5mm % used in \clippingoval macro
545
546  \_public \ovalparams \circleparams \roundness ;
```

OpTeX defines "Standard OpTeX markup language"[3] which lists selected commands from chapter 1 and gives their behavior when a converter from OpTeX document to HTML or Markdown or LaTeX is used. The structure-oriented commands are selected here, but the commands which declare typographical appearance (page layout, dimensions, selected font family) are omitted. More information for such a converter should be given in \cnvinfo{⟨*data*⟩}. OpTeX simply ignores this but the converter can read its configuration from here. For example, a user can write:

    \cnvinfo {type=html, ⟨*cnv-to-html-data*⟩}
    \cnvinfo {type=markdown, ⟨*cnv-to-markdown-data*⟩}

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

```
567  \_let\cnvinfo=\_ignoreit
```

2.8   More OpTeX macros

The second bundle of OpTeX macros is here.

```
3  \_codedecl \eoldef {OpTeX useful macros <2022-05-24>} % preloaded in format
```

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

To achieve this, we declare \optexcatcodes catcode table and \plaintexcatcodes. They save the commonly used catcode tables. Note that \catcodetable is a part of LuaTeX extension. The catcodetable stack is implemented by OpTeX macros. The \setctable ⟨*catcode table*⟩ pushes current catcode table to the stack and activates catcodes from the ⟨*catcode table*⟩. The \restorectable returns to the saved catcodes from the catcode table stack.

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

```
29  \_def\_opinput #1{\_setctable\_optexcatcodes
30     \_savecatcodetable\_tmpcatcodes \_catcodetable\_tmpcatcodes
31     \_input {#1}\_relax\_restorectable}
32
33  \_newcatcodetable \_optexcatcodes
34  \_newcatcodetable \_plaintexcatcodes
35  \_newcatcodetable \_tmpcatcodes
36
37  \_public \optexcatcodes \plaintexcatcodes \opinput ;
38
39  \_savecatcodetable\_optexcatcodes
40  {\_catcode`_=8 \savecatcodetable\plaintexcatcodes}
```

The implementation of the catcodetable stack follows.

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

---

[3] Will be developed in 2021.

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

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

```
77  \_def\_normalcatcodes {\_catcodetable\_optexcatcodes \_savecatcodetable0 \_catcodetable0 }
78  \_public \normalcatcodes ;
```

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

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

```
92   \_def \_load [#1]{\_loadA #1,,,\_end}
93   \_def \_loadA #1#2,{\_ifx,#1 \_ea \_loadE \_else \_loadB{#1#2}\_ea\_loadA\_fi}
94   \_def \_loadB #1{%
95      \_ifcsname _load:#1\_endcsname \_else
96         \_isfile {#1.opm}\_iftrue \_opinput {#1.opm}\_else \_opinput {#1}\_fi
97         \_sxdef{_load:#1}{}%
98         \_trycs{_afterload}{}\_let\_afterload=\_undefined
99      \_fi
100  }
101  \_def \_loadE #1\_end{}
102  \_public \load ;
```

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

```
118  \_def\_optdef#1[#2]{%
119     \_def#1{\_opt={#2}\_isnextchar[{\_cs{_oA:\_csstring#1}}{\_cs{_oB:\_csstring#1}}}%
120     \_sdef{_oA:\_csstring#1}[##1]{\_opt={##1}\_cs{_oB:\_csstring#1\_nospaceafter}}%
121     \_sdef{_oB:\_csstring#1\_nospaceafter}%
122  }
123  \_def\_nospaceafter#1{\_ea#1\_romannumeral-`\.}
124  \_newtoks\_opt
125
126  \_public \opt \optdef ;
```

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

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

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

```
144 \_def\_eoldef #1{\_def #1{\_begingroup \_catcode`\^^M=12 \_eoldefA #1}%
145    \_ea\_def\_csname _\_csstring #1:M\_endcsname}
146 \_catcode`\^^M=12 %
147 \_def\_eoldefA #1#2^^M{\_endgroup\_csname _\_csstring #1:M\_endcsname{#2}}%
148 \_normalcatcodes %
149
150 \_eoldef\_skiptoeol#1{}
151 \_def\_bracedparam#1{\_ifcsname _\_csstring #1:M\_endcsname
152    \_csname _\_csstring #1:M\_ea \_endcsname
153    \_else \_csname __in\_csstring #1:M\_ea \_endcsname \_fi
154 }
155 \_public \eoldef \skiptoeol \bracedparam ;
```

\scantoeol\macro ⟨*text to end of line*⟩ scans the ⟨*text to end of line*⟩ in verbatim mode and runs the
\macro{⟨*text to end of line*⟩}. The \macro can be defined \def\macro#1{...\scantextokens{#1}...}.
The new tokenization of the parameter is processed when the parameter is used, no when the parameter
is scanned. This principle is used in definition of **\chap**, **\sec**, **\secc** and **_Xtoc** macros. It means that
user can write \sec text `&` text for example. Inline verbatim works in title sections.

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

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

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

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

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

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

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

```
222 \_def\catcode#1{\_catcode \_if`\_noexpand#1\_ea`\_else\_if"\_noexpand#1"\_else
223    \_if'\_noexpand#1'\_else \_ea\_ea\_ea\_ea\_ea\_ea\_ea#1\_fi\_fi\_fi}
```

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

```
232 \_def\_removespaces #1 {\_isempty{#1}\_iffalse #1\_ea\_removespaces\_fi}
233 \_ea\_def \_ea\_ignorept \_ea#\_ea1\_detokenize{pt}{#1}
234
235 \public \removespaces \ignorept ;
```

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 the decimal point. The ⟨*expression*⟩ can include `+-*/()` and decimal numbers in common syntax.

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

```
254 \_def\_decdigits{3} % digits after decimal point in \_bp and \_expr outputs.
255 \_def\_pttopb{%
256    \_directlua{tex.print(string.format('\_pcent.\_decdigits f',
257             token.scan_dimen()/65781.76))}%  pt to bp conversion
258 }
259 \def\_bp#1{\_ea\_pttopb\_dimexpr#1\_relax}
260 \def\_expr#1{\_directlua{tex.print(string.format('\_pcent.\_decdigits f',#1))}}
261
262 \_public \_expr \bp ;
```

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

```
\_doc ⟨ignored text⟩
⟨documentation⟩
\_cod ⟨ignored text⟩
```

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

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

2.9   Using key=value format in parameters

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

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

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

or simply `\myframe {text3}`. You can define `\myframe` as follows:

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

We recommend using `\optdef` for defining macros with optional parameters written in `[]`. Then the optional parameters are saved in the `\opt` tokens register. First: we read default parameters by `\readkv\myframedefaults` and secondly the actual parameters are read by `\readkv{\the\opt}`. The

last setting wins. Third: the values can be used by the expandable \kv{⟨key⟩} macro. The \kv{⟨key⟩} returns ??? if such key is not declared.

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

```
\optdef\myframe [] #1{...
    \ea\addto\ea\myframedefaults\ea{\the\opt}%
    \replstring\myframedefaults{draft}{DRAFTv=1}%
    \readkv\myframedefaults
    ...
    \ifnum\kv{DRAFTv}=1 draft mode\else normal mode\fi
    ...}
```

```
 3 \_codedecl \readkv {Key-value dictionaries <2020-12-21>} % preloaded in format
```

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

```
15 \_def\_readkv#1{\_ea\_def\_ea\_tmpb\_ea{#1}%
16    \_replstring\_tmpb{= }{=}\_replstring\_tmpb{ =}{=}%
17    \_replstring\_tmpb{, }{,}\_replstring\_tmpb{,,}{,}%
18    \_ea \_kvscan \_tmpb,,=,}
19 \_def\_kvscan #1#2=#3,{\_ifx#1,\_else \_sdef{_kv:#1#2}{#3}\_ea\_kvscan\_fi}
20 \_def\_kv#1{\_trycs{_kv:#1}{\_kvunknown}}
21 \_def\_kvunknown{???}
22
23 \public \readkv \kv ;
```

2.10   Plain TeX macros

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

```
 3 \_codedecl \magstep {Macros from plain TeX <2022-02-19>} % 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\_}
```

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

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

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

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

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

Plain TeX classical \obeylines and \obeyspaces.

```
51 % In \obeylines, we say `\let^^M=\_par' instead of `\def^^M{\_par}'
52 % since this allows, for example, `\let\_par=\cr \obeylines \halign{...'
53 {\_catcode`\^^M=13 % these lines must end with %
54   \_gdef\_obeylines{\_catcode`\^^M=13\_let^^M\_par}%
55   \_global\_let^^M\_par} % this is in case ^^M appears in a \write
56 \_def\_obeyspaces{\_catcode`\ =13 }
57 {\_obeyspaces\_global\_let =\_space}
58 \_public \obeylines \obeyspaces ;
```

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

```
68 \_protected\_def\_thinspace {\_kern .16667em }
69 \_protected\_def\_negthinspace {\_kern-.16667em }
70 \_protected\_def\_enspace {\_kern.5em }
71 \_protected\_def\_enskip {\_hskip.5em\_relax}
72 \_protected\_def\_quad {\_hskip1em\_relax}
73 \_protected\_def\_qquad {\_hskip2em\_relax}
74 \_protected\_def\_smallskip {\_vskip\_smallskipamount}
75 \_protected\_def\_medskip {\_vskip\_medskipamount}
76 \_protected\_def\_bigskip {\_vskip\_bigskipamount}
77 \_def\_nointerlineskip {\_prevdepth=-1000pt }
78 \_def\_offinterlineskip {\_baselineskip=-1000pt \_lineskip=0pt \_lineskiplimit=\_maxdimen}
79
80 \_public \thinspace \negthinspace \enspace \enskip \quad \qquad \smallskip
81   \medskip \bigskip \nointerlineskip \offinterlineskip ;
82
83 \_def\_topglue {\_nointerlineskip\_vglue-\_topskip\_vglue} % for top of page
84 \_def\_vglue {\_afterassignment\_vglA \_skip0=}
85 \_def\_vglA {\_par \_dimen0=\_prevdepth \_hrule height0pt
86   \_nobreak\_vskip\_skip0 \_prevdepth=\_dimen0 }
87 \_def\_hglue {\_afterassignment\_hglA \_skip0=}
88 \_def\_hglA {\_leavevmode \_count255=\_spacefactor \_vrule width0pt
89   \_nobreak\_hskip\_skip0 \_spacefactor=\_count255 }
90 \_protected\_def~{\_penalty10000 \ } % tie
91 \_protected\_def\_slash {/\_penalty\_exhyphenpenalty} % a `/' that acts like a `-'
92
93 \_public \topglue \vglue \hglue \slash ;
```

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

```
102 \_protected\_def \_break {\_penalty-10000 }
103 \_protected\_def \_nobreak {\_penalty10000 }
104 \_protected\_def \_allowbreak {\_penalty0 }
105 \_protected\_def \_filbreak {\_par\_vfil\_penalty-200\_vfilneg}
106 \_protected\_def \_goodbreak {\_par\_penalty-500 }
107 \_protected\_def \_eject {\_par\_break}
108 \_protected\_def \_superject {\_par\_penalty-20000 }
109 \_protected\_def \_dosuperject {\_ifnum \_insertpenalties>0 % something is being held over
110   \_line{}\_kern-\_topskip \_nobreak \_vfill \_superject \_fi}
111 \_def \_removelastskip {\_ifdim\_lastskip=\_zo \_else \_vskip-\_lastskip \_fi}
112 \_def \_smallbreak {\_par\_ifdim\_lastskip<\_smallskipamount
113   \_removelastskip \_penalty-50 \_smallskip \_fi}
114 \_def \_medbreak {\_par\_ifdim\_lastskip<\_medskipamount
115   \_removelastskip \_penalty-100 \_medskip \_fi}
116 \_def \_bigbreak {\_par\_ifdim\_lastskip<\_bigskipamount
117   \_removelastskip \_penalty-200 \_bigskip \_fi}
```

56

```
118
119  \_public \break \nobreak \allowbreak \filbreak \goodbreak \eject \supereject \dosupereject
120      \removelastskip \smallbreak \medbreak \bigbreak ;
```

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

```
128  \_def \_line {\_hbox to\_hsize}
129  \_def \_leftline #1{\_line{#1\_hss}}
130  \_def \_rightline #1{\_line{\_hss#1}}
131  \_def \_centerline #1{\_line{\_hss#1\_hss}}
132  \_def \_rlap #1{\_hbox to\_zo{#1\_hss}}
133  \_def \_llap #1{\_hbox to\_zo{\_hss#1}}
134  \_def\_underbar #1{$\_setbox0=\_hbox{#1}\_dp0=\_zo \_math \_underline{\_box0}$}
135
136  \_public \line \leftline \rightline \centerline \rlap \llap \underbar ;
```

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

```
143  \_newbox\_strutbox
144  \_setbox\_strutbox=\_hbox{\_vrule height8.5pt depth3.5pt width0pt}
145  \_def \_strut {\_relax\_ifmmode\_copy\_strutbox\_else\_unhcopy\_strutbox\_fi}
146
147  \_public \strutbox \strut ;
```

Alignment. \hidewidth \ialign \multispan.

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

Tabbing macros are omitted because they are obsolete.

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

```
169  \_def \_hang {\_hangindent\_parindent}
170  \_def \_textindent #1{\_indent\_llap{#1\_enspace}\_ignorespaces}
171  \_def \_item {\_par\_hang\_textindent}
172  \_def \_itemitem {\_par\_indent \_hangindent2\_parindent \_textindent}
173  \_def \_narrower {\_advance\_leftskip\_parindent
174      \_advance\_rightskip\_parindent}
175  \_def \_raggedright {\_rightskip=0pt plus2em
176      \_spaceskip=.3333em \_xspaceskip=.5em\_relax}
177  \_def \_ttraggedright {\_tt \_rightskip=0pt plus2em\_relax} % for use with \tt only
178  \_def \_leavevmode {\_unhbox\_voidbox} % begins a paragraph, if necessary
179
180  \_public \hang \textindent \item \itemitem \narrower \raggedright \ttraggedright \leavevmode ;
```

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

```
191  %\chardef\%=`\%
192  \_let\% = \_pcent  % more natural, can be used in lua codes.
193  \_chardef\&=`\&
194  \_chardef\#=`\#
195  \_chardef\$=`\$
196  \_chardef\ss="FF
197  \_chardef\ae="E6
198  \_chardef\oe="F7
199  \_chardef\o="F8
200  \_chardef\AE="C6
201  \_chardef\OE="D7
```

```
202  \_chardef\O="D8
203  \_chardef\i="11 \chardef\j="12 % dotless letters
204  \_chardef\aa="E5
205  \_chardef\AA="C5
206  \_chardef\S="9F
207  \_def\l{\_errmessage{\_usedirectly ł}}
208  \_def\L{\_errmessage{\_usedirectly Ł}}
209  %\def\_{\_ifmmode \kern.06em \vbox{\hrule width.3em}\else _\fi} % obsolete
210  \_def\_{\_hbox{_}}
211  \_def\dag{\_errmessage{\_usedirectly †}}
212  \_def\ddag{\_errmessage{\_usedirectly ‡}}
213  \_def\copyright{\_errmessage{\_usedirectly ©}}
214  %\_def\Orb{\_mathhexbox20D} % obsolete (part of Copyright)
215  %\_def\P{\_mathhexbox27B}   % obsolete
216
217  \_def \_usedirectly #1{Load Unicoded font by \string\fontfam\space and use directly #1}
218  \_def \_mathhexbox #1#2#3{\_leavevmode \_hbox{$\_math \_mathchar"#1#2#3$}}
219  \_public \mathhexbox ;
```

The `\_unichars` macro is run in `\initunifonts`, Unicodes are used instead old plain TeX settings.

```
226  \def\_unichars{% characters with different codes in Unicode:
227      \_chardef\ss=`ß
228      \_chardef\oe=`œ
229      \_chardef\OE=`Œ
230      \_chardef\S=`§
231      \_chardef\dag`†
232      \_chardef\ddag`‡
233      \_chardef\copyright`©
234  }
```

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

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

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

```
264  \_def \_oldaccents {%
265      \_def\`##1{{\_accent\_tgrave  ##1}}%
266      \_def\'##1{{\_accent\_tacute  ##1}}%
267      \_def\v##1{{\_accent\_caron   ##1}}%
268      \_def\u##1{{\_accent\_tbreve  ##1}}%
269      \_def\=##1{{\_accent\_macron ##1}}%
270      \_def\^##1{{\_accent\_circumflex ##1}}%
271      \_def\.##1{{\_accent\_dotaccent ##1}}%
272      \_def\H##1{{\_accent\_hungarumlaut ##1}}%
273      \_def\~##1{{\_accent\_ttilde  ##1}}%
274      \_def\"##1{{\_accent\_dieresis ##1}}%
275      \_def\r##1{{\_accent\_ring    ##1}}%
276  }
277  \_public \oldaccents ;
278
279  % ec-lmr encoding (will be changed after \fontfam macro):
280  \_chardef\_tgrave=0
281  \_chardef\_tacute=1
282  \_chardef\_circumflex=2
```

```
283  \_chardef\_ttilde=3
284  \_chardef\_dieresis=4
285  \_chardef\_hungarumlaut=5
286  \_chardef\_ring=6
287  \_chardef\_caron=7
288  \_chardef\_tbreve=8
289  \_chardef\_macron=9
290  \_chardef\_dotaccent=10
291  \_chardef\_cedilla=11
292
293  \_def \_uniaccents {% accents with Unicode
294      \_chardef\_tgrave="0060
295      \_chardef\_tacute="00B4
296      \_chardef\_circumflex="005E
297      \_chardef\_ttilde="02DC
298      \_chardef\_dieresis="00A8
299      \_chardef\_hungarumlaut="02DD
300      \_chardef\_ring="02DA
301      \_chardef\_caron="02C7
302      \_chardef\_tbreve="02D8
303      \_chardef\_macron="00AF
304      \_chardef\_dotaccent="02D9
305      \_chardef\_cedilla="00B8
306      \_chardef\_ogonek="02DB
307      \_let \_uniaccents=\_relax
308  }
```

The plain TEX macros \hrulefill, \dotfill, \rightarrowfill, \leftarrowfill, \downbracefill, \upbracefill. The last four are used in non-Unicode variants of \overrightarrow, \overleftarrow, \overbrace and \underbrace macros, see section 2.15.

plain-macros.opm
```
319  \_def \_hrulefill {\_leaders\_hrule\_hfill}
320  \_def \_dotfill {\_cleaders\_hbox{$\_math \_mkern1.5mu.\_mkern1.5mu$}\_hfill}
321  \_def \_rightarrowfill {$\_math\_smash-\_mkern-7mu%
322    \_cleaders\_hbox{$\_mkern-2mu\_smash-\_mkern-2mu$}\_hfill
323    \_mkern-7mu\_mathord\_rightarrow$}
324  \_def \_leftarrowfill {$\_math\_mathord\_leftarrow\_mkern-7mu%
325    \_cleaders\_hbox{$\_mkern-2mu\_smash-\_mkern-2mu$}\_hfill
326    \_mkern-7mu\_smash-$}
327
328  \_mathchardef \_braceld="37A \_mathchardef \_bracerd="37B
329  \_mathchardef \_bracelu="37C \_mathchardef \_braceru="37D
330  \_def \_downbracefill {$\_math \_setbox0=\_hbox{$\_braceld$}%
331    \_braceld \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_braceru
332    \_bracelu \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_bracerd$}
333  \_def \_upbracefill {$\_math \_setbox0=\_hbox{$\_braceld$}%
334    \_bracelu \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_bracerd
335    \_braceld \_leaders\_vrule height\_ht0 depth\_zo \_hfill \_braceru$}
336
337  \_public \hrulefill \dotfill
338      \rightarrowfill \leftarrowfill \downbracefill \upbracefill ;
```

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

plain-macros.opm
```
346  \_def \_magnification {\_afterassignment \_magA \_count255 }
347  \_def \_magA {\_mag=\_count255 \_truedimen\_hsize \_truedimen\_vsize
348    \_dimen\_footins=8truein
349  }
350  % only for backward compatibility, but \margins macro is preferred.
351  \_public \magnification ;
352
353  \_def \_showhyphens #1{\_setbox0=\_vbox{\_parfillskip=0pt \_hsize=\_maxdimen \_tenrm
354    \_pretolerance=-1 \tolerance=-1 \hbadness=0 \showboxdepth=0 \ #1}}
355
356  \_def \_bye {\_par \_vfill \_supereject \_byehook \_end}
357  \_public \showhyphens \bye ;
```

Plain TEX reads hyphen.tex with patterns as \language=0. We do the same.

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

2.11   Preloaded fonts for text mode

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

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

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

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

```
 3  \_codedecl \tenrm {Latin Modern fonts (EC) preloaded <2022-02-12>} % preloaded in format
 4
 5  \_ifx\fontspreload\_relax
 6     \_let\_tenrm=\_nullfont \_let\_tenbf=\_nullfont \_let\_tenit=\_nullfont
 7     \_let\_tenbi=\_nullfont \_let\_tentt=\_nullfont
 8  \_else
 9     % Only few text fonts are preloaded:
10     % allow missing fonts during format generation
11     \_suppressfontnotfounderror=1
12     \_font\_tenrm=ec-lmr10   % roman text
13     \_font\_tenbf=ec-lmbx10  % boldface extended
14     \_font\_tenit=ec-lmri10  % text italic
15     \_font\_tenbi=ec-lmbxi10 % bold italic
16     \_font\_tentt=ec-lmtt10  % typewriter
17     \_suppressfontnotfounderror=0
18  \_fi
19
20  \_tenrm
21
22  \_public \tenrm \tenbf \tenit \tenbi \tentt ;
```

2.12   Using `\font` primitive directly

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

```
\font \⟨font switch⟩ = ⟨font file name⟩ ⟨size spec⟩
% for example:
\font \tipa = tipa10 at12pt % the font tipa10 at 10pt is loaded
% usage:
{\tipa TEXT}  % the TEXT is printed in the loaded font.
```

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

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

```
\font \⟨font switch⟩ = {[⟨font file name⟩]:⟨font features⟩} ⟨size spec⟩
% or
\font \⟨font switch⟩ = {⟨font name⟩:⟨font features⟩} ⟨size spec⟩
```

where ⟨*font file name*⟩ is name of the OpenType font file without extension (extensions `.otf` or `.ttf` are assumed). The braces in the syntax are optional, use them when the ⟨*font file name*⟩ or ⟨*font name*⟩ includes spaces. The original syntax for `tfm` files is also available. Example:

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

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

Note, that \fontfam does (among other things) \initunifonts internally. You need not to specify it if \fontfam is used.

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

2.12.1 The \setfontsize macro

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

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

The initial value in OpTeX is given by \setfontsize{at10pt}.

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

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

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

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

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

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

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

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

This is not only an academical example. The \csrimson command defined here behaves like variant selector in the Font Selection System (section 2.13). It takes only information about size from the font context, but it is sufficient. You can use it in titles, footnotes, etc. The font size depending on surrounding size is automatically selected.

2.12.2 The \fontlet declarator

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

>   \fontlet \⟨new font switch⟩ = \⟨given font switch⟩ ⟨size spec⟩
>   example:
>   \fontlet \bigfont = _tenbf at15pt

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

>   \fontlet \⟨new font switch⟩ = \font ⟨size spec⟩

2.12.3 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". Each design size is implemented in its individual font file and OpTeX is able to choose right file if various optical sizes and corresponding file names are declared for the font by _regtfm or _regoptsizes command. The command \setfontsize sets the internal reuirements for optical size if the parameter is in the format at⟨dimen⟩ or mag⟨factor⟩. Then the command \resizethefont or variant selectors try to choose the font suitable for the required optical size. For example

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

See also section 2.13.12.

2.12.4 Font rendering

If \initunifonts isn't declared then OpTeX uses classical font renderer (like in pdftex). The extended font renderer implemented in the Luaotfload package is started after \initunifonts.

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

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

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

2.12.5 Implementation of resizing

Only "resizing" macros and \initunifonts are implemented here. Other aspects of Font Selection System and their implementation are desribed in section 2.13.14.

<div align="right">fonts-resize.opm</div>

```
  3 \_codedecl \setfontsize {Font resizing macros <2022-02-22>} % preloaded in format
```

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

<div align="right">fonts-resize.opm</div>

```
 15 \_def\_initunifonts {%
 16    \_directlua{%
 17       require('luaotfload-main')
 18       luaotfload.main()
 19       optex_hook_into_luaotfload()
 20    }%
 21    \_glet \_fmodtt=\_unifmodtt  % use \_ttunifont for \tt
 22    \_glet \_initunifonts=\_relax % we need not to do this work twice
 23    \_glet \initunifonts=\_relax
 24 }
 25 \_public \initunifonts ;
```

The **\setfontsize** {⟨*size spec*⟩} saves the ⟨*size spec*⟩ to the **_sizespec** macro. The **_optsize** value is calculated from the ⟨*size spec*⟩. If the ⟨*size spec*⟩ is in the format `scaled`⟨*factor*⟩ then **_optsize** is set from **\defaultoptsize**. If the ⟨*size spec*⟩ is in the `mag`⟨*number*⟩ format then the contents of the **_sizespec** macro is re-calculated to the `at`⟨*dimen*⟩ format using previous **_optsize** value.

```
38  \_newdimen \_optsize          \_optsize=10pt
39  \_newdimen \_defaultoptsize  \_defaultoptsize=10pt
40  \_newdimen\_lastmagsize
41
42  \_def\_setfontsize #1{%
43      \_edef\_sizespec{#1}%
44      \_ea \_setoptsize \_sizespec\_relax
45  }
46  \_def\_setoptsize {\_isnextchar a{\_setoptsizeA}
47                                  {\_isnextchar m{\_setoptsizeC}{\_setoptsizeB}}}
48  \_def\_setoptsizeA at#1\_relax{\_optsize=#1\_relax\_lastmagsize=\_optsize}   % at<dimen>
49  \_def\_setoptsizeB scaled#1\_relax{\_optsize=\_defaultoptsize\_relax} % scaled<scalenum>
50  \_def\_setoptsizeC mag#1\_relax{%
51      \_ifdim\_lastmagsize>\_zo \_optsize=\_lastmagsize \_else \_optsize=\_pdffontsize\_font \_fi
52      \_optsize=#1\_optsize
53      \_lastmagsize=\_optsize
54      \_edef\_sizespec{at\_the\_optsize}%
55  }
56  \_public \setfontsize \defaultoptsize ;
```

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

```
65  \_edef\_stringat{\_string a\_string t}
66  \_edef\_xfontname#1{\_unexpanded{\_ea\_xfontnameA\_fontname}#1 \_stringat\_relax}
67  \_expanded{\_def\_noexpand\_xfontnameA#1 \_stringat#2\_relax}{{#1}}
```

\fontlet ⟨*font switch A*⟩ ⟨*font switch B*⟩ ⟨*size spec*⟩ does

> **\font** ⟨*font switch A*⟩ = {⟨*font file name*⟩} ⟨*size spec*⟩

Note, that the **_xfontname** output is converted due to optical size data using **_optfn**.

```
77  \_def\_fontlet#1#2{\_ifx #2=\_ea\_fontlet \_ea#1\_else \_font #1{\_optfn{\_xfontname#2}}\_fi}
78  \_public \xfontname \fontlet ;
```

\newcurrfontsize ⟨*size spec*⟩ does **\fontlet** ⟨*saved switch*⟩=\font ⟨*size spec*⟩_relax ⟨*saved switch*⟩. It changes the current font at the given ⟨*size spec*⟩.
\resizethefont is implemented by **\newcurrfontsize** using data from the **_sizespec** macro.

```
89  \_def \_newcurrfontsize #1{% \newcurrfontsize{at25pt}
90      \_ea\_def \_ea\_tmp \_ea{\_csname \_ea\_csstring \_the\_font \_endcsname}%
91      \_ea\_fontlet \_tmp \_font #1\_relax
92      \_ea\_fontloaded \_tmp
93      \_tmp
94  }
95  \_protected\_def \_resizethefont{\_newcurrfontsize\_sizespec}
96  \_public \newcurrfontsize \resizethefont ;
```

The **_regtfm** ⟨*font id*⟩ ⟨*optical size data*⟩ registers optical sizes data directly by the font file names. This can be used for `tfm` files or OpenType files without various font features. See also **_regoptsizes** in section 2.13.12. The **_regtfm** command saves the ⟨*optical size data*⟩ concerned to the ⟨*font id*⟩. The ⟨*optical size data*⟩ is in the form as shown below in the code where **_regtfm** is used.
The **_optfn** ⟨*fontname*⟩ expands to the ⟨*fontname*⟩ or to the corrected ⟨*fontname*⟩ read from the ⟨*optical size data*⟩ registered by **_regtfm**. It is used in the **\fontlet** macro.

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

```
119 \_def\_regtfm #1 0 #2 *{\_ea\_def \_csname _reg:#1\_endcsname{#2 16380 \_relax}%
120   \_def\_tmpa{#1}\_reversetfm #2 * %
121 }
122 \_def\_reversetfm #1 #2 {% we need this data for \_setmathfamily
123   \_ea\_let\_csname _reg:#1\_ea\_endcsname
124   \_csname _reg:\_tmpa\_endcsname
125   \_if*#2\_else \_ea\_reversetfm \_fi
126 }
127 \_def\_optfn #1{%
128   \_ifcsname _reg:#1\_endcsname
129     \_ea\_ea\_ea \_runoptfn
130     \_csname _reg:#1\_ea\_endcsname
131   \_else
132     #1%
133   \_fi
134 }
135 \_def\_runoptfn #1 #2 {%
136   \_ifdim\_optsize<#2pt #1\_ea\_ignoretfm\_else \_ea\_runoptfn
137 \_fi
138 }
139 \_def\_ignoretfm #1\_relax{}
```

Optical sizes data for preloaded 8bit Latin Modern fonts:

```
145 \_regtfm lmr  0 ec-lmr5 5.5 ec-lmr6 6.5 ec-lmr7 7.5 ec-lmr8 8.5 ec-lmr9 9.5
146                    ec-lmr10 11.1 ec-lmr12 15 ec-lmr17 *
147 \_regtfm lmbx 0 ec-lmbx5 5.5 ec-lmbx6 6.5 ec-lmbx7 7.5 ec-lmbx8 8.5 ec-lmbx9 9.5
148                    ec-lmbx10 11.1 ec-lmbx12 *
149 \_regtfm lmri 0 ec-lmri7 7.5 ec-lmri8 8.5 ec-lmri9 9.5 ec-lmri10 11.1 ec-lmri12 *
150 \_regtfm lmtt 0 ec-lmtt8 8.5 ec-lmtt9 9.5 ec-lmtt10 11.1 ec-lmtt12 *
```

2.13   The Font Selection System

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

2.13.1   Terminology

We distinguish between

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

These commands set their values locally. When the TEX group is left then the selected font and the *font context* are returned back to the values used when the group was opened. They have the following features:

The *font context* is a set of macro values that will affect the selection of real font when the variant selector is processed. It includes the value of *current family*, current font size, and more values stored by font modifiers.

The *family context* is the current family name stored in the font context. The variant selectors declared by \famvardef and font modifiers declared by \moddef are dependent on the *family context*. They can have the same names but different behavior in different families.

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

64

2.13.2 Font families, selecting fonts

The `\fontfam` [⟨*Font Family*⟩] opens the relevant font family file where the ⟨*Font Family*⟩ is declared.
The family selector is defined here by rules described in the section 2.13.11. Font modifiers and variant
selectors may be declared here. The loaded family is set as current and `\rm` variant selector is processed.

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

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

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

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

For example:

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

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

You can use the `\setfontsize` {⟨*size spec*⟩} command in the same sense as other font modifiers. It
saves information about font size to the font context. See section 2.12.1. Example:

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

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

2.13.3 Math Fonts

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

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

2.13.4 Declaring font commands

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

- Font switches can be decared by `\font` primitive (see section 2.12) or by `\fontlet` command (see
  section 2.12.2) or by `\fontdef` command (see sections 2.13.5). When the font switches are used then
  they select the given font independently of the current font context. They can be used in `\output`
  routine (for example) because we need to set fixed fonts in headers and footers.
- Variant selectors are `\rm`, `\bf`, `\it`, `\bi`, `\tt` and `\currvar`. More variant selectors can be declared by
  `\famvardef` command. They select a font dependent on the current font context, see section 2.13.6.
  The `\tt` selector is documented in section 2.13.7.
- Font modifiers are "built-in" or declared by `\moddef` command. They do modifications in the font
  context but don't select any font.

- "built-in" font modifiers are `\setfontsize` (see section 2.12.1), `\setff` (see section 2.13.9), `\setletterspace` and `\setwordspace` (see section 2.13.10). They are independent of font family.
- Font modifiers declared by `\moddef` depend on the font family and they are typically declared in font family files, see section 2.13.11.
- Family selectors set the given font family as current and re-set data used by the family-dependent font modifiers to initial values and to the currently used modifiers. They are declared in font family files by `\_famdecl` macro, see section 2.13.11.
- Font macros can be defined arbitrarily by `\def` primitive by users. See an example in section 2.13.8.

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

2.13.5 The `\fontdef` declarator in detail

You can declare `\`⟨*font-switch*⟩ by the `\fontdef` command.

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

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

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

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

2.13.6 The `\famvardef` declarator

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

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

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

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

Suppose that the selected font family provides the font modifier `\medium` for mediate weight of fonts. Then you can declare:

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

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

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

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

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

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

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

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

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

2.13.7 The \tt variant selector

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

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

The behavior of \tt can be re-defined by \famvardef. For example:

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

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

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

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

2.13.8 Font commands defined by \def

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

The following example shows how to define a "title-font selector". Titles are not only bigger but they are typically in the bold variant. When a user puts {\it...} into the title text then he/she expects bold italic here, no normal italic. You can remember the great song by John Lennon "Let It Be" and define:

```
\def\titlefont{\setfontsize{at14pt}\bf \let\it\bi}
...
{\titlefont Title in bold 14pt font and {\it bold 14pt italics} too}
```

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

2.13.9 Modifying font features

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

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

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

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

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

2.13.10 Special font modifiers

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

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

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

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

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

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

```
\setff{embolden=1.5}\rm  % font is bolder because outline has nonzero width
\setff{slant=0.2}\rm     % font is slanted by a linear transformation
\setff{extend=1.2}\rm    % font is extended by a linear transformation.
\setff{colr=yes}\rm      % if the font includes colored characters, use colors
\setff{upper}\rm         % to uppercase (lower=lowercase) conversion at font level
\setff{fallback=name}\rm % use fonts from a list given by name if missing chars
```

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

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

2.13.11 How to create the font family file

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

```
 3  \_famdecl [Heros] \Heros {TeX Gyre Heros fonts based on Helvetica}
 4      {\caps \cond} {\rm \bf \it \bi} {FiraMath}
 5      {[texgyreheros-regular]}
 6      {\_def\_fontnamegen{[texgyreheros\_condV-\_currV]:\_capsV\_fontfeatures}}
 7
 8  \_wlog{\_detokenize{%
 9  Modifiers:^^J
10   \caps ...... caps & small caps^^J
11   \cond ...... condensed variants^^J
12  }}
13
14  \_moddef \resetmod {\_fsetV caps={},cond={} \_fvars regular bold italic bolditalic }
15  \_moddef \caps     {\_fsetV caps=+smcp;\_ffonum; }
16  \_moddef \nocaps   {\_fsetV caps={} }
17  \_moddef \cond     {\_fsetV cond=cn }
18  \_moddef \nocond   {\_fsetV cond={} }
19
20  \_initfontfamily % new font family must be initialized
21
22  \_ifmathloading
23      \_loadmath {[FiraMath-Regular]}
24      \_addto\_normalmath{\_loadumathfamily 5 {xitsmath-regular}{} }
25      \_addto\_boldmath  {\_loadumathfamily 5 {xitsmath-bold}{} }
26      \_addto\_frak{\_fam5 }\_addto\_cal{\_fam5 } \_public \frak \cal ;
27      \_normalmath
28      \_wterm{MATH-FONT(5): "[XITSMath-Regular/Bold]" -- used for \_string\cal, \_string\frak}
29      % \bf, \bi from FiraMath:
30      \_let\_bsansvariables=\_bfvariables
31      \_let\_bsansGreek=\_bfGreek
32      \_let\_bsansgreek=\_bfgreek
33      \_let\_bsansdigits=\_bfdigits
34      \_let\_bisansvariables=\_bivariables
35      \_let\_bisansgreek=\_bigreek
36      % \_resetmathchars <fam-number> <list of \Umathchardef csnames> ;
37      \_def\_resetmathchars #1{\_chardef\_mafam=#1\_relax \_xargs \_resetmathcharX}
38      \_def\_resetmathcharX#1{\_ea\_resetmathcharY
39        \_directlua{tex.print(string.format("\_pcent08X", \_the#1))}#1}
40      \_def\_resetmathcharY#1#2#3#4#5#6#7#8#9{%
41        \_Umathchardef #9\_numexpr"#3/2\_relax \_mafam "#4#5#6#7#8
42        \_Umathcode "#4#5#6#7#8=\_numexpr"#3/2\_relax \_mafam "#4#5#6#7#8 }
43      \_resetmathchars 5 \bigtriangleup \bigblacktriangleup \blacktriangle
44        \vartriangle \smallblacktriangleright ; % ... etc. you can add more
45  \_fi
```

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

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

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

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

69

- If math font should be loaded, use `\_loadmath{`⟨*math font*⟩`}`.

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

    `\font \`⟨*font-switch*⟩ `{\_fontnamegen} \_sizespec`

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

Example 1
Assume an abstract font family with fonts `xx-Regular.otf`, `xx-Bold.otf`, `xx-Italic.otf` and `xx-BoldItalic.otf`. Then you can declare the `\resetmod` (for initializing the family) by:

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

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

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

The following auxiliary macros are used here:

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

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

    See more advanced examples are in `f-`⟨*family*⟩`.opm` files.

Example 2
The `f-heros.opm` is listed here. Look at it. When Heros family is selected and `\bf` is asked then `\font {[texgyreheros-bold]:+tlig;} at10pt` is processed.

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

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

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

Example 3
If both font modifiers `\caps`, `\cond` were applied in Heros family, then `\def\_capsV{+smcp;\_ffonum;}` and `\def\_condV{cn}` were processed by these font modifiers. If a user needs the `\bf` variant at 11 pt now then the

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

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

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

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

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

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

The \famvardef macro has the same features.

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

```
\protected\def\⟨Familyselector⟩ {%
   \_def\_currfamily {⟨Familyselector⟩}%
   \_def\_fontnamegen {...}% this is copied from 7-th parameter of \_famdecl
   \resetmod
   ⟨run all family-dependent font modifiers used before Familyselector without warnings⟩
```

The _initfontfamily
must be run after modifier's decaration. It runs the \⟨*Familyselector*⟩ and it runs _rm, so the first font
from the new family is loaded and it is ready to use it.

Name conventions
Create font modifiers, new variants, and the \⟨*Familyselector*⟩ only as public, i.e. in user namespace
without _ prefix. We assume that if a user re-defines them then he/she needs not them, so we have no
problems. If the user's definition was done before loading the font family file then it is re-defined and
OpTEX warns about it. See the end of section 2.13.4.

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

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

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

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

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

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

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

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

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

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

2.13.12 How to write the font family file with optical sizes

You can use _optname macro when _fontnamegen 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 < $ `\_optsize` ≤ 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 with an error message about missing font. You can redefine `\_failedoptname` macro to some existing font if you find it useful.

We are using a special macro `\_LMregfont` in `f-lmfonts.opm`. It sets the file names to lowercase and enables us to use shortcuts instead of real ⟨*resizing-data*⟩. There are shortcuts `\_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}
```

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

```
 3 \_famdecl [Poltawski] \Poltawski {Antykwa Poltawskiego, Polish traditional font family}
 4     {\light \noexpd \expd \eexpd \cond \ccond \osize \caps} {\rm \bf \it \bi} {}
 5     {[antpolt-regular]}
 6     {\_def\_fontnamegen {[antpolt\_liV\_condV-\_currV]\_capsV\_fontfeatures}}
 7
 8 \_wlog{\_detokenize{%
 9 Modifiers:^^J
10  \light ..... light weight, \bf,\bi=semibold^^J
11  \noexpd .... no expanded, no condensed, designed for 10pt size (default)^^J
12  \eexpd ..... expanded, designed for 6pt size^^J
13  \expd ...... semi expanded, designed for 8pt size^^J
14  \cond ...... semi condensed, designed for 12pt size^^J
15  \ccond ..... condensed, designed for 17pt size^^J
16  \osize ..... auto-sitches between \ccond \cond \noexpd \expd \eexpd by size^^J
17  \caps ...... caps & small caps^^J
18 }}
19
20 \_moddef \resetmod {\_fsetV li={},cond={},caps={} \_fvars regular bold italic bolditalic }
21 \_moddef \light     {\_fsetV li=lt }
22 \_moddef \noexpd    {\_fsetV cond={} }
23 \_moddef \eexpd     {\_fsetV cond=expd }
24 \_moddef \expd      {\_fsetV cond=semiexpd }
25 \_moddef \cond      {\_fsetV cond=semicond }
26 \_moddef \ccond     {\_fsetV cond=cond }
```

```
27  \_moddef \caps      {\_fsetV caps=+smcp;\_ffonum; }
28  \_moddef \nocaps    {\_fsetV caps={} }
29  \_moddef \osize     {\_def\_fontnamegen{[antpolt\_liV\_optname{x}-\_currV]:\_capsV\_fontfeatures}%
30                        \_regoptsizes x ? expd <7 semiexpd <9 {} <11.1 semicond <15 cond <*}
31
32  \_initfontfamily % new font family must be initialized
```

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

Once you have prepared a font family file with the name f-⟨*famname*⟩.opm and TeX can see it in
your filesystem then you can type \fontfam[⟨*famname*⟩] and the file is read, so the information about
the font family is loaded. The name ⟨*famname*⟩ must be lowercase and without spaces in the file name
f-⟨*famname*⟩.opm. On the other hand, the \fontfam command is more tolerant: you can write uppercase
letters and spaces here. The spaces are ignored and uppercase letters are converted to lowercase. For
example \fontfam [LM Fonts] is equivalent to \fontfam [LMfonts] and both commands load the file
f-lmfonts.opm.

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

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

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

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

The _faminfo commad has the syntax:

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

The ⟨*mod-plus-vars*⟩ data is used only when printing the catalog. It consists of one or more pairs
⟨*mods*⟩: {⟨*vars*⟩}. For each pair: each modifier (separated by comma) is applied to each variant selector
in ⟨*vars*⟩ and prepared samples are printed. The - character means no modifiers should be applied.

The _famalias declares an alias to the last declared family.

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

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

2.13.14 Implementation of the Font Selection System

```
3 \_codedecl \fontfam {Fonts selection system <2022-02-22>} % preloaded in format
```

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

```
17 \_def\_famv{rm}  % default value
18 \_protected\_def \_fmodrm {\_def\_famv{rm}}
19 \_protected\_def \_fmodbf {\_def\_famv{bf}}
20 \_protected\_def \_fmodit {\_def\_famv{it}}
21 \_protected\_def \_fmodbi {\_def\_famv{bi}}
22 \_protected\_def \_fmodtt {\_def\_famv{tt}}
23
24 \_protected\_def \_rm {\_fmodrm \_fontsel \_marm}
25 \_protected\_def \_bf {\_fmodbf \_fontsel \_mabf}
26 \_protected\_def \_it {\_fmodit \_fontsel \_mait}
27 \_protected\_def \_bi {\_fmodbi \_fontsel \_mabi}
28 \_protected\_def \_tt {\_fmodtt \_fontsel \_matt}
29 \_protected\_def \_currvar {\_fontsel} \_protected\_def \currvar{\_currvar}
30 \_public \rm \bf \it \bi \tt ;
```

The \fontsel creates the ⟨font switch⟩ in the format _ten⟨famv⟩ and loads the font associated to the ⟨font switch⟩. The loading is done by:

a) \letfont ⟨font switch⟩ = \savedswitch _sizespec
b) \font ⟨font switch⟩ = \fontnamegen _sizespec

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

After the font is loaded, final job is done by _fontselA⟨font-switch⟩.

```
47 \_protected\_def \_fontsel {%
48    \_ifx\_fontnamegen\_undefined % \fontfam was not used
49       \_ea\_let \_ea\_tmpf \_csname _ten\_famv\_endcsname
50       \_ea\_fontlet \_csname _ten\_xfamv\_endcsname \_tmpf \_sizespec
51    \_else % \fontfam is used
52       \_ea\_font \_csname _ten\_xfamv\_endcsname {\_fontnamegen}\_sizespec
53    \_fi \_relax
54    \_ea \_fontselA \_csname _ten\_xfamv\_endcsname
55 }
56 \_def\_fontselA #1{%
57    \_protected\_def \_currvar {\_fontsel}% default value of \_currvar
58    \_logfont #1%     font selecting should be logged.
59    \_setwsp  #1%     wordspace setting
60    \_fontloaded #1% initial settings if font is loaded firstly
61    #1% select the font
62 }
63 \_def \_logfont #1{}
64 \_def \_xfamv {\_famv}
65
66 \_public \fontsel ;
```

If a font is loaded by macros \fontsel or \resizethefont then the _fontloaded⟨font switch⟩ is called immediately after it. If the font is loaded first then its \skewchar is equal to −1. We run

`\_newfontloaded`⟨*font switch*⟩ and set `\skewchar=-2` in this case. A user can define a `\_newfontloaded` macro. We are sure that `\_newfontloaded` macro is called only once for each instance of the font given by its name, OTF features and size specification. The `\skewchar` value is globally saved to the font (like `\fontdimen`). If it is used in math typesetting then it is set to a positive value.

The `\_newfontloaded` should be defined for micro-typographic configuration of fonts, for example. The `mte.opm` package uses it. See also OpTeX trick 0058.

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

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

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

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

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

The `\_famdecl` [⟨*Family Name*⟩] \⟨*Famselector*⟩ {⟨*comment*⟩} {⟨*modifiers*⟩} {⟨*variants*⟩} {⟨*math*⟩} {⟨*font for testing*⟩} {\def`\_fontnamegen`{⟨*data*⟩}} runs `\initunifonts`, then checks if \⟨*Famselector*⟩ is defined. If it is true, then closes the file by `\endinput`. Else it defines \⟨*Famselector*⟩ and saves it to the internal `\_f:`⟨*currfamily*⟩`:main.fam` command. The macro `\_initfontfamily` needs it. The `\_currfamily` is set to the ⟨*Famselector*⟩ because the following `\moddef` commands need to be in the right font family context. The `\_currfamily` is set to the ⟨*Famselector*⟩ by the \⟨*Famselector*⟩ too, because \⟨*Famselector*⟩ must set the right font family context. The font family context is given by the current `\_currfamily` value and by the current meaning of the `\_fontnamegen` macro. The `\_mathfaminfo` is saved for usage in the catalog.

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

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

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

`\_fsetV` ⟨*key*⟩=⟨*value*⟩,...,⟨*key*⟩=⟨*value*⟩ expands to \def`\_`⟨*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*⟩.

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

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

```
171 \_def\_fvars #1 #2 #3 #4 {%
172    \_sdef{_fvar:rm}{#1}%
173    \_sdef{_fvar:bf}{#2}%
174    \_ifx.#2\_slet{_fvar:bf}{_fvar:rm}\_fi
175    \_sdef{_fvar:it}{#3}%
176    \_ifx.#3\_slet{_fvar:it}{_fvar:rm}\_fi
177    \_sdef{_fvar:bi}{#4}%
178    \_ifx.#4\_slet{_fvar:bi}{_fvar:it}\_fi
179 }
180 \_def\_currV{\_trycs{_fvar:\_famv}{rm}}
181 \_def\_V{ }
182 \_def \_fsetV #1 {\_fsetVa #1,=,}
183 \_def \_fsetVa #1=#2,{\_isempty{#1}\_iffalse
184    \_ifx,#1\_else\_sdef{_#1V}{#2}\_ea\_ea\_ea\_fsetVa\_fi\_fi
185 }
186 \_def \_onlyif #1=#2:#3{%
187    \_edef\_act{\_noexpand\_isinlist{,\_prepcommalist #2,\_end,}{,\_cs{_#1V},}}\_act
188    \_iftrue #3\_fi
189 }
190 \_def\_prepcommalist#1,{\_ifx\_end#1\_empty\_else #1,\_ea\_prepcommalist\_fi}
191 \_def\_ffonum {+onum;+pnum}
```

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

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

$\_addtomodlist$ ⟨*font modifier*⟩ adds given modifier to $\_modlist$ macro. This list is used after **\resetmod** when a new family is selected by a family selector, see **_resetfam** macro. This allows reinitializing the same current modifiers in the font context after the family is changed.

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

76

```
242      \_fi\_fi
243      \_let#1=\_tmpa
244  }
245  \_public \moddef ;
```

\fontdef ⟨*font-switch*⟩ {⟨*data*⟩} does:

> \begingroup ⟨*data*⟩ \ea\endgroup \ea\let \ea⟨*font-switch*⟩ \the\font

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

```
261  \_def\_fontdef #1#2{\_begingroup
262      \_edef\_xfamv{\_csstring#1}\_let\_ttfamv\_xfamv #2%
263      \_ea\_endgroup\_ea \_let\_ea #1\_the\_font
264  }
265  \_public \fontdef ;
```

The **\famvardef** \xxx {⟨*data*⟩} does, roughly speaking:

> \def \xxx {{⟨*data*⟩\ea}\the\font \def_currvar{\xxx}}

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

```
281  \_def\_famvardef#1#2{%
282      \_sdef{_f:\_currfamily:\_csstring#1}%
283          {{\_edef\_xfamv{\_csstring#1}\_let\_ttfamv\_xfamv #2\_ea}\_the\_font \_def\_currvar{#1}}%
284      \_protected\_edef\_tmpa {%
285          \_noexpand\_famdepend\_noexpand#1{_f:\_noexpand\_currfamily:\_csstring#1}}%
286      \_ifx #1\tt
287          \_protected\_def\_tt{{\_def\_xfamv{tt}#2\_ea}\_the\_font \_def\_currvar{\_tt}}%
288          \_let\tt=\_tt
289      \_else \_setnewmeaning #1=\_tmpa \famvardef
290      \_fi
291  }
292  \_public \famvardef ;
```

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

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

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

```
310  \_def\_fontfam[#1]{%
311      \_lowercase{\_edef\_famname{\_ea\_removespaces #1 {} }}%
312      \_isfile {f-\_famname.opm}\_iftrue \_opinput {f-\_famname.opm}%
313      \_else
314          \_tryloadfamslocal
315          \_edef\_famfile{\_trycs{_famf:\_famname}{}}%
316          \_ifx\_famfile\_empty \_listfamnames
317          \_else \_opinput {\_famfile.opm}%
318      \_fi\_fi
319  }
320  \_def\_tryloadfamslocal{%
321      \_isfile {fams-local.opm}\_iftrue
322          \_opinput {fams-local.opm}\_famfrom={}%
```

77

```
323     \_fi
324     \_let \_tryloadfamslocal=\_relax  % need not to load fams-local.opm twice
325 }
326 \_def\_listfamnames {%
327     \_wterm{===== List of font families ======}
328     \_begingroup
329         \_let\_famtext=\_wterm
330         \_def\_faminfo [##1]##2##3##4{%
331             \_wterm{ \_space\_noexpand\fontfam [##1] -- ##2}%
332         \_let\_famalias=\_famaliasA}%
333         \_opinput {fams-ini.opm}%
334         \_isfile {fams-local.opm}\_iftrue \_opinput {fams-local.opm}\_fi
335         \_message{^^J}%
336     \_endgroup
337 }
338 \_def\_famaliasA{\_message{ \_space\_space\_space\_space -- alias:}
339     \_def\_famalias[##1]{\_message{[##1]}}\_famalias
340 }
341 \_public \fontfam ;
```

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

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

`\_faminfo` [⟨*Family Name*⟩] {⟨*comments*⟩} {⟨*file-name*⟩} {⟨*mod-plus-vars*⟩} does `\_def \_famf:`⟨*familyname*⟩ {⟨*file-name*⟩} or prints information on the terminal.

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

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

```
364 \_def\_famtext #1{}
365 \_def\_faminfo [#1]#2#3#4{%
366     \_lowercase{\_edef\_tmp{\_ea\_removespaces #1 {} }}%
367     \_sdef{_famf:\_tmp}{#3}%
368     \_def\_famfile{#3}%
369 }
370 \_def\_famalias [#1]{%
371     \_lowercase{\_edef\_tmpa{\_ea\_removespaces #1 {} }}%
372     \_sdef{_famf:\_tmpa\_ea}\_ea{\_famfile}%
373 }
374 \_newtoks\_famfrom
375 \_input fams-ini.opm
376 \_let\_famfile=\_undefined
377 \_famfrom={}
```

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

```
390 \_newtoks \_catalogsample
391 \_newtoks \_catalogmathsample
392 \_newtoks \_catalogonly
393 \_newtoks \_catalogexclude
394 \_catalogsample={ABCDabcd Qsty fi fl áéíóúüů řžč ÁÉÍÓÚ ŘŽČ 0123456789}
395
396 \_public \catalogonly \catalogexclude \catalogsample \catalogmathsample ;
```

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

- `\_defaultfontfeatures` – used for each font,
- `\_ffadded` – features added by `\setff`,
- `\_ffcolor` – features added by `\setfontcolor` (this is obsolette)
- `\_ffletterspace` – features added by `\setletterspace`,
- `\_ffwordspace` – features added by `\setwordspace`.

78

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

```
412 \_def \_fontfeatures{\_defaultfontfeatures\_ffadded\_ffcolor\_ffletterspace\_ffwordspace}
413 \_def \_defaultfontfeatures {+tlig;}
414 \_def \_ffadded{}
415 \_def \_ffcolor{}
416 \_def \_ffletterspace{}
417 \_def \_ffwordspace{}
```

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

```
424 \_def \_setff #1{%
425    \_ifx^#1^\_def\_ffadded{}\_else \_edef\_ffadded{\_ffadded #1;}\_fi
426 }
427 \_public \setff ;
```

`\setletterspace` is based on the special font features provided by `luaotfload` package. The `\setwordspace` recalculates the `\fontdimen2,3,4` of the font using the `\setwsp` macro which is used by the `\_fontselA` macro. It activates a dummy font feature `+Ws` too in order the font is reloded by the `\font` primitive (with independent `\fontdimen` registers). `\setfontcolor` is kept here only for backward compatibility but not recommended. Use real color switches and the `\transparency` instead.

```
441 \_def \_setfontcolor #1{%
442    \_edef\_tmp{\_calculatefontcolor{#1}}%
443    \_ifx\_tmp\_empty \_def\_ffcolor{}\_else \_edef\_ffcolor{color=\_tmp;}\_fi
444 }
445 \_def \_setletterspace #1{%
446    \_if^#1^\_def\_ffletterspace{}\_else \_edef\_ffletterspace{letterspace=#1;}\_fi
447 }
448 \_def \_setwordspace #1{%
449    \_if^#1^\_def\_setwsp##1{}\_def\_ffwordspace{}%
450    \_else \_def\_setwsp{\_setwspA#1/}\_def\_ffwordspace{+Ws;}\_fi
451 }
452 \_def\_setwsp #1{}
453 \_def\_setwspA #1{\_ifx/#1\_ea\_setwspB \_else\_afterfi{\_setwspC#1}\_fi}
454 \_def\_setwspB #1/#2/#3/#4{\_fontdimen2#4=#1\_fontdimen2#4%
455    \_fontdimen3#4=#2\_fontdimen3#4\_fontdimen4#4=#3\_fontdimen4#4}
456 \_def\_setwspC #1/{\_setwspB #1/#1/#1/}
457
458 \_def\_calculatefontcolor#1{\_trycs{_fc:#1}{#1}} % you can define more smart macro ...
459 \_sdef{_fc:red}{FF0000FF}    \_sdef{_fc:green}{00FF00FF} \_sdef{_fc:blue}{0000FFFF}
460 \_sdef{_fc:yellow}{FFFF00FF} \_sdef{_fc:cyan}{00FFFFFF}  \_sdef{_fc:magenta}{FF00FFFF}
461 \_sdef{_fc:white}{FFFFFFFF}  \_sdef{_fc:grey}{00000080}  \_sdef{_fc:lgrey}{00000025}
462 \_sdef{_fc:black}{} % ... you can declare more colors...
463
464 \_public \setfontcolor \setletterspace \setwordspace ;
```

`\_regoptsizes` ⟨*internal-template*⟩ ⟨*left-output*⟩`?`⟨*right-output*⟩ ⟨*resizing-data*⟩ prepares data for using by the `\_optname` ⟨*internal-template*⟩ macro. The data are saved to the `\_oz:`⟨*internal-template*⟩ macro. When the `\_optname` is expanded then the data are scanned by the macro `\_optnameA` ⟨*left-output*⟩`?`⟨*right-output*⟩ ⟨*mid-output*⟩ `<`⟨*size*⟩ in the loop.
`\_optfontalias` {⟨*template A*⟩}{⟨*template B*⟩} is defined as `\let\_oz:`⟨*templateA*⟩`=\_oz:`⟨*templateB*⟩.

```
477 \_def\_regoptsizes #1 #2?#3 #4*{\_sdef{_oz:#1}{#2?#3 #4* }}
478 \_def\_optname #1{\_ifcsname _oz:#1\_endcsname
479    \_ea\_ea\_ea \_optnameA \_csname _oz:#1\_ea\_endcsname
480    \_else \_failedoptname{#1}\_fi
481 }
482 \_def\_failedoptname #1{optname-fails:(#1)}
483 \_def\_optnameA #1?#2 #3 <#4 {\_ifx*#4#1#3#2\_else
484    \_ifdim\_optsize<#4pt #1#3#2\_optnameC
485    \_else \_afterfifi \_optnameA #1?#2 \_fi\_fi
486 }
487 \_def\_optnameC #1* {\_fi\_fi}
488 \_def\_afterfifi #1\_fi\_fi{\_fi\_fi #1}
489 \_def\_optfontalias #1#2{\_slet{_oz:#1}{_oz:#2}}
490
491 \_setfontsize {at10pt} % default font size
```

79

2.14 Preloaded fonts for math mode

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

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

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

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

math-preload.opm

```
 3 \_codedecl \normalmath {Math fonts CM + AMS preloaded <2022-07-22>} % preloaded in format
```

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

math-preload.opm

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

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

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

math-preload.opm

```
55 \_chardef\_bffam = 8
56 \_chardef\_bifam = 9
57 %\_chardef\_ttfam = 10
58 %\_chardef\_itfam = 11
59
60 \_protected\_def \_marm {\_fam0 }
61 \_protected\_def \_mabf {\_fam\_bffam}
62 \_protected\_def \_mait {\_fam1 }
63 \_protected\_def \_mabi {\_fam\_bifam}
64 \_protected\_def \_matt {}
65
```

```
66 \_protected\_def \_mit    {\_fam1 }
67 \_protected\_def \_cal    {\_fam2 }
68 \_protected\_def \_bbchar {\_fam5 }  % double stroked letters
69 \_protected\_def \_frak   {\_fam7 }  % fraktur
70 \_protected\_def \_script {\_fam6 }  % more extensive script than \cal
71
72 \_public \mit \cal \bbchar \frak \script ;
```

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

```
79 %% CM math fonts, optical sizes:
80
81 \_regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
82              cmmi10 11.1 cmmi12 *
83 \_regtfm cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
84 \_regtfm cmtex 0 cstex8 8.5 cstex9 9.5 cstex10 *
85 \_regtfm cmsy 0 cmsy5 5.5 cmsy6 6.5 cmsy7 7.5 cmsy8 8.5 cmsy9 9.5 cmsy10 *
86 \_regtfm cmbsy 0 cmbsy5 5.5 cmbsy6 6.5 cmbsy7 7.5 cmbsy8 8.5 cmbsy9 9.5 cmbsy10 *
87 \_regtfm cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
88 \_regtfm cmexb 0 cmexb10 *
89
90 \_regtfm cmr  0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
91              cmr10 11.1 cmr12 15 cmr17 *
92 \_regtfm cmbx 0 cmbx5 5.5 cmbx6 6.5 cmbx7 7.5 cmbx8 8.5 cmbx9 9.5
93              cmbx10 11.1 cmbx12 *
94 \_regtfm cmti 0 cmti7 7.5 cmti8 8.5 cmti9 9.5 cmti10 11.1 cmti12 *
95 \_regtfm cmtt 0 cmtt8 8.5 cmtt9 9.5 cmtt10 11.1 cmtt12 *
96
97 %% AMS math fonts, optical sizes:
98
99 \_regtfm msam 0 msam5 5.5 msam6 6.5 msam7 7.5 msam8 8.5 msam9 9.5 msam10 *
100 \_regtfm msbm 0 msbm5 5.5 msbm6 6.5 msbm7 7.5 msbm8 8.5 msbm9 9.5 msbm10 *
101
102 %% fraktur, rsfs, optical sizes:
103
104 \_regtfm eufm 0 eufm5 6 eufm7 8.5 eufm10 *
105 \_regtfm eufb 0 eufb5 6 eufb7 8.5 eufb10 *
106 \_regtfm rsfs 0 rsfs5 6 rsfs7 8.5 rsfs10 *
107
108 %% bf and bi sansserif math alternatives:
109
110 \_regtfm bfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
111          8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
112 \_regtfm bisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
113          8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *
114 \_regtfm bbfsans 0 ecsx0500 5.5 ecsx0600 6.5 ecsx0700 7.5 ecsx0800
115          8.5 ecsx0900 9.5 ecsx1000 11.1 ecsx1200 *
116 \_regtfm bbisans 0 ecso0500 5.5 ecso0600 6.5 ecso0700 7.5 ecso0800
117          8.5 ecso0900 9.5 ecso1000 11.1 ecso1200 *
```

_loadmathfamily ⟨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 default.

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

```
146 \_def\_corrmsize#1 {\_ptmunit=#1\_ptunit} % for corrections of sizes in diferent fonts
147
```

```
148  \_def\_loadmathfamily #1 #2 {%
149    \_edef\_optsizesave{\_the\_optsize}%
150    \_optsize=\_sizemtext    \_font\_mF=\_optfn{#2} at\_optsize \_textfont#1=\_mF
151    \_optsize=\_sizemscript  \_font\_mF=\_optfn{#2} at\_optsize \_scriptfont#1=\_mF
152    \_optsize=\_sizemsscript \_font\_mF=\_optfn{#2} at\_optsize \_scriptscriptfont#1=\_mF
153    \_optsize=\_optsizesave \_ptmunit=\_ptunit
154  }
155  \_def\_setmathfamily #1 #2{\_let\_mF=#2%
156    \_edef\_optsizesave{\_the\_optsize}%
157    \_optsize=\_sizemtext    \_fontlet#2=#2 at\_optsize \_textfont#1=#2%
158    \_optsize=\_sizemscript  \_fontlet#2=#2 at\_optsize \_scriptfont#1=#2%
159    \_optsize=\_sizemsscript \_fontlet#2=#2 at\_optsize \_scriptscriptfont#1=#2%
160    \_optsize=\_optsizesave \_ptmunit=\_ptunit \_let#2=\_mF
161  }
162  \_def\_setmathsizes[#1/#2/#3]{\_ptmunit=\_ptunit
163    \_def\_sizemtext{#1\_ptmunit}\_def\_sizemscript{#2\_ptmunit}%
164    \_def\_sizemsscript{#3\_ptmunit}%
165  }
166  \_newdimen\_ptunit   \_ptunit=1pt
167  \_newdimen\_ptmunit  \_ptmunit=1\_ptunit
168
169  \_public \setmathsizes \ptunit \ptmunit ;
```

_setmathparam⟨*luatex-param*⟩ {⟨*factor*⟩} sets ⟨*luatex-param*⟩ (like \Umathspaceafterscript) to values
dependent on **1em** of textfont, scriptfont, scriptscriptfont. The ⟨*factor*⟩ is scaling factor of mentioned **1em**.

```
178  \_def\_setmathparam#1#2{%
179    #1\_displaystyle      =#2\_fontdimen6\_textfont1
180    #1\_textstyle         =#2\_fontdimen6\_textfont1
181    #1\_scriptstyle       =#2\_fontdimen6\_scriptfont1
182    #1\_scriptscriptstyle =#2\_fontdimen6\_scriptscriptfont1
183  }
```

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.

_setmathparam\Umathspaceafterscript is used instead \scriptspace setting because LuaTeX ig-
nores \scriptspace in most cases. There is small difference from classical TeX: we set "scaled"
\Umathspaceafterscript dependent on textstyle, scriptstyle, etc. sizes. The _scriptspacefactor
is set to 0.05 which gives the same result as Plain TeX \scriptspace=0.5pt at 10 pt font size.

```
198  \_def\_setmathdimens{% PlainTeX sets these dimens for 10pt size only:
199    \_delimitershortfall=0.5\_fontdimen6\_textfont3
200    \_nulldelimiterspace=0.12\_fontdimen6\_textfont3
201    \_setmathparam\_Umathspaceafterscript \_scriptspacefactor
202    \_skewchar\_textfont1=127 \_skewchar\_scriptfont1=127
203    \_skewchar\_scriptscriptfont1=127
204    \_skewchar\_textfont2=48  \_skewchar\_scriptfont2=48
205    \_skewchar\_scriptscriptfont2=48
206    \_skewchar\_textfont6=127 \_skewchar\_scriptfont6=127
207    \_skewchar\_scriptscriptfont6=127
208  }
209  \_def\_scriptspacefactor{.05}
```

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

```
219  \_suppressfontnotfounderror=1
220  \_setmathsizes[10/7/5]
221  \_ifx\fontspreload\_relax \_else \_normalmath \_fi
222  \_suppressfontnotfounderror=0
```

2.15   Math macros

```
3  \_codedecl \sin {Math macros plus mathchardefs <2021-08-02>} % preloaded in format
```

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

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

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

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

math-macros.opm

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

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

math-macros.opm

```
55 \_mathcode`\^^@="2201 % \cdot
56 \_mathcode`\^^A="3223 % \downarrow
57 \_mathcode`\^^B="010B % \alpha
58 \_mathcode`\^^C="010C % \beta
59 \_mathcode`\^^D="225E % \land
60 \_mathcode`\^^E="023A % \lnot
61 \_mathcode`\^^F="3232 % \in
62 \_mathcode`\^^G="0119 % \pi
63 \_mathcode`\^^H="0115 % \lambda
64 \_mathcode`\^^I="010D % \gamma
65 \_mathcode`\^^J="010E % \delta
66 \_mathcode`\^^K="3222 % \uparrow
67 \_mathcode`\^^L="2206 % \pm
68 \_mathcode`\^^M="2208 % \oplus
69 \_mathcode`\^^N="0231 % \infty
70 \_mathcode`\^^O="0140 % \partial
71 \_mathcode`\^^P="321A % \subset
72 \_mathcode`\^^Q="321B % \supset
73 \_mathcode`\^^R="225C % \cap
74 \_mathcode`\^^S="225B % \cup
75 \_mathcode`\^^T="0238 % \forall
76 \_mathcode`\^^U="0239 % \exists
77 \_mathcode`\^^V="220A % \otimes
78 \_mathcode`\^^W="3224 % \leftrightarrow
79 \_mathcode`\^^X="3220 % \leftarrow
80 \_mathcode`\^^Y="3221 % \rightarrow
81 \_mathcode`\^^Z="8000 % \ne
82 \_mathcode`\^^[="2205 % \diamond
83 \_mathcode`\^^\="3214 % \le
84 \_mathcode`\^^]="3215 % \ge
85 \_mathcode`\^^^="3211 % \equiv
```

```
 86  \_mathcode`\^^_="225F % \lor
 87  \_mathcode`\ ="8000 % \space
 88  \_mathcode`\!="5021
 89  \_mathcode`\'="8000 % ^\prime
 90  \_mathcode`\(="4028
 91  \_mathcode`\)="5029
 92  \_mathcode`\*="2203 % \ast
 93  \_mathcode`\+="202B
 94  \_mathcode`\,="613B
 95  \_mathcode`\-="2200
 96  \_mathcode`\.="013A
 97  \_mathcode`\/="013D
 98  \_mathcode`\:="303A
 99  \_mathcode`\;="603B
100  \_mathcode`\<="313C
101  \_mathcode`\=="303D
102  \_mathcode`\>="313E
103  \_mathcode`\?="503F
104  \_mathcode`\[="405B
105  \_mathcode`\\="026E % \backslash
106  \_mathcode`\]="505D
107  \_mathcode`\_="8000 % math-active subscript
108  \_mathcode`\{="4266
109  \_mathcode`\|="026A
110  \_mathcode`\}="5267
111  \_mathcode`\^^?="1273 % \smallint
112
113  \_delcode`\(="028300
114  \_delcode`\)="029301
115  \_delcode`\[="05B302
116  \_delcode`\]="05D303
117  \_delcode`\<="26830A
118  \_delcode`\>="26930B
119  \_delcode`\/="02F30E
120  \_delcode`\|="26A30C
121  \_delcode`\\="26E30F
```

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

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

```
134  \_mathchardef\alpha="010B
135  \_mathchardef\beta="010C
136  \_mathchardef\gamma="010D
137  \_mathchardef\delta="010E
138  \_mathchardef\epsilon="010F
139  \_mathchardef\zeta="0110
140  \_mathchardef\eta="0111
141  \_mathchardef\theta="0112
142  \_mathchardef\iota="0113
143  \_mathchardef\kappa="0114
144  \_mathchardef\lambda="0115
145  \_mathchardef\mu="0116
146  \_mathchardef\nu="0117
147  \_mathchardef\xi="0118
148  \_mathchardef\pi="0119
```
. . . etc. (see math-macros.opm)

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

```
306  \_protected\_def\log {\_mathop{\_rm log}\_nolimits}
307  \_protected\_def\lg {\_mathop{\_rm lg}\_nolimits}
308  \_protected\_def\ln {\_mathop{\_rm ln}\_nolimits}
309  \_protected\_def\lim {\_mathop{\_rm lim}}
310  \_protected\_def\limsup {\_mathop{\_rm lim\_thinsk sup}}
```

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

\sp is an alternative for ^. The \sb alternative for _ was defined at line 27 of the file math-macros.opm.

```
348 \_let\_sp=^ \public \sp ;
349 % \sb=_ , defined at beginning of this file
350
351 \_def\_thinsk {\_mskip\_thinmuskip}
352 \_protected\_def\,{\_relax\_ifmmode \_thinsk \_else \_thinspace \_fi}
353 \_protected\_def\>{\_mskip\_medmuskip}    \let\_medsk   = \>
354 \_protected\_def\;{\_mskip\_thickmuskip}  \let\_thicksk = \;
355 \_protected\_def\!{\_mskip-\_thinmuskip}  \let\_thinneg = \!
356 %\_def\*{\discretionary{\thinspace\the\textfont2\char2}{}{}} % obsolete
```

Active \prime character is defined here.

```
362 {\_catcode`\'=\active \_gdef'{^\_bgroup\_primes}} % primes dance
363 \_def\_primes{\_prime\_isnextchar'{\_primesA}%
364                              {\_isnextchar^{\_primesB}{\_egroup}}}
365 \_def\_primesA #1{\_primes}
366 \_def\_primesB #1#2{#2\_egroup}
367 \_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.

```
376 %{\_catcode`\^^Z=\active \gdef^^Z{\not=}} % ^^Z is like \ne in math %obsolete
377
378 \_def\_scalebig#1#2{{\_left#1\_vbox to#2\_fontdimen6\_textfont1{}%
379                     \_kern-\_nulldelimiterspace\_right.}}
380 \_protected\_def\_big#1{\_scalebig{#1}{.85}}
381 \_protected\_def\_Big#1{\_scalebig{#1}{1.15}}
382 \_protected\_def\_bigg#1{\_scalebig{#1}{1.45}}
383 \_protected\_def\_Bigg#1{\_scalebig{#1}{1.75}}
384 \_public \big \Big \bigg \Bigg ;
385
386 \_protected\_def\_bigl{\_mathopen\_big}
387 \_protected\_def\_bigm{\_mathrel\_big}
388 \_protected\_def\_bigr{\_mathclose\_big}
```

```
389  \_protected\_def\_Bigl{\_mathopen\_Big}
390  \_protected\_def\_Bigm{\_mathrel\_Big}
391  \_protected\_def\_Bigr{\_mathclose\_Big}
392  \_protected\_def\_biggl{\_mathopen\_bigg}
393  \_protected\_def\_biggm{\_mathrel\_bigg}
394  \_protected\_def\_biggr{\_mathclose\_bigg}
395  \_protected\_def\_Biggl{\_mathopen\_Bigg}
396  \_protected\_def\_Biggm{\_mathrel\_Bigg}
397  \_protected\_def\_Biggr{\_mathclose\_Bigg}
398  \_public \bigl \bigm \bigr \Bigl \Bigm \Bigr \biggl \biggm \biggr \Biggl \Biggm \Biggr ;
```

Math relations defined by the \jointrel plain TeX macro:

```
404  \_protected\_def\_joinrel{\_mathrel{\_mkern-2.5mu}}  % -3mu in plainTeX
405  \_protected\_def\relbar{\_mathrel{\_smash-}} % \_smash, because - has the same height as +
406  \_protected\_def\Relbar{\_mathrel=}
407  \_mathchardef\lhook="312C
408  \_protected\_def\hookrightarrow{\_lhook\_joinrel\_rightarrow}
409  \_mathchardef\rhook="312D
410  \_protected\_def\hookleftarrow{\_leftarrow\_joinrel\_rhook}
411  \_protected\_def\bowtie{\_mathrel\_triangleright\_joinrel\_mathrel\_triangleleft}
412  \_protected\_def\models{\_mathrel|\_joinrel=}
413  \_protected\_def\Longrightarrow{\_Relbar\_joinrel\_Rightarrow}
414  \_protected\_def\longrightarrow{\_relbar\_joinrel\_rightarrow}
415  \_protected\_def\longleftarrow{\_leftarrow\_joinrel\_relbar}
416  \_protected\_def\Longleftarrow{\_Leftarrow\_joinrel\_Relbar}
417  \_protected\_def\longmapsto{\_mapstochar\_longrightarrow}
418  \_protected\_def\longleftrightarrow{\_leftarrow\_joinrel\_rightarrow}
419  \_protected\_def\Longleftrightarrow{\_Leftarrow\_joinrel\_Rightarrow}
420  \_protected\_def\iff{\_thicksk\_Longleftrightarrow\_thicksk}
421  \_private \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
422     \Relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
423     \longrightarrow \Longleftrightarrow ;
424  \_public \joinrel ;
```

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

```
430  \_mathchardef\_ldotp="613A % ldot as a punctuation mark
431  \_mathchardef\_cdotp="6201 % cdot as a punctuation mark
432  \_mathchardef\_colon="603A % colon as a punctuation mark
433  \_public \ldotp \cdotp \colon ;
434
435  \_protected\_def\_ldots{\_mathinner{\_ldotp\_ldotp\_ldotp}}
436  \_protected\_def\_cdots{\_mathinner{\_cdotp\_cdotp\_cdotp}}
437  \_protected\_def\_vdots{\_vbox{\_baselineskip=.4em \_lineskiplimit=\_zo
438     \_kern.6em \_hbox{.}\_hbox{.}\_hbox{.}}}
439  \_protected\_def\_ddots{\_mathinner{%
440     \_mkern1mu\_raise.7em\_vbox{\_kern.7em\_hbox{.}}\_mkern2mu
441     \_raise.4em\_hbox{.}\_mkern2mu\_raise.1em\_hbox{.}\_mkern1mu}}
442
443  \_public \ldots \cdots \vdots \ddots ;
```

\adots inspired by plain TeX

```
449  \_protected\_def\_adots{\_mathinner{%
450     \_mkern1mu\_raise.1em\_hbox{.}\_mkern2mu
451     \_raise.4em\_hbox{.}\_mkern2mu\_raise.7em\_vbox{\_kern.7em\_hbox{.}}\_mkern1mu}}
452
453  \_public \adots ;
```

Math accents (encoding dependent declarations).

```
459  \_protected\_def\acute{\_mathaccent"7013 }
460  \_protected\_def\grave{\_mathaccent"7012 }
461  \_protected\_def\ddot{\_mathaccent"707F }
462  \_protected\_def\tilde{\_mathaccent"707E }
463  \_protected\_def\bar{\_mathaccent"7016 }
464  \_protected\_def\breve{\_mathaccent"7015 }
465  \_protected\_def\check{\_mathaccent"7014 }
466  \_protected\_def\hat{\_mathaccent"705E }
```

```
467  \_protected\_def\vec{\_mathaccent"017E }
468  \_protected\_def\dot{\_mathaccent"705F }
469  \_protected\_def\widetilde{\_mathaccent"0365 }
470  \_protected\_def\widehat{\_mathaccent"0362 }
```

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

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

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

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

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

```
540  \_def\_mathpalette#1#2{\_mathchoice{#1\_displaystyle{#2}}%
541    {#1\_textstyle{#2}}{#1\_scriptstyle{#2}}{#1\_scriptscriptstyle{#2}}}
542  \_newbox\_rootbox
543  \_protected\_def\root#1\of{\_setbox\_rootbox
544    \_hbox{$\_math\_scriptscriptstyle{#1}$}\_mathpalette\_rootA}
```

87

```
545  \_def\_rootA#1#2{\_setbox0=\_hbox{$\_math#1\_sqrt{#2}$}\_dimen0=\_ht0
546    \_advance\_dimen0by-\_dp0
547    \_mkern5mu\_raise.6\_dimen0\_copy\_rootbox \_mkern-10mu\_box0 }
548  \_newifi\_ifvp \_newifi\_ifhp
549  \_protected\_def\_vphantom{\_vptrue\_hpfalse\_phant}
550  \_protected\_def\_hphantom{\_vpfalse\_hptrue\_phant}
551  \_protected\_def\_phantom{\_vptrue\_hptrue\_phant}
552  \_def\_phant{\_ifmmode\_def\_next{\_mathpalette\_mathphant}%
553    \_else\_let\_next=\_makephant\_fi\_next}
554  \_def\_makephant#1{\_setbox0\_hbox{#1}\_finphant}
555  \_def\_mathphant#1#2{\_setbox0=\_hbox{$\_math#1{#2}$}\_finphant}
556  \_def\_finphant{\_setbox2=\_null
557    \_ifvp \_ht2=\_ht0 \_dp2=\_dp0 \_fi
558    \_ifhp \_wd2=\_wd0 \_fi \_hbox{\_box2}}
559  \_def\_mathstrut{\_vphantom(}
560  \_protected\_def\_smash{\_relax % \_relax, in case this comes first in \halign
561    \_ifmmode\_def\_next{\_mathpalette\_mathsmash}\_else\_let\_next\_makesmash
562    \_fi\_next}
563  \_def\_makesmash#1{\_setbox0=\_hbox{#1}\_finsmash}
564  \_def\_mathsmash#1#2{\_setbox0=\_hbox{$\_math#1{#2}$}\_finsmash}
565  \_def\_finsmash{\_ht0=\_zo \_dp0=\_zo \_hbox{\_box0}}
566  \_public \mathpalette \vphantom \hphantom \phantom \mathstrut \smash ;
```

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

```
573  \_protected\_def\_cong{\_mathrel{\_mathpalette\_overeq\_sim}} % congruence sign
574  \_def\_overeq#1#2{\_lower.05em\_vbox{\_lineskiplimit\_maxdimen\_lineskip=-.05em
575    \_ialign{$\_math#1\_hfil##\_hfil$\_crcr#2\_crcr=\_crcr}}}
576  \_protected\_def\_notin{\_mathrel{\_mathpalette\_cancel\_in}}
577  \_def\_cancel#1#2{\_math\_ooalign{$\_hfil#1\_mkern1mu/\_hfil$\_crcr$#1#2$}}
578  \_protected\_def\_rightleftharpoons{\_mathrel{\_mathpalette\_rlhp{}}}
579  \_def\_rlhp#1{\_vcenter{\_math\_hbox{\_ooalign{\_raise.2em
580        \_hbox{$#1\_rightharpoonup$}\_crcr
581      $#1\_leftharpoondown$}}}}
582  \_protected\_def\_buildrel#1\over#2{\_mathrel{\_mathop{\_kern\_zo #2}\_limits^{#1}}}
583  \_protected\_def\_doteq{\_buildrel\_textstyle.\over=}
584  \_private \in \sim ;
585  \_public \cong \notin \rightleftharpoons \buildrel \doteq ;
586
587  \_protected\_def\_bmod{\_nonscript\_mskip-\_medmuskip\_mkern5mu
588    \_mathbin{\_rm mod}\_penalty900\_mkern5mu\_nonscript\_mskip-\_medmuskip}
589  \_protected\_def\_pmod#1{\_allowbreak\_mkern18mu({\_rm mod}\_thinsk\_thinsk#1)}
590  \_public \bmod \pmod ;
```

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

```
600  \_protected\_def\_matrix#1{\_null\_thinsk
601      \_edef\_tmpa{\_the\_numexpr \_mathstyle/4\_relax}% 0 0 1 1 1 1 2 2
602      \_vcenter{\_matrixbaselines\_math
603      \_ialign{\_the\_lmfil$\_matrixstyle##$\_hfil&&\_quad\_the\_lmfil$\_matrixstyle##$\_hfil\_crcr
604        \_mathstrut\_crcr\_noalign{\_kern-\_baselineskip}
605        #1\_crcr\_mathstrut\_crcr\_noalign{\_kern-\_baselineskip}}}\_thinsk}
606
607  \_def\_matrixbaselines{\_normalbaselines \_def\_matrixstyle{}%
608    \_let\_matrixbaselines=\_relax % \matrix inside matrix does not change size again
609    \_ifcase\_tmpa \_or
610        \_baselineskip=.7\_baselineskip \_def\_quad {\_hskip.7em\_relax}%
611        \_let\_matrixstyle=\_scriptstyle
612      \_or
613        \_baselineskip=.5\_baselineskip \_def\_quad {\_hskip.5em\_relax}%
614        \_let\_matrixstyle=\_scriptscriptstyle
615      \_fi
616  }
617  \_protected\_def\_pmatrix#1{\_left(\_matrix{#1}\_right)}
618
619  \_public \matrix \pmatrix ;
```

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

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

The `\eqalign` macro behaves like in Plain TeX by default. It creates the `\vcenter` in the math mode. The content is two column `\halign` with right-aligned left column and left-aligned right column. The table items are in `\displaystyle` and the `\baselineskip` is advanced by `\jot` (3pt in plain TeX). It follows from the default settings of `\eqlines` and `\eqstyle` parameters.

In OpTeX, this macro is more flexible. See section 4.4 in the Typesetting Math with OpTeX. The `\baselineskip` value is set by the `\eqlines` parameter and math style by the `\eqstyle` parameter.

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

```
668 \_long\_def\_eqalign#1{\_null\_thinsk\_vcenter{\_the\_eqlines\_math
669     \_ialign{&\_hfil$\_the\_eqstyle{##}$&$\_the\_eqstyle{{}##}$\_hfil
670         &\_hskip.5\_eqspace\_hfil$\_the\_eqstyle{##}$\_hskip.5\_eqspace\_hfil
671         \_crcr#1\_crcr}}\_thinsk}
672
673 \_public \_eqalign ;
```

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

OpTeX offers `\dislaylines to`⟨*dimen*⟩`{`⟨*formula*⟩`\cr`⟨*formula*⟩`\cr`...⟨*formula*⟩`}` as an alternative case of usage `\displaylines`. See section 4.3 in the Typesetting Math with OpTeX. The centered formulas are in `\vcenter` in this case, so lines cannot be broken into more pages, but this case enables to use `\eqno` or `\leqno` or `\eqmark`.

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

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

```
710 \_def\_openup{\_afterassignment\_openupA\_dimen0=}
711 \_def\_openupA{\_advance\_lineskip by\_dimen0
712     \_advance\_baselineskip by\_dimen0
```

```
713    \_advance\_lineskiplimit by\_dimen0 }
714 \_newifi\_ifdtop
715 \_def\_display{\_global\_dtoptrue\_openup\_jot\_math
716    \_everycr{\_noalign{\_ifdtop \_global\_dtopfalse \_ifdim\_prevdepth>-1000pt
717        \_vskip-\_lineskiplimit \_vskip\_normallineskiplimit \_fi
718        \_else \_penalty\_interdisplaylinepenalty \_fi}}}
719 \_def\_elign{\_tabskip=\_zoskip\_everycr{}} % restore inside \_display
720 \_long\_def\_eqalignno#1{\_display \_tabskip=\_centering
721    \_halign to\_displaywidth{\_hfil$\_elign\_displaystyle{##}$\_tabskip=\_zoskip
722      &$\_elign\_displaystyle{{}##}$\_hfil\_tabskip\_centering
723      &\_hbox to\_zo{\_hss$\_elign##$}\_tabskip\_zoskip\_crcr
724      #1\_crcr}}
725 \_long\_def\_leqalignno#1{\_display \_tabskip=\_centering
726    \_halign to\_displaywidth{\_hfil$\_elign\_displaystyle{##}$\_tabskip=\_zoskip
727      &$\_elign\_displaystyle{{}##}$\_hfil\_tabskip=\_centering
728      &\_kern-\_displaywidth\_hbox to\_zo{$\_elign##$\_hss}\_tabskip\_displaywidth\_crcr
729      #1\_crcr}}
730 \_public \_openup \eqalignno \leqalignno ;
```

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

```
737 \_def\_amsafam{4} \_def\_amsbfam{5}
738
739 \_mathchardef \boxdot     "2\_amsafam 00
740 \_mathchardef \boxplus    "2\_amsafam 01
741 \_mathchardef \boxtimes   "2\_amsafam 02
742 \_mathchardef \square     "0\_amsafam 03
743 \_mathchardef \blacksquare   "0\_amsafam 04
744 \_mathchardef \centerdot   "2\_amsafam 05
745 \_mathchardef \lozenge    "0\_amsafam 06
746 \_mathchardef \blacklozenge   "0\_amsafam 07
747 \_mathchardef \circlearrowright   "3\_amsafam 08
748 \_mathchardef \circlearrowleft    "3\_amsafam 09
749 \_mathchardef \rightleftharpoons   "3\_amsafam 0A
750 \_mathchardef \leftrightharpoons   "3\_amsafam 0B
751 \_mathchardef \boxminus   "2\_amsafam 0C
```
. . . etc. (see math-macros.opm)

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

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

```
986 \_mathchardef \_notchar   "3236
987
988 \_protected\_def \_not#1{%
989    \_ifx #1<\_nless \_else
990    \_ifx #1>\_ngtr \_else
991    \_edef\_tmpn{\_csstring#1}%
992    \_ifcsname _not\_tmpn\_endcsname \_csname _not\_tmpn\_endcsname
993    \_else \_ifcsname _n\_tmpn\_endcsname \_csname _n\_tmpn\_endcsname
994    \_else \_mathrel{\_mathord{\_notchar}\_mathord{#1}}%
995    \_fi \_fi \_fi \_fi}
996 \_private
997    \nleq \ngeq \nless \ngtr \nprec \nsucc \nleqslant \ngeqslant \npreceq
998    \nsucceq \nleqq \ngeqq \nsim \ncong \nsubseteqq \nsupseteqq \nsubseteq
999    \nsupseteq \nparallel \nmid \nshortmid \nshortparallel \nvdash \nVdash
1000    \nvDash \nVDash \ntrianglerighteq \ntrianglelefteq \ntriangleleft
1001    \ntriangleright \nleftarrow \nrightarrow \nLeftarrow \nRightarrow
1002    \nLeftrightarrow \nleftrightarrow \nexists ;
1003 \_public \not ;
```

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

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

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

```
1023  \_newcount\_stylenum
1024  \_def\_mathstyles#1{{\_mathchoice{\_stylenum0 #1}{\_stylenum1 #1}%
1025                      {\_stylenum2 #1}{\_stylenum3 #1}}}
1026  \_def\_dobystyle#1#2#3#4{\_ifcase\_stylenum#1\_or#2\_or#3\_or#4\_fi}
1027  \_def\_currstyle{\_dobystyle\_displaystyle\_textstyle\_scriptstyle\_scriptscriptstyle}
1028  \_public \mathstyles \dobystyle \currstyle \stylenum ;
```

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

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

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

```
1042  \_def\_cramped{\_ifcase\_numexpr(\_mathstyle+1)/2\_relax\_or
1043    \_crampeddisplaystyle \_or \_crampedtextstyle \_or
1044    \_crampedscriptstyle \_or \_crampedscriptscriptstyle \_fi
1045  }
1046  \_public \cramped ;
```

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

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

```
1059  \_def\_mathbox#1{{\_mathstyles{\_hbox{%
1060    \_ifnum\_stylenum<2 \_everymath{\_currstyle}%
1061    \_else
1062        \_ifnum\_stylenum=2 \_def\_textmff{ssty=1;}\_fi
1063        \_ifnum\_stylenum=3 \_def\_textmff{ssty=2;}\_def\_scriptmff{ssty=2;}\_fi
1064        \_typoscale[\_dobystyle{}{}{700}{500}/]\_fi #1}}}%
1065  }
1066  \_public \mathbox ;
```

2.16 Unicode-math fonts

The **\loadmath** {⟨*Unicode-math font*⟩} macro loads math fonts and redefines all default math-codes using \input unimath-codes.opm. If Unicode-math font is loaded then **_mathloadingfalse** is set, so the new Unicode-math font isn't loaded until **\doloadmath** is used.

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

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

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

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

```
\addUmathfont \famname {[⟨normal-font⟩]}{⟨ffeatures⟩} {[⟨bold-font⟩]}{⟨ffeatures⟩} {⟨factor⟩}
```

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

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

declares `latinmodern-math` as main math font (its bold variant is "faked bold") and the additional math font `\xits` is `XITSMath-Regular` for normal printing and `XITSMath-Bold` for bold printing.

All characters used in math formula is printed from main math font by default. But you can redeclare a list of characters for printing from additional font by `\resetmathchars \famname` ⟨*list of sequences*⟩ `;`. For example:

```
\resetmathchars \xits \stareq \triangleq \veeeq \wedgeq ;
```

sets the characters `\stareq, \triangleq, \veeeq, \wedgeq` from the `\xits` additional font. You can see all possible sequences by `\input print-unimath.opm`. If you want to use whole math alphabet from additional math font, use `\addto\selector{\fam\famname}`, for example `\addto\cal{\fam\xits}` declares all `\cal` characters from the `\xits` font loaded by `\addUmathfont`.

2.16.1   Unicode-math macros preloaded in the format

```
 3 \_codedecl \loadmath {Unicode Math fonts <2022-07-22>} % preloaded in format
```

`\loadmath` {⟨*Unicode-math font*⟩} loads the 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 the loaded font on the terminal,
- redefine all encoding dependent setting by `\input unimath-codes.opm`,
- protect new loading by setting `\_ifmathloading` to false.

`\noloadmath` disallows Unicode-math loading by `\_mathloadingfalse`.
`\doloadmath` allows Unicode-math loading by `\_mathloadingtrue`.

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

```
48 \_def\_loadboldmath#1#2\to #3{%
49    \_def\_tmp{#3}\_ifx\_unimathfont\_tmp % do work only if #3 is loaded as normal Math
50    \_isfont{#1}\_iffalse
51       \_opwarning{Bold-Math font "#1" not found, skipped...}
52    \_else
53       \_def\_unimathboldfont{#1}%
54       \_wterm {MATH-FONT: "#1" -- unicode math bold prepared.}%
55    \_fi\_fi}
56
57 \_public \loadboldmath ;
```

The Unicode version of the `\normalmath` and `\boldmath` macros are defined here as `\_normalunimath` and `\_boldunimath` macros. They are using `\_setunimathdimens` in a similar sense as `\_setmathdimens`. You can combine more fonts if you register them to another math families (5, 6, 7, etc.) in the `\normalmath` macro.

The default value of `\_normalunimath` shows a combination of base Unicode-math font with 8bit Math font at family 4. See definition of `\script` macro where `\fam4` is used.

```
73  \_def\_normalunimath{%
74      \_loadumathfamily 1 {\_unimathfont}{} % Base font
75      \_loadmathfamily  4 rsfs              % script
76      \_setunimathdimens
77  }%
78  \_def\_boldunimath{%
79      \_ifx\_unimathboldfont \_undefined
80          \_loadumathfamily 1 {\_unimathfont}{embolden=1.7;} % Base faked bold
81      \_else
82          \_loadumathfamily 1 {\_unimathboldfont}{} % Base real bold font
83      \_fi
84      \_loadmathfamily  4 rsfs              % script
85      \_setunimathdimens
86  }%
87  \_def\_setunimathdimens{% PlainTeX sets these dimens for 10pt size only:
88      \_delimitershortfall=0.5\_fontdimen6\_textfont1
89      \_nulldelimiterspace=0.12\_fontdimen6\_textfont1
90      \_setmathparam\_Umathspaceafterscript \_scriptspacefactor
91      \_setbox0=\_hbox{\_everymath{}$\_fam1\_displaystyle{0\_atop0}$}%
92      \_Umathfractiondelsize\_displaystyle = \_dimexpr(\_ht0-\_Umathaxis\_displaystyle)*2\_relax
93      \_setbox0=\_box\_voidbox
94  }
```

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

```
\loadmath{[xitsmath-regular]}
\def\_boldmath{%
    \_loadumathfamily 1 {[xitsmath-bold]}{} % Base font
    \_loadmathfamily 4  rsfs % script
    \_loadumathfamily 5 {[xitsmath-regular]}{}
    \_def\|{\_Udelimiter 0 5 "02016 }%       % norm delimiter from family 5
    \_setmathdimens
}
```

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

`\_mparams`⟨*number*⟩ inserts additional font feature `nomathparam` if the ⟨*number*⟩ of the family is greater than 3. LuaTeX sets math parameters (thickness of fraction rules etc., see section 7.4 in LuaTeX documentation) repeatedly from loaded math fonts if `nomathparam` is not given. We want to load these parameters only from fonts at families 0–3 (and actually we are using only family 1 as main math font). The `\_corrmsize` ⟨*factor*⟩⟨*space*⟩ can be used just before `\_loadumathfamily`, see section 2.14 for more information.

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

```
133  \_def\_umathname#1#2{"#1:\_mfontfeatures#2"}
134  \_def\_mfontfeatures{mode=base;script=math;}
135
136  \_def\_loadumathfamily #1 #2#3 {%
137    \_font\_mF=\_umathname{#2}{\_textmff   \_mparams{#1}#3} at\_sizemtext \_textfont        #1=\_mF
138    \_font\_mF=\_umathname{#2}{\_scriptmff \_mparams{#1}#3} at\_sizemtext \_scriptfont       #1=\_mF
139    \_font\_mF=\_umathname{#2}{\_sscriptmff\_mparams{#1}#3} at\_sizemtext \_scriptscriptfont#1=\_mF
140    \_ptmunit=\_ptunit
```

```
141  }
142  \_def\_textmff    {ssty=0;mathsize=1;}
143  \_def\_scriptmff  {ssty=1;mathsize=2;}
144  \_def\_sscriptmff {ssty=2;mathsize=3;}
145  \_def\_mparams#1{\_ifnum#1>3 nomathparam;\_fi}
```

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

_umathrange {⟨from⟩-⟨to⟩}⟨class⟩⟨family⟩\⟨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, they cause that the math alphabets are not continuously spread out in the table; I mean that the designers were under the influence of drugs when they created this part of the Unicode table). The ⟨from⟩-⟨to⟩ clause includes characters 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.

```
176  \_newcount\_umathnumA  \_newcount\_umathnumB
177
178  \_def\_umathcorr#1#2{\_ea#1\_ea{\_the#2}}
179  \_def\_umathprepare#1{\_def\_umathscanholes##1[#1]##2##3\_relax{##2}}
180  \_def\_umathvalue#1{\_ea\_umathscanholes\_umathcharholes[#1]{#1}\_relax}
181
182  \_def\_umathcharholes{% holes in math alphabets:
183     [119893]{"210E}[119965]{"212C}[119968]{"2130}[119969]{"2131}%
184     [119971]{"210B}[119972]{"2110}[119975]{"2112}[119976]{"2133}[119981]{"211B}%
185     [119994]{"212F}[119996]{"210A}[120004]{"2134}%
186     [120070]{"212D}[120075]{"210C}[120076]{"2111}[120085]{"211C}[120093]{"2128}%
187     [120122]{"2102}[120127]{"210D}[120133]{"2115}[120135]{"2119}
188     [120136]{"211A}[120137]{"211D}[120145]{"2124}%
189  }
190  \_def\_umathrange#1#2#3#4{\_umathnumB=#4\_def\_tmp{#2 #3 }\_umathrangeA#1}
191  \_def\_umathrangeA#1-#2{\_umathnumA=`#1\_relax
192     \_loop
193        \_umathcorr\_umathprepare\_umathnumB
194        \_Umathcode \_umathnumA = \_tmp \_umathcorr\_umathvalue{\_umathnumB}
195        \_ifnum\_umathnumA<`#2\_relax
196           \_advance\_umathnumA by1 \_advance\_umathnumB by1
197     \_repeat
198  }
199  \_def\_umathrangeGREEK{\_umathrange{^^^^0391-^^^^03a9}}
200  \_def\_umathrangegreek{\_umathrange{^^^^03b1-^^^^03d6}}
201  \_def\_greekdef#1{\_ifx#1\_relax \_else
202     \_begingroup \_lccode`X=\_umathnumB \_lowercase{\_endgroup \_def#1{X}}%
203     \_advance\_umathnumB by 1
204     \_ea\_greekdef \_fi
205  }
```

\addUmathfont ⟨fam⟩ {[⟨normal-font⟩]}{⟨ffeatures⟩} {[⟨bold-font⟩]}{⟨ffeatures⟩} {⟨factor⟩} allocates new ⟨fam⟩ using \newfam and adds loading this font to the \normalmath and \boldmath macros. Note that allocationos using \newfam starts from 43 because numbers 1–42 are reserved for direct usage without \newfam.

```
215  \_def\_addUmathfont #1#2#3#4#5#6{% #1: fam (will be set), #2#3: normal font, #4#5: bold font
216     \_ifx\_ncharrmA\_undefined \_errmessage{basic Unicode math font must be loaded first}%
217     \_else \_isfont{#2}\_iffalse \_opwarning{font #2 is unavailable}%
218     \_else
219        \_newfam#1\_relax
220        \_global\_addto\_normalmath{\_corrmsize#6 \_loadumathfamily #1 {#2}{#3} }%
221        \_ifx\_relax#4\_relax
222           \_global\_addto\_boldmath{\_corrmsize#6 \_loadumathfamily #1 {#2}{embolden=1.7;} }%
223        \_else
224           \_global\_addto\_boldmath{\_corrmsize#6 \_loadumathfamily #1 {#4}{#5} }%
225        \_fi
```

```
226        \_normalmath
227        \_wterm{add-MATH-FONT: #1=\the#1, "#2", \ifx"#4"\else bold: "#4"\fi}%
228     \_fi \_fi
229 }
```

\resetmathchars ⟨*fam*⟩ ⟨*list of sequences*⟩ ; saves ⟨*fam*⟩ to **_mafam** and applies **_resetmathcharsX** to each sequence from ⟨*list of sequences*⟩. The \directlua chunk prints the 8-digits hexadecimal code of the sequence followed by the sequence itself. The digts are scanned by **_resetmathcharsY** and new \Umathchardef and \Umathcode is declared. Note that the used primitives have the syntax:

> \Umathchardef ⟨*sequence*⟩ ⟨*math class*⟩ ⟨*math family*⟩ ⟨*slot number*⟩
> \Umathcode ⟨*code*⟩ ⟨*math class*⟩ ⟨*math family*⟩ ⟨*slot number*⟩

```
245 \_def\_resetmathchars #1{\_chardef\_mafam=#1\_relax \_xargs \_resetmathcharX}
246 \_def\_resetmathcharX#1{\_ea\_resetmathcharY
247     \_directlua{tex.print(string.format("\_pcent08X", \_the#1))}#1}
248 \_def\_resetmathcharY#1#2#3#4#5#6#7#8#9{% #9 is given \math-sequence
249     \_Umathchardef #9\_numexpr"#3/2\_relax \_mafam "#4#5#6#7#8
250     \_Umathcode "#4#5#6#7#8=\_numexpr"#3/2\_relax \_mafam "#4#5#6#7#8
251 }
252 \_public \addUmathfont \resetmathchars ;
```

2.16.2 Macros and codes set when \loadmath is processed firstly

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

```
3 \_codedecl \_ncharrmA {Uni math codes <2022-07-20>} % preloaded on demand by \loadmath
```

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

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

The math alphabets are declared here using the **_umathrange**{⟨*range*⟩}⟨*class*⟩⟨*family*⟩⟨*starting-code*⟩ macro.

```
28 \_chardef\_ncharrmA=`A        \_chardef\_ncharrma=`a
29 \_chardef\_ncharbfA="1D400     \_chardef\_ncharbfa="1D41A
30 \_chardef\_ncharitA="1D434     \_chardef\_ncharita="1D44E
31 \_chardef\_ncharbiA="1D468     \_chardef\_ncharbia="1D482
32 \_chardef\_ncharclA="1D49C     \_chardef\_ncharcla="1D4B6
33 \_chardef\_ncharbcA="1D4D0     \_chardef\_ncharbca="1D4EA
34 \_chardef\_ncharfrA="1D504     \_chardef\_ncharfra="1D51E
35 \_chardef\_ncharbrA="1D56C     \_chardef\_ncharbra="1D586
36 \_chardef\_ncharbbA="1D538     \_chardef\_ncharbba="1D552
37 \_chardef\_ncharsnA="1D5A0     \_chardef\_ncharsna="1D5BA
38 \_chardef\_ncharbsA="1D5D4     \_chardef\_ncharbsa="1D5EE
39 \_chardef\_ncharsiA="1D608     \_chardef\_ncharsia="1D622
40 \_chardef\_ncharsxA="1D63C     \_chardef\_ncharsxa="1D656
41 \_chardef\_nchartA="1D670      \_chardef\_nchartta="1D68A
42
43 \_protected\_def\_rmvariables     {\_umathrange{A-Z}71\_ncharrmA \_umathrange{a-z}71\_ncharrma}
44 \_protected\_def\_bfvariables     {\_umathrange{A-Z}71\_ncharbfA \_umathrange{a-z}71\_ncharbfa}
45 \_protected\_def\_itvariables     {\_umathrange{A-Z}71\_ncharitA \_umathrange{a-z}71\_ncharita}
46 \_protected\_def\_bivariables     {\_umathrange{A-Z}71\_ncharbiA \_umathrange{a-z}71\_ncharbia}
47 \_protected\_def\_calvariables    {\_umathrange{A-Z}71\_ncharclA \_umathrange{a-z}71\_ncharcla}
```

```
48 \_protected\_def\_bcalvariables   {\_umathrange{A-Z}71\_ncharbcA \_umathrange{a-z}71\_ncharbca}
49 \_protected\_def\_frakvariables   {\_umathrange{A-Z}71\_ncharfrA \_umathrange{a-z}71\_ncharfra}
50 \_protected\_def\_bfrakvariables  {\_umathrange{A-Z}71\_ncharbrA \_umathrange{a-z}71\_ncharbra}
51 \_protected\_def\_bbvariables     {\_umathrange{A-Z}71\_ncharbbA \_umathrange{a-z}71\_ncharbba}
52 \_protected\_def\_sansvariables   {\_umathrange{A-Z}71\_ncharsnA \_umathrange{a-z}71\_ncharsna}
53 \_protected\_def\_bsansvariables  {\_umathrange{A-Z}71\_ncharbsA \_umathrange{a-z}71\_ncharbsa}
54 \_protected\_def\_isansvariables  {\_umathrange{A-Z}71\_ncharsiA \_umathrange{a-z}71\_ncharsia}
55 \_protected\_def\_bisansvariables {\_umathrange{A-Z}71\_ncharsxA \_umathrange{a-z}71\_ncharsxa}
56 \_protected\_def\_ttvariables     {\_umathrange{A-Z}71\_ncharttA \_umathrange{a-z}71\_nchartta}
57
58 \_chardef\_greekrmA="0391   \_chardef\_greekrma="03B1
59 \_chardef\_greekbfA="1D6A8  \_chardef\_greekbfa="1D6C2
60 \_chardef\_greekitA="1D6E2  \_chardef\_greekita="1D6FC
61 \_chardef\_greekbiA="1D71C  \_chardef\_greekbia="1D736
62 \_chardef\_greeksnA="1D756  \_chardef\_greeksna="1D770
63 \_chardef\_greeksiA="1D790  \_chardef\_greeksia="1D7AA
64
65 \_protected\_def\_itgreek    {\_umathrangegreek71\_greekita}
66 \_protected\_def\_rmgreek    {\_umathrangegreek71\_greekrma}
67 \_protected\_def\_bfgreek    {\_umathrangegreek71\_greekbfa}
68 \_protected\_def\_bigreek    {\_umathrangegreek71\_greekbia}
69 \_protected\_def\_bsansgreek {\_umathrangegreek71\_greeksna}
70 \_protected\_def\_bisansgreek{\_umathrangegreek71\_greeksia}
71 \_protected\_def\_itGreek    {\_umathrangeGREEK71\_greekitA \_setnablait}
72 \_protected\_def\_rmGreek    {\_umathrangeGREEK71\_greekrmA \_setnablarm}
73 \_protected\_def\_bfGreek    {\_umathrangeGREEK71\_greekbfA \_setnablabf}
74 \_protected\_def\_biGreek    {\_umathrangeGREEK71\_greekbiA \_setnablabi}
75 \_protected\_def\_bsansGreek {\_umathrangeGREEK71\_greeksnA \_setnablabsans}
76 \_protected\_def\_bisansGreek{\_umathrangeGREEK71\_greeksiA \_setnablabisans}
```

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

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

Digits are configured like math alphabets.

```
96  \_chardef\_digitrm0=`0
97  \_chardef\_digitbf0="1D7CE
98  \_chardef\_digitbb0="1D7D8
99  \_chardef\_digitsn0="1D7E2
100 \_chardef\_digitbs0="1D7EC
101 \_chardef\_digittt0="1D7F6
102
103 \_protected\_def\_rmdigits    {\_umathrange{0-9}71\_digitrm0}
104 \_protected\_def\_bfdigits    {\_umathrange{0-9}71\_digitbf0}
105 \_protected\_def\_bbdigits    {\_umathrange{0-9}71\_digitbb0}
106 \_protected\_def\_sansdigits  {\_umathrange{0-9}71\_digitsn0}
107 \_protected\_def\_bsansdigits {\_umathrange{0-9}71\_digitbs0}
108 \_protected\_def\_ttdigits    {\_umathrange{0-9}71\_digittt0}
```

The math alphabets `\cal`, `\bbchar`, `\frak`, `\script` are re-defined here. The `\_marm`, `\_mabf`, `\_mait`, `\_mabi`, `\_matt` used in `\rm`, `\bf`, `\it`, `\bi` are re-defined too.

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

```
\_protected\_def\_mabf {\_inmath{\_bfvariables\_bfgreek\_bfGreek\_bfdigits}}
\_protected\_def\_mabi {\_inmath{\_bivariables\_bigreek\_bfGreek\_bfdigits}}
```

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

```
123 \_protected\_def\_inmath#1{\_relax \_ifmmode#1\_fi} % to keep off \loop processing in text mode
124
```

```
125  % You can redefine these macros to follow your wishes.
126  % For example, you need upright lowercase greek letters, you don't need
127  % \bf and \bi behave as sans serif in math, ...
128
129  \_protected\_def\_marm {\_inmath{\_rmvariables \_rmdigits}}
130  \_protected\_def\_mait {\_inmath{\_itvariables \_itGreek}}
131  \_protected\_def\_mabf {\_inmath{\_bsansvariables \_bsansgreek \_bsansGreek \_bsansdigits}}
132  \_protected\_def\_mabi {\_inmath{\_bisansvariables \_bisansgreek \_bisansGreek \_bsansdigits}}
133  \_protected\_def\_matt {\_inmath{\_ttvariables \_ttdigits}}
134  \_protected\_def\_bbchar  {\_bbvariables \_bbdigits}
135  \_protected\_def\_cal     {\_calvariables}
136  \_protected\_def\_frak    {\_frakvariables}
137  \_protected\_def\_misans  {\_isansvariables \_sansdigits}
138  \_protected\_def\_mbisans {\_bisansvariables \_bisansgreek \_bsansGreek \_bsansdigits}
139  \_protected\_def\_script  {\_rmvariables \_fam4 }
140  \_protected\_def\_mit     {\_itvariables \_rmdigits \_itgreek \_rmGreek }
141
142  \_public \bbchar \cal \frak \misans \mbisans \script \mit ;
```

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

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

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

```
152  \_begingroup  % \input mathclass.opm  (which is a copy of MathClass.txt):
153     \_long\_def\_p#1;#2 {\_ifx^#2^\_else
154       \_edef\_tmp{\_csname _c:#2\_endcsname}\_if\_relax\_tmp\_else \_pA#1....\_end#2\_fi
155       \_ea\_p \_fi }
156     \_def\_pA#1..#2..#3\_end#4{%
157       \_ifx\_relax#2\_relax \_pset{"#1}{#4}\_else \_fornum "#1.."#2\_do{\_pset{##1}{#4}}\_fi
158     }
159     \_sdef{_c:L}{1}\_sdef{_c:B}{2}\_sdef{_c:V}{2}\_sdef{_c:R}{3}\_sdef{_c:N}{0}\_sdef{_c:U}{0}
160     \_sdef{_c:F}{0}\_sdef{_c:O}{4}\_sdef{_c:C}{5}\_sdef{_c:P}{6}\_sdef{_c:A}{7}
161     \_def\_pset#1#2{\_Umathcode#1=\_tmp\_space 1 #1\_relax
162       \_if#2O\_Udelcode#1=1 #1\_relax\_fi
163       \_if#2C\_Udelcode#1=1 #1\_relax\_fi
164       \_if#2F\_Udelcode#1=1 #1\_relax\_fi
165     }
166     \_catcode`#=14 \_everyeof={;{} } \_def\par{}
167     \_globaldefs=1 \_ea \_p \_input mathclass.opm
168  \_endgroup
```

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

```
70  \UnicodeMathSymbol{"00393}{\mupGamma                }{\mathalpha}{capital gamma, greek}%
71  \UnicodeMathSymbol{"00394}{\mupDelta                }{\mathalpha}{capital delta, greek}%
72  \UnicodeMathSymbol{"00395}{\mupEpsilon               }{\mathalpha}{capital epsilon, greek}%
73  \UnicodeMathSymbol{"00396}{\mupZeta                 }{\mathalpha}{capital zeta, greek}%
74  \UnicodeMathSymbol{"00397}{\mupEta                  }{\mathalpha}{capital eta, greek}%
75  \UnicodeMathSymbol{"00398}{\mupTheta                }{\mathalpha}{capital theta, greek}%
76  \UnicodeMathSymbol{"00399}{\mupIota                 }{\mathalpha}{capital iota, greek}%
77  \UnicodeMathSymbol{"0039A}{\mupKappa                }{\mathalpha}{capital kappa, greek}%
```

```
78  \UnicodeMathSymbol{"0039B}{\mupLambda            }{\mathalpha}{capital lambda, greek}%
79  \UnicodeMathSymbol{"0039C}{\mupMu                }{\mathalpha}{capital mu, greek}%
80  \UnicodeMathSymbol{"0039D}{\mupNu                }{\mathalpha}{capital nu, greek}%
81  \UnicodeMathSymbol{"0039E}{\mupXi                }{\mathalpha}{capital xi, greek}%
82  \UnicodeMathSymbol{"0039F}{\mupOmicron           }{\mathalpha}{capital omicron, greek}%
83  \UnicodeMathSymbol{"003A0}{\mupPi                }{\mathalpha}{capital pi, greek}%
84  \UnicodeMathSymbol{"003A1}{\mupRho               }{\mathalpha}{capital rho, greek}%
85  \UnicodeMathSymbol{"003A3}{\mupSigma             }{\mathalpha}{capital sigma, greek}%
```

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

```
177  \_begingroup  % \input unimath-table.opm (it is a copy of unicode-math-table.tex):
178      \_def\UnicodeMathSymbol #1#2#3#4{%
179         \_ifnum#1=\_Umathcodenum#1 % the code isn't set by mathclass.opm
180            \_Umathchardef#2=0 1 #1 \_Umathcode#1=0 1 #1
181         \_else \_Umathcharnumdef#2=\_Umathcodenum#1 \_fi
182         \_ifx#3\_mathopen   \_def#2{\_Udelimiter 4 1 #1 }\_fi
183         \_ifx#3\_mathclose  \_def#2{\_Udelimiter 5 1 #1 }\_fi
184         \_ifx#3\_mathaccent \_def#2{\_Umathaccent fixed 7 1 #1 }\_fi
185      }
186      \_globaldefs=1 \_input unimath-table.opm
187  \_endgroup
```

Many special characters must be declared with care...

```
193  \_global\_Udelcode`<=1 "027E8 % these characters have different meaning
194  \_global\_Udelcode`>=1 "027E9 % as normal and as delimiter
195
196  \_mit % default math alphabets setting
197
198  % hyphen character is transformed to minus:
199  \_Umathcode `- = 2 1 "2212
200
201  % mathclass defines : as Punct, plain.tex as Rel, we keep mathclass,
202  % i.e. there is difference from plain.tex, you can use $f:A\to B$.
203
204  % mathclas defines ! as Ord, plain.tex as Close
205  \_Umathcode `! = 5 1 `!  % keep plain.tex declaration
206  \_Umathchardef \mathexclam = 5 1 `!
207  % mathclas defines ? as Punct, plain.tex as Close
208  \_Umathcode `? = 5 1 `?  % keep plain.tex declaration
209  \_Umathchardef \mathquestion = 5 1 `?
210
211  \_Umathcode `* = 2 1 "02217  % equivalent to \ast, like in plain TeX
212
213  \_Umathcode "03A2 = 7 1 "03F4 % \varTheta
214
215  \_Umathcode `© = 0 1 `© % usage $\copyright$ can be seen in old documents
216
217  \_protected\_def \_sqrt        {\_Uradical 1 "0221A }
218  \_protected\_def \_cuberoot    {\_Uradical 1 "0221B }
219  \_protected\_def \_fourthroot {\_Uradical 1 "0221C }
220
221  \_def \nabla {^^^^2207} % \nabla behaves as uppercase Gereek letter, see \_setnabla
222
223  \_public \sqrt \cuberoot \fourthroot ;
224
225  \_def\_intwithnolimits#1#2 {\_ifx#1\_relax \_else
226     \_ea\_let\_csname\_csstring#1op\_endcsname=#1%
227     \_ea\_def\_ea #1\_ea{\_csname\_csstring#1op\_endcsname \_nolimits}%
228     \_bgroup \_lccode`\~=#2 \_lowercase{\_egroup \_mathcode`~="8000 \_let ~=#1}%
229     \_ea \_intwithnolimits \_fi
230  }
231  \_intwithnolimits \int "0222B \iint "0222C \iiint "0222D
232     \oint "0222E \oiint "0222F \oiiint "02230
233     \intclockwise "02231 \varointclockwise "02232 \ointctrclockwise "02233
234     \sumint "02A0B \iiiint "02A0C \intbar "02A0D \intBar "02A0E \fint "02A0F
235     \pointint "02A15 \sqint "02A16 \intlarhk "02A17 \intx "02A18
236     \intcap "02A19 \intcup "02A1A \upint "02A1B \lowint "02A1C \_relax "0
237
```

```
238  \_protected\_def \vert  {\_Udelimiter 0 1 "07C }
239  \_protected\_def \Vert  {\_Udelimiter 0 1 "02016 }
240  \_protected\_def \Vvert {\_Udelimiter 0 1 "02980 }
241
242  \_protected\_def \_overbrace    #1{\_mathop {\_Umathaccent   7 1 "023DE{#1}}\_limits}
243  \_protected\_def \_underbrace   #1{\_mathop {\_Umathaccent bottom 7 1 "023DF{#1}}\_limits}
244  \_protected\_def \_overparen    #1{\_mathop {\_Umathaccent   7 1 "023DC{#1}}\_limits}
245  \_protected\_def \_underparen   #1{\_mathop {\_Umathaccent bottom 7 1 "023DD{#1}}\_limits}
246  \_protected\_def \_overbracket  #1{\_mathop {\_Umathaccent   7 1 "023B4{#1}}\_limits}
247  \_protected\_def \_underbracket #1{\_mathop {\_Umathaccent bottom 7 1 "023B5{#1}}\_limits}
248
249  \_public \overbrace \underbrace \overparen \underparen \overbracket \underbracket ;
250
251  \_protected\_def \widehat            {\_Umathaccent 7 1 "00302 }
252  \_protected\_def \widetilde          {\_Umathaccent 7 1 "00303 }
253  \_protected\_def \overleftharpoon    {\_Umathaccent 7 1 "020D0 }
254  \_protected\_def \overrightharpoon   {\_Umathaccent 7 1 "020D1 }
255  \_protected\_def \overleftarrow      {\_Umathaccent 7 1 "020D6 }
256  \_protected\_def \overrightarrow     {\_Umathaccent 7 1 "020D7 }
257  \_protected\_def \overleftrightarrow {\_Umathaccent 7 1 "020E1 }
258
259  \_protected\_def \wideoverbar  {\_Umathaccent 7 1 "00305 }
260  \_protected\_def \widebreve    {\_Umathaccent 7 1 "00306 }
261  \_protected\_def \widecheck    {\_Umathaccent 7 1 "0030C }
262  \_protected\_def \wideutilde   {\_Umathaccent bottom 7 1 "00330 }
263  \_protected\_def \mathunderbar {\_Umathaccent bottom 7 1 "00332 }
264  \_protected\_def \underleftrightarrow   {\_Umathaccent bottom 7 1 "0034D }
265  \_protected\_def \widebridgeabove       {\_Umathaccent 7 1 "020E9 }
266  \_protected\_def \underrightharpoondown {\_Umathaccent bottom 7 1 "020EC }
267  \_protected\_def \underleftharpoondown  {\_Umathaccent bottom 7 1 "020ED }
268  \_protected\_def \underleftarrow        {\_Umathaccent bottom 7 1 "020EE }
269  \_protected\_def \underrightarrow       {\_Umathaccent bottom 7 1 "020EF }
270
271  \_mathchardef\ldotp="612E
272  \_let\|=\Vert
273  \_mathcode`\_="8000
274
275  \_global\_Umathcode    "22EF        = 0 1 "22EF % mathclass says that it is Rel
276  \_global\_Umathcode    "002E        = 0 1 "002E % mathclass says that dot is Punct
277  \_global\_Umathchardef \unicodecdots = 0 1 "22EF
278
279  \_global\_Umathcode `/ = 0 1 `/  % mathclass says that / is Bin, Plain TeX says that it is Ord.
280
281  % compressed dots in S and SS styles (usable in \matrix when it is in T, S and SS style)
282  \_protected\_def \vdots {\_relax \_ifnum \_mathstyle>3 \_unicodevdots \_else \_vdots \_fi}
283  \_protected\_def \ddots {\_relax \_ifnum \_mathstyle>3 \_unicodeddots \_else \_ddots \_fi}
284  \_protected\_def \adots {\_relax \_ifnum \_mathstyle>3 \_unicodeadots \_else \_adots \_fi}
285
286  % Unicode superscripts (²) and subscripts as simple macros with \mathcode"8000
287  \_bgroup
288      \_def\_tmp#1#2{\_global\_mathcode#1="8000 \_lccode`\~=#1 \_lowercase{\_gdef~}{#2}}
289      \_fornum 0..1 \_do {\_tmp{"207#1}{{^#1}}}
290      \_tmp{"B2}{{^2}}\_tmp{"B3}{{^3}}
291      \_fornum 4..9 \_do {\_tmp{"207#1}{{^#1}}}
292      \_fornum 0..9 \_do {\_tmp{"208#1}{{_#1}}}
293  \_egroup
```

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

```
300  \_let \setminus=\smallsetminus
301  \_let \diamond=\smwhtdiamond
302  \_let \colon=\mathcolon
303  \_let \bullet=\smblkcircle
304  \_let \circ=\vysmwhtcircle
305  \_let \bigcirc=\mdlgwhtcircle
306  \_let \to=\rightarrow
307  \_let \le=\leq
308  \_let \ge=\geq
```

```
309  \_let \neq=\ne
310  \_protected\_def \triangle {\mathord{\bigtriangleup}}
311  \_let \emptyset=\varnothing
312  \_let \hbar=\hslash
313  \_let \land=\wedge
314  \_let \lor=\vee
315  \_let \owns=\ni
316  \_let \gets=\leftarrow
317  \_let \mathring=\ocirc
318  \_let \lnot=\neg
319  \_let \longdivisionsign=\longdivision
320  \_let \backepsilon=\upbackepsilon
321  \_let \eth=\matheth
322  \_let \dbkarow=\dbkarrow
323  \_let \drbkarow=\drbkarrow
324  \_let \hksearow=\hksearrow
325  \_let \hkswarow=\hkswarrow
326  \_let \square=\mdlgwhtsquare
327  \_let \blacksquare=\mdlgblksquare
328
329  \_let \upalpha=\mupalpha
330  \_let \upbeta=\mupbeta
331  \_let \upgamma=\mupgamma
332  \_let \updelta=\mupdelta
333  \_let \upepsilon=\mupvarepsilon
334  \_let \upvarepsilon=\mupvarepsilon
335  \_let \upzeta=\mupzeta
336  \_let \upeta=\mupeta
337  \_let \uptheta=\muptheta
338  \_let \upiota=\mupiota
339  \_let \upkappa=\mupkappa
340  \_let \uplambda=\muplambda
341  \_let \upmu=\mupmu
342  \_let \upnu=\mupnu
343  \_let \upxi=\mupxi
344  \_let \upomicron=\mupomicron
345  \_let \uppi=\muppi
346  \_let \uprho=\muprho
347  \_let \upvarrho=\mupvarrho
348  \_let \upvarsigma=\mupvarsigma
349  \_let \upsigma=\mupsigma
350  \_let \uptau=\muptau
351  \_let \upupsilon=\mupupsilon
352  \_let \upvarphi=\mupvarphi
353  \_let \upchi=\mupchi
354  \_let \uppsi=\muppsi
355  \_let \upomega=\mupomega
356  \_let \upvartheta=\mupvartheta
357  \_let \upphi=\mupphi
358  \_let \upvarpi=\mupvarpi
```

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

```
366  \_protected\_def\_not#1{%
367    \_trycs{_not!\_csstring#1}{\_mathrel\_mathstyles{%
368       \_setbox0=\_hbox{\_math$\_currstyle#1$}%
369       \_hbox to\_wd0{\_hss$\_currstyle/$\_hss}\_kern-\_wd0 \_box0
370  }}}
371  \_def\_negationof #1#2{\_ea\_let \_csname _not!\_csstring#1\_endcsname =#2}
372
373  \_negationof =          \neq
374  \_negationof <          \nless
375  \_negationof >          \ngtr
376  \_negationof \gets      \nleftarrow
377  \_negationof \simeq     \nsime
378  \_negationof \equal     \ne
379  \_negationof \le        \nleq
380  \_negationof \ge        \ngeq
```

```
381  \_negationof \greater  \ngtr
382  \_negationof \forksnot \forks
383  \_negationof \in        \notin
384  \_negationof \mid       \nmid
385  \_negationof \cong      \ncong
386  \_negationof \leftarrow \nleftarrow
387  \_negationof \rightarrow \nrightarrow
388  \_negationof \leftrightarrow \nleftrightarrow
389  \_negationof \Leftarrow \nLeftarrow
390  \_negationof \Leftrightarrow \nLeftrightarrow
391  \_negationof \Rightarrow \nRightarrow
392  \_negationof \exists   \nexists
393  \_negationof \ni       \nni
394  \_negationof \paralel \nparalel
395  \_negationof \sim      \nsim
396  \_negationof \approx   \napprox
397  \_negationof \equiv    \nequiv
398  \_negationof \asymp    \nasymp
399  \_negationof \lesssim \nlesssim
400  \_negationof \ngtrsim \ngtrsim
401  \_negationof \lessgtr \nlessgtr
402  \_negationof \gtrless \ngtrless
403  \_negationof \prec     \nprec
404  \_negationof \succ     \nsucc
405  \_negationof \subset   \nsubset
406  \_negationof \supset   \nsupset
407  \_negationof \subseteq \nsubseteq
408  \_negationof \supseteq \nsupseteq
409  \_negationof \vdash    \nvdash
410  \_negationof \vDash    \nvDash
411  \_negationof \Vdash   \nVdash
412  \_negationof \VDash   \nVDash
413  \_negationof \preccurlyeq \npreccurlyeq
414  \_negationof \succcurlyeq \nsucccurlyeq
415  \_negationof \sqsubseteq \nsqsubseteq
416  \_negationof \sqsupseteq \nsqsupseteq
417  \_negationof \vartriangleleft \nvartriangleleft
418  \_negationof \vartriangleright \nvartriangleright
419  \_negationof \trianglelefteq \ntrianglelefteq
420  \_negationof \trianglerighteq \ntrianglerighteq
421  \_negationof \vinfty \nvinfty
422
423  \_public \not ;
```

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

```
431  \_private
432     \ldotp \cdotp \bullet \triangleleft \triangleright \mapstochar \rightarrow
433     \prime \lhook \rightarrow \leftarrow \rhook \triangleright \triangleleft
434     \rbrace \lbrace \Relbar \Rightarrow \relbar \rightarrow \Leftarrow \mapstochar
435     \longrightarrow \Longleftrightarrow \unicodevdots \unicodeddots \unicodeadots ;
```

2.16.3 More Unicode-math examples

Example of using additional math font is in section 5.3 in the optex-math.pdf documentation

You can combine more Unicode math fonts in single formula simply by the \addUmathfont macro, see OpTeX trick 0030.

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

2.16.4 Printing all Unicode math slots in used math font

This file can be used for testing your Unicode-math font and/or for printing TeX sequences which can be used in math.

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

```
  3  \_codedecl \_undefined {Printing Unicode-math table \string<2020-06-08>}

  4

  5  \_ifx\_ncharrmA\_undefined \_opwarning{No Unicode math font loaded, printing ignored}
  6      \_endinput \_fi

  7

  8  \_begingroup
  9      \_def\UnicodeMathSymbol#1#2#3#4{%
 10          \_ifnum#1>"10000 \_endinput \_else \_printmathsymbol{#1}{#2}{#3}{#4}\_fi
 11      }
 12      \_def\UnicodeMathSymbolA#1#2#3#4{%
 13          \_ifnum#1>"10000 \_printmathsymbol{#1}{#2}{#3}{#4}\_fi
 14      }
 15      \_def\_printmathsymbol#1#2#3#4{%
 16          \_hbox{\_hbox to2em{$#2{}$\_hss}\_hbox to3em
 17              {\small\_printop#3\_hss}{\_tt\_string#2\_trycs{_eq:\_string#2}{}}}
 18      }
 19      \_def\_eq#1#2{\_sdef{_eq:\_string#2}{=\_string#1}}
 20      \_eq \diamond\smwhtdiamond \_eq \bullet\smblkcircle \_eq \circ\vysmwhtcircle
 21      \_eq \bigcirc\mdlgwhtcircle \_eq \to\rightarrow \_eq \le\leq
 22      \_eq \ge\geq \_eq \neq\ne \_eq \emptyset\varnothing \_eq \hbar\hslash
 23      \_eq \land\wedge \_eq \lor\vee \_eq \owns\ni \_eq \gets\leftarrow
 24      \_eq \mathring\ocirc \_eq \lnot\neg \_eq \backepsilon\upbackepsilon
 25      \_eq \eth\matheth \_eq \dbkarow\dbkarrow \_eq \drbkarow\drbkarrow
 26      \_eq \hksearow\hksearrow  \_eq \hkswarow\hkswarrow

 27

 28      \_tracinglostchars=0
 29      \_fontdef\small{\_setfontsize{at5pt}\_rm}
 30      \_def\_printop{\_def\mathop{Op}}
 31      \_def\mathalpha{Alph}\_def\mathord{Ord}\_def\mathbin{Bin}\_def\mathrel{Rel}
 32      \_def\mathopen{Open}\_def\mathclose{Close}\_def\mathpunct{Punct}\_def\mathfence{Fence}
 33      \_def\mathaccent{Acc}\_def\mathaccentwide{Accw}\_def\mathbotaccentwide{AccBw}
 34      \_def\mathbotaccent{AccB}\_def\mathaccentoverlay{AccO}
 35      \_def\mathover{Over}\_def\mathunder{Under}
 36      \_typosize[7.5/9]\_normalmath \_everymath={}

 37

 38      Codes U+00000 \_dots\ U+10000
 39      \_begmulti 3
 40          \_input unimath-table.opm
 41      \_endmulti

 42

 43      \_medskip\_goodbreak
 44      Codes U+10001 \_dots\ U+1EEF1  \_let\UnicodeMathSymbol=\UnicodeMathSymbolA
 45      \_begmulti 4
 46          \_input unimath-table.opm
 47      \_endmulti
 48  \_endgroup
```

2.17 Scaling fonts in document (high-level macros)

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

```
  3  \_codedecl \typosize {Font managing macros from OPmac <2022-02-22>} % preloaded in format
```

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

```
 15  \_protected\_def \_typosize [#1/#2]{%
 16      \_textfontsize{#1}\_mathfontsize{#1}\_setbaselineskip{#2}%
 17      \_setmainvalues \_ignorespaces
 18  }
 19  \_protected\_def \_textfontsize #1{\_if$#1$\_else \_setfontsize{at#1\_ptunit}\_fi}

 20

 21  \_def \_mathfontsize #1{\_if$#1$\_else
 22      \_tmpdim=#1\_ptunit
 23      \_edef\_sizemtext{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
 24      \_tmpdim=0.7\_tmpdim
```

```
25      \_edef\_sizemscript{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
26      \_tmpdim=#1\_ptunit \_tmpdim=0.5\_tmpdim
27      \_edef\_sizemsscript{\_ea\_ignorept \_the\_tmpdim \_ptmunit}%
28      \_fi
29  }
30  \_public \typosize ;
```

\typoscale [⟨*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.

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

_setbaselineskip {⟨*baselineskip*⟩} sets new \baselineskip and more values of registers which are dependent on the ⟨*baselineskip*⟩ including the \strutbox.

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

_setmainvalues sets the current font size and \baselineskip values to the \mainfosize and \mainbaselineskip registers and loads fonts at given sizes. It redefines itself as _setmainvaluesL to set the main values only first. The _setmainvaluesL does only fonts loading.
\scalemain returns to these values if they were set. Else they are set to 10/12 pt.
\mfontsrule gives the rule how math fonts are loaded when \typosize or \typoscale are used. The value of \mfontsrule can be:

- 0: no math fonts are loaded. User must use \normalmath or \boldmath explicitly.
- 1: _normalmath is run if \typosize/\typoscale are used first or they are run at outer group level. No \everymath/\everydisplay are set in this case. If \typosize/\typoscale are run repeatedly in a group then _normalmath is run only when math formula occurs. This is done using \everymath/\everydisplay and _setmathfonts. \mfontsrule=1 is default.
- 2: _normalmath is run whenever \typosize/\typoscale are used. \everymath/\everydisplay registers are untouched.

```
99  \_newskip   \_mainbaselineskip   \_mainbaselineskip=0pt \_relax
100 \_newdimen  \_mainfosize         \_mainfosize=0pt
101 \_newcount  \_mfontsrule         \_mfontsrule=1
102
103 \_def\_setmainvalues {%
104     \_mainbaselineskip=\_baselineskip
105     \_mainfosize=\_optsize
106     \_topskip=\_mainfosize \_splittopskip=\_topskip
107     \_ifmmode \_else \_rm \_fi                % load and initialize \rm variant
108     \_ifnum \_mfontsrule>0 \_normalmath \_fi  % load math fonts first
109     \_let \_setmainvalues =\_setmainvaluesL
```

```
110 }
111 \_def\_setmainvaluesL {\_relax \_ifmmode \_else \_rm \_fi % load text font
112     \_ifcase \_mfontsrule                          % load math fonts
113     \_or \_ifnum\_currentgrouplevel=0 \_normalmath
114         \_else \_everymath={\_setmathfonts}\_everydisplay={\_normalmath}%
115             \_let\_runboldmath=\_relax \_fi
116     \_or \_normalmath \_fi}
117 \_def\_scalemain {%
118     \_ifdim \_mainfosize=\_zo
119         \_mainfosize=10pt  \_mainbaselineskip=12pt
120         \_let \_setmainvalues=\_setmainvaluesL
121     \_fi
122     \_optsize=\_mainfosize  \_baselineskip=\_mainbaselineskip
123 }
124 \_public \scalemain \mainfosize \mainbaselineskip \mfontsrule ;
```

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

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

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

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

\em keeps the weight of the current variant and switches roman ↔ italic. It adds the italic correction
by the _additcorr and _afteritcorr macros. The second does not add italic correction if the next
character is dot or comma.

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

The \boldify macro does \let\rm\bf, \let\it\bi and \let\normalmath=\boldmath. All following
text will be in bold. If should be used after \typosize or \typoscale macros.
The internal _runboldmath macro runs _boldmath immediatelly if no delay of the math font loading
is set by _setmainvaluesL.
The \rm, \it in math mode must keep its original meaning.

```
189 \_protected\_def \_boldify {%
190     \_let \_setmainvalues=\_setmainvaluesL
191     \_let\it =\_bi \_let\rm =\_bf \_let\_normalmath=\_boldmath \_bf
```

```
192    \_runboldmath
193    \_ifx\_ncharrmA\_undefined \_protected\_addto\rm{\_fam0 }\_protected\_addto\it{\_fam1 }%
194    \_else \_protected\_def\rm {\_fmodbf \_fontsel \_marm}%
195          \_protected\_def\it {\_fmodbi \_fontsel \_mait}%
196    \_fi
197 }
198 \_def\_runboldmath{\_boldmath}
199
200 \_public \em \boldify ;
```

We need to use a font selector for default pagination. Because we don't know what default font size will be selected by the user, we use this `\_rmfixed` macro. It sets the \rm font from the default font size (declared by first `\typosize` command and redefines itself be only the font switch for the next pages.

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

2.18 Output routine

The output routine `\_optexoutput` is similar as in plain 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 `\topinsert`s),
    - `\box255` with completed vertical material from main vertical mode,
    - `\_footnoterule` and `\footins` box if nonempty (from `\fnote`, `\footnote`),
    - `\pgbottomskip` (default is 0 pt).
  - footline box by `\_makefootline`, if the `\footline` is nonempty
- `\_endoutput` which does:
  - increments `\pageno` using `\advancepageno`
  - runs output routine repeatedly if `\dosupereject` is activated.

```
3 \_codedecl \nopagenumbers {Output routine <2021-03-07>} % preloaded in format
```

`\_optexoutput` is the default output routine. You can create another...
The `\_preshipout`⟨destination box number⟩⟨box specification⟩ used here behaves similarly like `\setbox` but it does not only copy the box contents but adds the color literals depending on used attributes. It is defined using lua code, see section 2.39.

```
13 \_output={\_optexoutput}
14 \_def \_optexoutput{\_begoutput \_preshipout0\_completepage \_shipout\_box0 \_endoutput}
```

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

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

The `\_regmark` tokens list includes accumulated #2 from the `\regmacro`. Logos and other macros are re-defined here (locally) for their usage in headlines or footlines.

```
30 \_def \_begoutput{\_incr\_gpageno
31    \_immediate\_wref\_Xpage{{\_the\_gpageno}{\_folio}}%
32    \_setxhsize \_prepoffsets \_the\_regmark}
33 \_def \_endoutput{\_advancepageno
34    {\_globaldefs=1 \_the\_nextpages \_nextpages={}}%
35    \_ifnum\_outputpenalty>-20000 \_else\_dosupereject\_fi
36 }
37 \_def \_prepoffsets {}
```

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

```
51 \_newdimen \_xhsize
52 \_def\_setxhsize {\_global\_xhsize=\_hsize \_global\_let\_setxhsize=\_relax}
```

\gpageno counts pages from one in the whole document

```
58 \_newcount\_gpageno
59 \_public \gpageno ;
```

The _completepage is similar to what plain TeX does in its output routine. New is only _backgroundbox. It is \vbox with zero height with its contents (from \pgbackground) extended down. It is shifted directly to the left-upper corner of the paper.
The _resetattrs used here means that all newly created texts in output routine (texts used in headline, footline) have default color and no transparency.

```
71 \_def\_completepage{\_vbox{%
72     \_resetattrs
73     \_istoksempty \_pgbackground
74       \_iffalse \_backgroundbox{\_the\_pgbackground}\_nointerlineskip \_fi
75     \_makeheadline
76     \_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
77     \_makefootline}%
78 }
79 \_def \_backgroundbox #1{\_moveleft\_hoffset\_vbox to\_zo{\_kern-\_voffset #1\_vss}}
```

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

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

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

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

The _pagecontents is similar as in plain TeX. The only differnece is that the _pagedest is inserted at the top of _pagecontents.
The _footnoterule is defined here.

```
109 \_def\_pagecontents{\_pagedest % destination of the page
110   \_ifvoid\_topins \_else \_unvbox\_topins\_fi
111   \_dimen0=\_dp255 \_unvbox255 % open up \box255
112   \_ifvoid\_footins \_else % footnote info is present
113     \_vskip\_skip\_footins
114     \_footnoterule \_unvbox\_footins\_fi
115   \_kern-\_dimen0 \_vskip \_pgbottomskip
116 }
117 \_def \_pagedest {{\_def\_destheight{25pt}\_dest[pg:\_the\_gpageno]}}
118 \_def \_footnoterule {\_kern-3pt \_hrule width 2truein \_kern 2.6pt }
```

106

`\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.

```
129  \_countdef\_pageno=0 \_pageno=1 % first page is number 1
130  \_def \_folio {\_ifnum\_pageno<0 \_romannumeral-\_pageno \_else \_number\_pageno \_fi}
131  \_def \_nopagenumbers {\_footline={}}
132  \_def \_advancepageno {%
133    \_ifnum\_pageno<0 \_decr\_pageno \_else \_incr\_pageno \_fi
134  } % increase |pageno|
135  \_def \_raggedbottom {\_topskip=\_dimexpr\_topskip plus60pt \_pgbottomskip=0pt plus1fil\_relax}
136  \_def \_normalbottom {\_topskip=\_dimexpr\_topskip \_pgbottomskip=0pt\_relax}
137
138  \_public \pageno \folio \nopagenumbers \advancepageno \raggedbottom \normalbottom ;
```

Macros for footnotes are the same as in plain TeX. There is only one difference: `\vfootnote` is implemented as `\_opfootnote` with empty parameter `#1`. This parameter should do local settings inside the `\footins` group and it does it when `\fnote` macro is used.

The `\_opfootnote` nor `\vfootnote` don't take the footnote text as a parameter. This is due to a user can do catcode settings (like inline verbatim) in the footnote text. This idea is adapted from plain TeX. The `\footnote` and `\footstrut` is defined as in plain TeX.

```
151  \_newinsert\_footins
152  \_def \_footnote #1{\_let\_osf=\_empty % parameter #2 (the text) is read later
153    \_ifhmode \_edef\_osf{\_spacefactor\_the\_spacefactor}\/\_fi
154    #1\_osf\_vfootnote{#1}}
155  \_def\_vfootnote{\_opfootnote{}}
156  \_def \_opfootnote #1#2{\_insert\_footins\_bgroup
157    \_interlinepenalty=\_interfootnotelinepenalty
158    \_leftskip=\_zo \_rightskip=\_zo \_spaceskip=\_zo \_xspaceskip=\_zo \_relax
159    \_resetattrs
160    #1\_relax % local settings used by \fnote macro
161    \_splittopskip=\_ht\_strutbox % top baseline for broken footnotes
162    \_splitmaxdepth=\_dp\_strutbox \_floatingpenalty=20000
163    \_textindent{#2}\_footstrut
164    \_isnextchar \_bgroup
165      {\_bgroup \_aftergroup\_vfootA \_afterassignment\_ignorespaces \_let\_next=}{\_vfootB}%
166  }
167  \_def\_vfootA{\_unskip\_strut\_egroup}
168  \_def\_vfootB #1{#1\_unskip\_strut\_egroup}
169  \_def \_footstrut {\_vbox to\_splittopskip{}}
170  \_skip\_footins=\_bigskipamount % space added when footnote is present
171  \_count\_footins=1000 % footnote magnification factor (1 to 1)
172  \_dimen\_footins=8in % maximum footnotes per page
173  \_public
174    \footins \footnote \vfootnote \footstrut ;
```

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

```
182  \_newinsert\_topins
183  \_newifi\_ifupage \_newifi\_ifumid
184  \_def \_topinsert {\_umidfalse \_upagefalse \_oins}
185  \_def \_midinsert {\_umidtrue \_oins}
186  \_def \_pageinsert {\_umidfalse \_upagetrue \_oins}
187  \_skip\_topins=\_zoskip % no space added when a topinsert is present
188  \_count\_topins=1000 % magnification factor (1 to 1)
189  \_dimen\_topins=\_maxdimen % no limit per page
190  \_def \_oins {\_par \_begingroup\_setbox0=\_vbox\_bgroup\_resetattrs} % start a \_vbox
191  \_def \_endinsert {\_par\_egroup % finish the \_vbox
192    \_ifumid \_dimen0=\_ht0 \_advance\_dimen0 by\_dp0 \_advance\_dimen0 by\_baselineskip
193      \_advance\_dimen0 by\_pagetotal \_advance\_dimen0 by-\_pageshrink
194      \_ifdim\_dimen0>\_pagegoal \_umidfalse \_upagefalse \_fi \_fi
195    \_ifumid \_bigskip \_box0 \_bigbreak
196    \_else \_insert \_topins {\_penalty100 % floating insertion
197      \_splittopskip=0pt
198      \_splitmaxdepth=\_maxdimen \_floatingpenalty=0
199      \_ifupage \_dimen0=\_dp0
200      \_vbox to\_vsize {\_unvbox0 \_kern-\_dimen0}% depth is zero
```

```
201     \_else \_box0 \_nobreak \_bigskip \_fi}\_fi\_endgroup}
202
203 \_public \topins \topinsert \midinsert \pageinsert \endinsert ;
```

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

```
210 \_def \_draft {\_pgbackground={\_draftbox{\_draftfont DRAFT}}%
211     \_fontdef\_draftfont{\_setfontsize{at10pt}\_bf}%
212     \_global\_let\_draftfont=\_draftfont
213 }
214 \_def \_draftbox #1{\_setbox0=\_hbox{\_setgreycolor{.8}#1}%
215     \_kern.5\_vsize \_kern\_voffset \_kern4.5\_wd0
216     \_hbox to0pt{\_kern.5\_xhsize \_kern\_hoffset \_kern-2\_wd0
217     \_pdfsave \_pdfrotate{55}\_pdfscale{10}{10}%
218     \_hbox to0pt{\_box0\_hss}%
219     \_pdfrestore
220     \_hss}%
221 }
222 \_public \draft ;
```

2.19   Margins

The \margins macro is documented in the section 1.2.1.

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

\margins/⟨pg⟩ ⟨fmt⟩ (⟨left⟩,⟨right⟩,⟨top⟩,⟨bot⟩)⟨unit⟩ takes its parameters, does calculation and sets \hoffset, \voffset, \hsize and \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.

```
 13 \_newdimen\_pgwidth  \_newdimen\_pgheight  \_pgwidth=0pt
 14 \_newdimen\_shiftoffset
 15
 16 \_def\_margins/#1 #2 (#3,#4,#5,#6)#7 {\_def\_tmp{#7}%
 17     \_ifx\_tmp\_empty
 18        \_opwarning{\_string\_margins: missing unit, mm inserted}\_def\_tmp{mm}\_fi
 19     \_setpagedimens #2 % setting \_pgwidth, \_pgheight
 20     \_ifdim\_pgwidth=0pt \_else
 21        \_hoffset=0pt \_voffset=0pt
 22        \_if$#3$\_if$#4$\_hoffset =\_dimexpr (\_pgwidth -\_hsize)/2 \_relax
 23              \_else  \_hoffset =\_dimexpr \_pgwidth -\_hsize - #4\_tmp \_relax % only right margin
 24              \_fi
 25        \_else  \_if$#4$\_hoffset = #3\_tmp \_relax  % only left margin
 26              \_else  \_hsize =\_dimexpr \_pgwidth - #3\_tmp - #4\_tmp \_relax % left+right margin
 27                      \_hoffset = #3\_tmp \_relax
 28                      \_xhsize =\_hsize \_setxhsize % \_xhsize used by \output routine
 29        \_fi\_fi
 30        \_if$#5$\_if$#6$\_voffset =\_dimexpr (\_pgheight -\_vsize)/2 \_relax
 31              \_else  \_voffset =\_dimexpr \_pgheight -\_vsize - #6\_tmp \_relax % only bottom margin
 32              \_fi
 33        \_else  \_if$#6$\_voffset = #5\_tmp \_relax  % only top margin
 34              \_else  \_vsize=\_dimexpr \_pgheight - #5\_tmp - #6\_tmp \_relax % top+bottom margin
 35                      \_voffset = #5\_tmp \_relax
 36        \_fi\_fi
 37        \_if 1#1\_shiftoffset=0pt \_def\_prepoffsets{}\_else \_if 2#1% double-page layout
 38           \_shiftoffset = \_dimexpr \_pgwidth -\_hsize -2\_hoffset \_relax
 39           \_def\_prepoffsets{\_ifodd\_pageno \_else \_advance\_hoffset \_shiftoffset \_fi}%
 40        \_else \_opwarning{use \_string\_margins/1 or \_string\_margins/2}%
 41     \_fi\_fi\_fi
 42 }
 43 \_def\_setpagedimens{\_isnextchar({\_setpagedimensB}{\_setpagedimensA}}
 44 \_def\_setpagedimensA#1 {\_ifcsname _pgs:#1\_endcsname
 45    \_ea\_ea\_ea\_setpagedimensB \_csname _pgs:#1\_ea\_endcsname\_space
 46    \_else \_opwarning{page specification "#1" is undefined}\_fi}
 47 \_def\_setpagedimensB (#1,#2)#3 {\_setpagedimensC\_pgwidth=#1:#3
 48                                 \_setpagedimensC\_pgheight=#2:#3
 49        \_pdfpagewidth=\_pgwidth \_pdfpageheight=\_pgheight
 50 }
```

```
51  \_def\_setpagedimensC #1=#2:#3 {#1=#2\_ifx^#3^\_tmp\_else#3\_fi\_relax\_truedimen#1}
52
53  \_public \margins ;
```

The common page dimensions are defined here.

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

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

```
68  \_def\_trueunit{}
69  \_def\_magscale[#1]{\_mag=#1\_def\_trueunit{true}%
70    \_ifdim\_pgwidth=0pt \_else \_truedimen\_pgwidth \_truedimen\_pgheight \_fi
71    \_truedimen\_pdfpagewidth \_truedimen\_pdfpageheight
72  }
73  \_def\_truedimen#1{\_ifx\_trueunit\_empty \_else#1=\_ea\_ignorept\_the#1truept \_fi}
74
75  \_public \magscale ;
```

2.20 Colors

2.20.1 Basic concept

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

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

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

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

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

2.20.2 Color mixing

The color mixing processed by the \colordef is done in the subtractive color model CMYK. If the result has a component greater than 1 then all components are multiplied by a coefficient in order to the maximal component is equal to 1.

You can move a shared amount of CMY components (i.e. their minimum) to the *K* component. This saves the color tonners and the result is more true. This should be done by \useK command at the end of a linear combination used in \colordef. For example

    \colordef \myColor {.3\Green + .4\Blue \useK}

The \useK command exactly does:

$$k' = \min(C, M, Y),$$
$$C = (C - k')/(1 - k'), \ M = (M - k')/(1 - k'), \ Y = (Y - k')/(1 - k'),$$
$$K = \min(1, K + k').$$

You can use minus instead of 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 the black component from the color. You can use the -\Black trick after \useK command to remove grey components occurred during color mixing.

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

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

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

If a CMYK to RGB or RGB to CMYK conversion is needed then direct conversion of given color is used (if declared using \rgbcmykmap{⟨rgb⟩}{⟨cmyk⟩}) or the following simple formulae are used (ICC profiles are not supported):

$$\text{CMYK to RGB:}$$
$$R = (1 - C)(1 - K), \ G = (1 - M)(1 - K), \ B = (1 - Y)(1 - K).$$
$$\text{RGB to CMYK:}$$
$$K' = \max(R, G, B), \ C = (K' - R)/K', \ M = (K' - G)/K', \ Y = (K' - B)/K', \ K = 1 - K'.$$

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.

2.20.3 Implementation

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

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

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

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

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

The \onlyrgb declaration redefines \setcmykcolor to do conversion to RGB just before _setcolor is used. The \onlycmyk declaration redefines \setrgbcolor to do conversion to CMYK just before

`\_setcolor` is used. Moreover, `\onlyrgb` re-defines three basic RGB colors for RGB color space and re-declares `\colordef` as `\rgbcolordef`.

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

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

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

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

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

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

The attribute `\_transpattr` is allocated and set by the `\transparency⟨number⟩` macro. If such level of the tranparency was never used in the document then `\addextgstate{tr⟨number⟩}{<</ca X /CA X>>}` is applied (where X is (255−⟨number⟩)/255). This information is used when shipout is processed (similarly as colors). It means `/tr⟨number⟩ gs` is inserted when the attribute is changed.
`\_resetattrs` resets the `\_colorattr` and `\_transpattr` to their initial value -"7FFFFFFF.

```
119  \_newattribute\_transpattr
120  \_def\_transparency {\_afterassignment\_transparencyA \_transpattr}
121  \_def\_transparencyA{%
122     \_ifnum\_transpattr<1 \_transpattr=\_noattr \_fi
123     \_ifnum\_transpattr>255 \_opwarning{\_noexpand\transparency > 255 not allowed}%
124        \_transpattr=\_noattr
125     \_else
126        \_ifcsname _transp:\_the\_transpattr\_endcsname \_else
127           \_edef\_transpv{\_expr{(255-\_the\_transpattr)/255}}%
```

```
128        \_addextgstate{tr\_the\_transpattr}{<</ca \_transpv\_space /CA \_transpv>>}%
129        \_sxdef{_transp:\_the\_transpattr}{}%
130        \_ifcsname _transp:0\_endcsname \_else
131          \_addextgstate{tr0}{<</ca 1 /CA 1>>}%
132          \_sxdef{_transp:0}{}%
133        \_fi
134      \_fi
135    \_fi
136 }
137 \_def\_thetransparency{\_ifnum \_transpattr=-"7FFFFFFF 0\_else \_the\_transpattr \_fi}
138 \_def\_resetattrs{\_colorattr=\_noattr \_transpattr=\_noattr}
139
140 \_public \transparency \thetransparency ;
```

We use Lua codes for RGB to CMYK or CMYK to RGB conversions and for addition color components in the \colordef macro. The _rgbtocmyk ⟨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. The _tocmyk:⟨rgb⟩ or _torgb:⟨cmyk⟩ control sequences (given by \rgbcmykmap) have precedence.

```
157 \_def\_rgbtocmyk #1 #2 #3 ;{\_trycs{_tocmyk:#1 #2 #3}{%
158    \_ea \_stripzeros \_detokenize \_ea{\_directlua{
159      local kr = math.max(#1,#2,#3)
160      if (kr==0) then
161        tex.print('0. 0. 0. 1 ;')
162      else
163        tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
164          (kr-#1)/kr, (kr-#2)/kr, (kr-#3)/kr, 1-kr))
165      end
166 }}}}
167 \_def\_cmyktorgb #1 #2 #3 #4 ;{\_trycs{_torgb:#1 #2 #3 #4}{%
168    \_ea \_stripzeros \_detokenize \_ea{\_directlua{
169      local kr = 1-#4
170      tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f ;',
171        (1-#1)*kr, (1-#2)*kr, (1-#3)*kr))
172 }}}}
173 \_def\_colorcrop{\_directlua{
174    local m=math.max(color_C, color_M, color_Y, color_K)
175    if (m>1) then
176      color_C=color_C/m  color_M=color_M/m  color_Y=color_Y/m color_K=color_K/m
177    end
178 }}
179 \_def\_colordefFin{\_colorcrop \_ea \_stripzeros \_detokenize \_ea{\_directlua{
180    tex.print(string.format('\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;',
181      color_C, color_M, color_Y, color_K))
182 }}}
183 \_def\_douseK{\_colorcrop \_directlua{
184    kr=math.min(color_C, color_M, color_Y)
185    if (kr>=1) then
186      color_C=0  color_M=0  color_Y=0  color_K=1
187    else
188      color_C=(color_C-kr)/(1-kr)  color_M=(color_M-kr)/(1-kr)
189      color_Y=(color_Y-kr)/(1-kr)  color_K=math.min(color_K+kr,1)
190    end
191 }}
```

We have a problem with the %.3f directive in Lua code. It prints trailed zeros: (0.300 instead desired 0.3) but we want to save PDF file space. The macro _stripzeros removes these trailing zeros at the expand processor level. So _stripzeros 0.300 0.400 0.560 ; expands to .3 .4 .56.

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

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

```
216 \_def\_rgbcmykmap#1#2{\_sxdef{_torgb:#2}{#1}\_sxdef{_tocmyk:#1}{#2}}
217 \_public \rgbcmykmap ;
```

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

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

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

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

```
254 \_def\_rgbcolordef  {\_commoncolordef \_setrgbcolor  krg}
255 \_def\_cmykcolordef {\_commoncolordef \_setcmykcolor KRG}
256 \_def\_commoncolordef#1#2#3#4#5#6{%
257   \_begingroup
258     \_directlua{color_C=0 color_M=0 color_Y=0 color_K=0}%
259     \_def\_setcmykcolor##1{!=#2 ##1 }%
260     \_def\_setrgbcolor ##1{!=#3 ##1 }%
261     \_def\_setgreycolor##1{!=#4 ##1 }%
262     \_let\_useK=\_relax
263     \_edef\_tmpb{+#6}%
264     \_replstring\_tmpb{+ }{+}\_replstring\_tmpb{- }{-}%
265     \_replstring\_tmpb{+}{\_addcolor}\_replstring\_tmpb{-}{\_addcolor-}%
266     \_replstring\_tmpb{^!=}{!^}\_replstring\_tmpb{-!}{-1!}%
267     \_ifx K#2\_let\_useK=\_douseK \_fi
268     \_tmpb
269     \_edef\_tmpb{\_colordefFin}%
270     \_ifx k#2\_edef\_tmpb{\_ea\_stripK \_tmpb;}\_fi
271   \_ea\_endgroup
272   \_ea\_def\_ea#5\_ea{\_ea#1\_ea{\_tmpb}}%
273 }
274 \_def\_addcolor#1!#2#3{\_cs{addcolor:#2#3}#1}
275 \_def\_addcolorA #1 #2 #3 #4 #5 {%
276   \_def\_tmpa{#1}\_ifx\_tmpa\_empty \_else \_edef\_tmpa{\_tmpa*}\_fi
277   \_directlua{color_C=math.max(color_C+\_tmpa#2,0)
278              color_M=math.max(color_M+\_tmpa#3,0)
279              color_Y=math.max(color_Y+\_tmpa#4,0)
280              color_K=math.max(color_K+\_tmpa#5,0)
281 }}
282 \_sdef{addcolor:=K}#1 #2 #3 #4 #5 {\_addcolorA #1 #2 #3 #4 #5 }
283 \_sdef{addcolor:^K}#1 #2 #3 #4 #5 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) #5 }
284 \_sdef{addcolor:^G}#1 #2 {\_addcolorA #1 0 0 0 #2 }
```

```
285  \_sdef{addcolor:=G}#1 #2 {\_addcolorA #1 0 0 0 (1-#2) }
286  \_sdef{addcolor:=R}#1 #2 #3 #4 {%
287     \_edef\_tmpa{\_noexpand\_addcolorA #1 \_rgbtocmyk #2 #3 #4 ; }\_tmpa
288  }
289  \_sdef{addcolor:^R}#1 #2 #3 #4 {\_cs{addcolor:=R}#1 (1-#2) (1-#3) (1-#4) }
290
291  \_sdef{addcolor:=k}#1 #2 #3 #4 #5 {%
292     \_edef\_tmpa{\_noexpand\_addcolorA #1 \_cmyktorgb #2 #3 #4 #5 ; 0 }\_tmpa
293  }
294  \_sdef{addcolor:^k}#1 #2 #3 #4 #5 {\_cs{addcolor:=k}#1 (1-#2) (1-#3) (1-#4) #5 }
295  \_sdef{addcolor:^g}#1 #2 {\_addcolorA #1 (1-#2) (1-#2) (1-#2) 0 }
296  \_sdef{addcolor:=g}#1 #2 {\_addcolorA #1 #2 #2 #2 0 }
297  \_sdef{addcolor:=r}#1 #2 #3 #4 {\_addcolorA #1 #2 #3 #4 0 }
298  \_sdef{addcolor:^r}#1 #2 #3 #4 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) 0 }
299  \_def\_stripK#1 .0;{#1}
300  \_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.

```
308  \_def \useK{\_useK}
309  \_def \colordef {\_colordef}
310  \_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[10/14]
\morecolors
\endmulti
```

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

2.21 The .ref file

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

```
\Xrefversion{⟨ref-version⟩}
\_Xpage{⟨gpageno⟩}{⟨pageno⟩}
\_Xtoc{⟨level⟩}{⟨type⟩}{⟨text⟩}{}⟨title⟩
\_Xlabel{⟨label⟩}{⟨text⟩}
\_Xlabel{⟨label⟩}{⟨text⟩}
...
\_Xpage{⟨gpageno⟩}{⟨pageno⟩}
\_Xlabel{⟨label⟩}{⟨text⟩}
...
```

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

<div style="text-align: right">ref-file.opm</div>

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

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

<div style="text-align: right">ref-file.opm</div>

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

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

<div style="text-align: right">ref-file.opm</div>

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

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

<div style="text-align: right">ref-file.opm</div>

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

If a macro needs to create and use the `.ref` file, then such macro must first use `\openref`. It creates the file and redefines `\_wref` \⟨*macro*⟩{⟨*data*⟩} so that it saves the line \⟨*macro*⟩⟨*data*⟩ to the `.ref` file using the asynchronous `\write` primitive. Finally, `\_openref` destroys itself, because we don't need to open the file again.
`\_wref`⟨*csname*⟩{⟨*params*⟩} in fact does `\write\_reffile{\string`⟨*csname*⟩⟨*params*⟩}` and similarly `_ewref`⟨*csname*⟩{⟨*params*⟩} does `\write_reffile{\string`⟨*csname*⟩⟨*expanded-params*⟩}`.

<div style="text-align: right">ref-file.opm</div>

```
57  \_def\_openref {%
58     \_immediate\_openout\_reffile="\_jobname.ref"\_relax
59     \_gdef\_wref ##1##2{\_write\_reffile{\_bslash\_csstring##1##2}}%
60     \_immediate\_write\_reffile {\_pcent\_pcent\_space OpTeX <\_optexversion> - REF file}%
61     \_immediate\_wref \Xrefversion{{\_REFversion}}%
62     \_ifx\_refdecldata\_empty \_else \_refdeclwrite \_fi
63     \_gdef\_openref{}%
64  }
65  \_def\_ewref #1#2{\_edef\_ewrefA{#2}\_ea\_wref\_ea#1\_ea{\_ewrefA}}
66  \_def\openref{\_openref}
```

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

   `\Xrefversion`{⟨*version*⟩}

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

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

You cannot define your own `.ref` macros before `.ref` file is read because it is read in `\everyjob`. But you can define such macros by using `\refdecl`{⟨*definitions of your ref macros*⟩}. This command writes ⟨*definitions of your ref macros*⟩ to the `.ref` file. Then the next lines written to the `.ref` file can include your macros. An example from CTUstyle2:

```
\refdecl{%
   \def\totlist{} \def\toflist{}^^J
   \def\Xtab#1#2#3{\addto\totlist{\totline{#1}{#2}{#3}}}^^J
   \def\Xfig#1#2#3{\addto\toflist{\tofline{#1}{#2}{#3}}}
}
```

We must read ⟨*definitions of your ref macros*⟩ while `#` has the catcode 12, because we don't want to duplicate each `#` in the `.ref` file.
`\refdecl` appends its data to the `\_refdecldata` macro. It is pushed to the `.ref` file immediatelly only if the file is opened already. Otherwise we are waiting to `\openref` because we don't want to open the `.ref` file if it is unnecessary.

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

2.22   References

If the references are "forward" (i. e. the `\ref` is used first, the destination is created later) or if the reference text is page number then we must read `.ref` file first in order to get appropriate information. See section 2.21 for more information about `.ref` file concept.

```
3 \_codedecl \ref {References <2021-04-13>} % 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.

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

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

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

```
22 \_newcount\_unresolvedrefs
23 \_unresolvedrefs=0
```

`\_Xlabel` {⟨*label*⟩}{⟨*text*⟩} saves the ⟨*text*⟩ to `\_lab:`⟨*label*⟩ and saves {⟨*gpageno*⟩}{⟨*pageno*⟩} to `\_pgref:`⟨*label*⟩.

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

\label[⟨*label*⟩] saves the declared label to _lastlabel and \wlabel{⟨*text*⟩} uses the _lastlabel and activates _wref_Xlabel{⟨*label*⟩}{⟨*text*⟩}.

```
38 \_def\_label[#1]{\_isempty{#1}\_iftrue \_global\_let \_lastlabel=\_undefined
39    \_else \_isdefined{l0:#1}%
40       \_iftrue \_slideshook\_opwarning{Duplicated label [#1], ignored}\_else \_xdef\_lastlabel{#1}\_fi
41    \_fi \_ignorespaces
42 }
43 \_let \_slideshook=\_relax % redefined if \slides + \slideshow.
44 \_def\_wlabel#1{%
45    \_ifx\_lastlabel\_undefined \_else
46       \_dest[ref:\_lastlabel]%
47       \_printlabel\_lastlabel
48       \_ewref \_Xlabel {{\_lastlabel}{#1}}%
49       \_sxdef{_lab:\_lastlabel}{#1}\_sxdef{l0:\_lastlabel}{}%
50       \_global\_let\_lastlabel=\_undefined
51    \_fi
52 }
53 \_public \label \wlabel ;
```

\ref[⟨*label*⟩]{⟨*given-text*⟩} prints (linked) ⟨*given-text*⟩. The missing optional {⟨*given-text*⟩} is replaced by {@}. The @ is replaced by ⟨*implicit-text*⟩ from saved \lab:⟨*label*⟩ using _reftext macro. If the reference is backward then we know \lab:⟨*label*⟩ without any need to read REF file. On the other hand, if the reference is forwarded, then we doesn't know _lab:⟨*label*⟩ in the first run of TeX and we print a warning and do _openref.

\pgref[⟨*label*⟩]{⟨*given-text*⟩} prints ⟨*given-text*⟩ where @ is replaced by ⟨*pageno*⟩. Data in the format {⟨*gpageno*⟩}{⟨*pageno*⟩} are read from _pgref:⟨*label*⟩ by _pgrefB{⟨*gpageno*⟩}{⟨*pageno*⟩}{⟨*given-text*⟩}.

_lastreflabel keeps the value of the last label read by \ref or \pgref. You can use it for example by definig a macro \pg by \def\pg{\pgref[_lastreflabel]} and then you need not repeat the same label in typical situations and you can write for instance: see section \ref[lab] at page \pg.

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

_reftext{⟨*implicit-text*⟩}{⟨*given-text*⟩} expands to the ⟨*given-text*⟩ but the optional @ in the ⟨*given-text*⟩ is replaced by the ⟨*implicit-text*⟩ first.

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

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.

```
107 \_def\_printlabel#1{}
108 \_def\_showlabels {%
109    \_def\_printlabel##1{\_vbox to\_zo{\_vss\_llap{\_labelfont[##1]}\_kern1.7ex}}%
110    \_fontdef\_labelfont{\_setfontsize{at10pt}\setfontcolor{blue}\_tt}
111 }
112 \_public \showlabels ;
```

2.23 Hyperlinks

There are six types of internal links and one type of external link used in OpTEX. They are used in the format ⟨*type*⟩:⟨*spec*⟩.

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

The ⟨*tocrefnum*⟩, ⟨*gpageno*⟩, ⟨*bibnum*⟩, and ⟨*gfnotenum*⟩ are numbers starting from one and globally incremented by one in the whole document. The registers `\tocrefnum`, `\gpageno`, `\bibnum`, and `\_gfnotenum` 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*⟩.

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

```
                                                                                    hyperlinks.opm
 3  \_codedecl \ulink {Hyperlinks <2021-08-31>} % 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 the top border of the PDF viewer but we want to see the line where the destination is. The destination box is positioned by a different way dependent on the current vertical or horizontal mode.

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

Each hyperlink is created internally by `\_xlink{`⟨*type*⟩`}{`⟨*spec*⟩`}{`⟨*color*⟩`}{`⟨*text*⟩`}`. This macro expands to `\_quitvmode{`⟨*text*⟩`}` by default, i.e. no active hyperlink is created, only ⟨*text*⟩ is printed in horizontal mode (and in a group). If `\hyperlinks` is used, then `\_xlink` gets the meaning of `\_xlinkactive` and hyperlinks are created by the `\pdfstartlink`/`\pdfendlink` primitives. The ⟨*text*⟩ has given ⟨*color*⟩ only when hyperlink is created. If `\_`⟨*type*⟩`linkcolor` is defined, it has precedence over ⟨*color*⟩.
The `\_linkdimens` macro declares the dimensions of link area.
A specific action can be defined for each link ⟨*type*⟩ by the macro `\_`⟨*type*⟩`action{`⟨*spec*⟩`}`. OpTEX defines only `\_urlaction{`⟨*url*⟩`}`. The default link action (when `\_`⟨*type*⟩`action` is not defined) is `goto name{`⟨*type*⟩`:`⟨*spec*⟩`}` (an internal link). It is declared in the `\_linkactions{`⟨*type*⟩`}{`⟨*spec*⟩`}` macro. The `\_pdfstartlink` primitive uses `attr{\_pdfborder{`⟨*type*⟩`}}`. The `\_pdfborder{`⟨*type*⟩`}` macro expands to `/C[? ? ?] /Border[0 0 .6]` if the `\_`⟨*type*⟩`border` macro (i.e. `\_refborder`, `\_citeborder`, `\_tocborder`, `\_pgborder`, `\_urlborder`, `\_fntborder` or `\_fnfborder`) is defined.

```
                                                                                    hyperlinks.opm
52  \_protected\_def\_xlinkactive#1#2#3#4{\_quitvmode
53    \_pdfstartlink \_linkdimens attr{\_pdfborder{#1}}\_linkactions{#1}{#2}\_relax
54    {\_localcolor\_trycs{_#1linkcolor}{#3}#4}\_pdfendlink
55  }
56  \_protected\_def\_xlink#1#2#3#4{\_quitvmode{#4}}
57
58  \_def\_linkdimens{height.9em depth.3em}
```

```
 59
 60  \_def\_linkactions#1#2{\_ifcsname _#1action\_endcsname
 61      \_lastnamedcs{#2}\_else goto name{#1:#2}\_fi}
 62  \_def\_urlaction #1{user{/Subtype/Link/A <</Type/Action/S/URI/URI(#1)>>}}
 63
 64  \_def\_pdfborder#1{\_ifcsname _#1border\_endcsname
 65        /C [\_csname _#1border\_endcsname] /Border [0 0 .6]\_else  /Border [0 0 0]\_fi
 66  }
```

\link[⟨type⟩:⟨spec⟩]{⟨color⟩}{⟨text⟩} creates a link. It is kept here for backward compatibility and it
is equivalent to _xlink{⟨type⟩}{⟨spec⟩}{⟨color⟩}{⟨text⟩}. If _⟨type⟩action is not defined then \link
creates internal link do the \dest[⟨type⟩:⟨spec⟩]. You can have more links with the same ⟨type⟩:⟨spec⟩
but only one \dest in the document.

\ilink[⟨type⟩:⟨spec⟩]{⟨text⟩} is equivalent to \link but the ⟨color⟩ is used from \hyperlinks declara-
tion (or it is overwriten by \def_⟨type⟩linkcolor).

\ulink[⟨url⟩]{⟨text⟩} creates external link. The ⟨url⟩ is detokenized with \escapechar=-1 before it is
used, so \%, \# etc. can be used in the ⟨url⟩.

```
 86  \_def\_link[#1:#2]{\_xlink{#1}{#2}}
 87  \_def\_ilink[#1:#2]#3{\_xlink{#1}{#2}\_ilinkcolor{#3}}
 88  \_def\_ulink[#1]#2{{\_escapechar=-1 \_ea}\_expanded
 89      {\_noexpand\_xlink{url}{\_detokenize{#1}}}\_elinkcolor{#2}}
 90
 91  \_public \ilink \ulink \link ;
```

\hyperlinks⟨ilink color⟩⟨ulink color⟩ activates \dest, \xlink, so that they create links. Not setting
colors (\hyperlinks{}{}) is also supported.

```
 99  \_def\_hyperlinks#1#2{%
100      \_let\_dest=\_destactive \_let\_xlink=\_xlinkactive
101      \_let\_ilinkcolor=#1\_empty
102      \_let\_elinkcolor=#2\_empty
103      \_public \dest \xlink ;%
104  }
105  \_public \hyperlinks ;
```

\url{⟨url⟩} does approximately the same as \ulink[⟨url⟩]{⟨url⟩}, but more work is done before the
\ulink is processed. The link-version of ⟨url⟩ is saved to _tmpa and the printed version in _tmpb. The
printed version is processed in four steps: 1. the \| are replaced by [||] (we suppose that such string does
not exist in any URL). 2. it is detokenized with \escapechar=-1. 3. muti-strings and spaces are replaced
by strings in braces {...}. 4. internal penalties and skips are put between characters using _urlA,
_urlB and _urlC. The step 4 do following: The _urlxskip is inserted between each pair of "normal
characters", i.e. characters not declared by \sdef{_ur:⟨character⟩}. The special characters declared
by \sdef{_ur:⟨character⟩} are replaced by the body of their corresponding macro. The _urlskip,
_urlbskip, _urlgskip are typical skips used for special characters, their meaning is documented in
the code below. You can change them. Default values: penalty 9990 is inserted between each pair of
normal chararacters, penalty 100 is inserted after special charcters, nobreak before special characters.
The URL can be broken at any place using these default values. If you want to disable breaking between
normal characters, say \let_urlxskip=\nobreak.

The text version of the ⟨url⟩ is printed in _urlfont.

```
132  \_def\_url#1{{%
133      \_def\_tmpa{#1}\_replstring\_tmpa {\|}{}%
134      \_def\_tmpb{#1}\_replstring\_tmpb {\|}{[||]}%
135      {\_escapechar=-1 \_ea}\_ea\_edef\_ea\_tmpb\_ea{\_detokenize\_ea{\_tmpb}}%
136      \_replstring\_tmpb{[||]}{{gb|}}%
137      \_replstring\_tmpb{ }{{ }}%
138      \_replstring\_tmpb{://}{{://}}%
139      \_ea\_ulink \_ea[\_ea{\_tmpa}] {\_urlfont \_ea\_urlA\_tmpb\_end}%
140  }}
141  \_def\_urlA#1{\_ifx\_end#1\_else \_urlC{}{#1}\_fi}
142  \_def\_urlB#1{\_ifx\_end#1\_else \_urlC{\_urlxskip}{#1}\_fi}
143  \_def\_urlC#1#2{%
144      \_ifcsname _ur:#2\_endcsname \_lastnamedcs \_ea\_ea\_ea \_urlA
145      \_else #1#2\_ea\_ea\_ea \_urlB \_fi
```

```
146 }
147 \_sdef{_ur:://}{\_urlskip:\_urlskip/\_urlskip/\_urlbskip}
148 \_sdef{_ur:/}{\_urlskip/\_urlbskip}
149 \_sdef{_ur:.}{\_urlskip.\_urlbskip}
150 \_sdef{_ur:?}{\_urlskip?\_urlbskip}
151 \_sdef{_ur:=}{\_urlskip=\_urlbskip}
152 \_sdef{_ur:-}{\_urlskip-\_urlbskip}
153 \_sdef{_ur:&}{\_urlskip\_char`\&\_urlbskip}
154 \_sdef{_ur:gb|}{\_urlgskip}
155
156 \_def\_urlfont{\_tt}                  % url font
157 \_def\_urlxskip{\_penalty9990\_hskip0pt plus0.03em\_relax} % skip between normal characters
158 \_def\_urlskip{\_null\_nobreak\_hskip0pt plus0.1em\_relax} % skip before :// / . ? = - &
159 \_def\_urlbskip{\_penalty100 \_hskip0pt plus0.1em\_relax}  % skip after  :// / . ? = - &
160 \_def\_urlgskip{\_penalty-500\_relax}   % "goodbreak" penalty generated by \|
161
162 \_public \url ;
```

2.24   Making table of contents

```
3 \_codedecl \maketoc {Macros for maketoc <2021-07-18>} % preloaded in format
```

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

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

The last two parameters are restored from previous _Xpage{⟨pageno⟩}{⟨gpageno⟩}, data were saved in the _currpage macro.
    We read the ⟨title⟩ parameter by \scantoeol from .ref file because the ⟨title⟩ can include something like `{`.

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

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

```
                    |                               |
   \llap{⟨number⟩} text text text text text
                  text text text text text
                  text text ................... ⟨pageno⟩
```

Margins given by \leftskip and \rightskip are denoted by | in the examle above.
\tocrefnum is the global counter of all TOC records (used by hyperlinks).

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

You can re-define default macros for each level of tocline if you want.
Parameters are {⟨*number*⟩}{⟨*title*⟩}{⟨*pageno*⟩}.

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

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`.

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

If you want a special formating of TOC with adding more special lines (no generated as titles from `\chap`, `\sec`, `\secc`), you can define `\addtotoc{`⟨*level*⟩`}{`⟨*type*⟩`}{`⟨*number*⟩`}{`⟨*o-title*⟩`}{`⟨*title*⟩`}` macro:

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

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

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

Now, users can add a blue line into TOC by

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

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

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

```
128  \_def\_maketoc{\_par \_ifx\_toclist\_empty
129      \_opwarning{\_noexpand\maketoc -- data unavailable, TeX me again}\_openref
130      \_incr\_unresolvedrefs
131  \_else \_begingroup
132      \_tocrefnum=0 \_penalty11333
133      \_the\_regtoc \_toclist
134  \_endgroup \_fi
135  }
```

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

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

2.25 PDF outlines

2.25.1 Nesting PDF outlines

The problem is that PDF format needs to know the number of direct descendants of each outline if we need to create the tree of structured outlines. But we know only the level of each outline. The required data should be calculated from TOC data. We use two steps over TOC data saved in the `\_toclist` where each record is represented by one `\_tocline`.

The first step, the `\outlines` macro sets `\_tocline` to `\_outlinesA` and calculates the number of direct descendants of each record. The second step, the `\outlines` macro sets `\_tocline` to `\_outlinesB` and it uses prepared data and creates outlines.

Each outline is mapped to the control sequence of the type `\_ol:`⟨*num*⟩ or `\_ol:`⟨*num*⟩:⟨*num*⟩ or `\_ol:`⟨*num*⟩:⟨*num*⟩:⟨*num*⟩ or etc. The first one is reserved for level 0, the second one for level 1 (chapters), the third one for level 2 (sections) etc. The number of direct descendants will be stored in these macros after the first step is finished. Each new outline of a given level increases the ⟨*num*⟩ at the given level. When the first step is processed then (above that) the `\_ol:...` sequence of the parent increases its value too. The `_ol:...` sequences are implemented by `\_ol:\_count0:\_count1:\count2` etc. For example, when section (level 2) is processed in the first step then we do:

```
\advance \count2 by 1
                % increases the mapping pointer of the type
                % \_ol:\_count0:\_count1:\_count2 of this section
\advance \_ol:\_count0:\_count1 by 1
                % increases the number of descendants connected
                % to the parent of this section.
```

When the second step is processed, then we only read the stored data about the number of descendants. And we use it in `count` parameter of `\_pdfoutline` primitive.

For linking, we use the same links as in TOC, i.e. the `toc:\_the\_tocrefnum` labels are used.

`\insertoutline` `{`⟨*text*⟩`}` inserts one outline with zero direct descendants. It creates a link destination of the type `oul:`⟨*num*⟩ into the document (where `\insertoutline` is used) and the link itself is created too in the outline.

```
3   \_codedecl \outlines {PDF outlines <2021-02-09>} % preloaded in format
4
5   \_def\_outlines#1{\_pdfcatalog{/PageMode/UseOutlines}\_openref
6      \_ifx\_toclist\_empty
7        \_opwarning{\_noexpand\outlines -- data unavailable. TeX me again}%
8        \_incr\_unresolvedrefs
9      \_else
10       \_ifx\_dest\_destactive \_else
11          \_opwarning{\_noexpand\outlines doesn't work when \_noexpand\hyperlinks isn't declared}\_fi
12       {\_let\_tocline=\_outlinesA
13        \_count0=0 \_count1=0 \_count2=0 \_count3=0 \_toclist % calculate numbers o childs
14        \_def\_outlinelevel{#1}\_let\_tocline=\_outlinesB
15        \_tocrefnum=0 \_count0=0 \_count1=0 \_count2=0 \_count3=0
16        \_toclist}% create outlines
17     \_fi
18  }
19  \_def\_outlinesA#1#2#3#4#5#6#7{%
20      \_isequal{\relax}{#4}\_iffalse
21          \_advance\_count#1 by1
22          \_ifcase#1\_or
```

```
23          \_addoneol{_ol:\_the\_count0}\_or
24          \_addoneol{_ol:\_the\_count0:\_the\_count1}\_or
25          \_addoneol{_ol:\_the\_count0:\_the\_count1:\_the\_count2}\_or
26          \_addoneol{_ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}\_fi
27      \_fi
28  }
29  \_def\_addoneol#1{%
30      \_ifcsname #1\_endcsname
31              \_tmpnum=\_csname#1\_endcsname\_relax
32              \_advance\_tmpnum by1 \_sxdef{#1}{\_the\_tmpnum}%
33      \_else \_sxdef{#1}{1}%
34      \_fi
35  }
36  \_def\_outlinesB#1#2#3#4#5#6#7{%
37      \_advance\_tocrefnum by1
38      \_isequal{\relax}{#4}\_iffalse
39          \_advance\_count#1 by1
40          \_ifcase#1%
41              \_tmpnum=\_trycs{_ol:\_the\_count0}{0}\_or
42              \_tmpnum=\_trycs{_ol:\_the\_count0:\_the\_count1}{0}\_relax\_or
43              \_tmpnum=\_trycs{_ol:\_the\_count0:\_the\_count1:\_the\_count2}{0}\_relax\_or
44              \_tmpnum=\_trycs{_ol:\_the\_count0:\_the\_count1:\_the\_count2:\_the\_count3}{0}\_relax\_or
45              \_tmpnum = 0\_relax\_fi
46          \_isempty{#4}\_iftrue \_pdfunidef\_tmp{#5}\_else \_pdfunidef\_tmp{#4}\_fi
47          \_outlinesC{toc:\_the\_tocrefnum}{\_ifnum#1<\_outlinelevel\_space\_else-\_fi}{\_tmpnum}{\_tmp}%
48      \_fi
49  }
50  \_def\_outlinesC#1#2#3#4{\_pdfoutline goto name{#1} count #2#3{#4}\_relax}
51
52  \_newcount\_oulnum
53  \_def\_insertoutline#1{\_incr\_oulnum
54      \_pdfdest name{oul:\_the\_oulnum} xyz\_relax
55      \_pdfunidef\_tmp{#1}%
56      \_pdfoutline goto name{oul:\_the\_oulnum} count0 {\_tmp}\_relax
57  }
58  \_public \outlines \insertoutline ;
```

2.25.2 Strings in PDF outlines

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

The second encoding is Big Endian UTF-16 which is implemented in this file. It encodes a single character in either two or four bytes. This encoding is TeX-discomfortable because it looks like

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

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

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

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

```
10  \bgroup
11  \_catcode`\%=12
12  \_gdef\_hexprint{\_directlua{
13      local num = token.scan_int()
14      if num < 0x10000 then
15          tex.print(string.format("%04X", num))
16      else
17          num = num - 0x10000
```

```
18        local high = bit32.rshift(num, 10) + 0xD800
19        local low = bit32.band(num, 0x3FF) + 0xDC00
20        tex.print(string.format("%04X%04X", high, low))
21     end
22  }}
23  \egroup
```

\pdfunidef\macro{⟨text⟩} defines \macro as ⟨text⟩ converted to Big Endian UTF-16 and enclosed to
<>. Example of usage: \pdfunidef\infoauthor{Petr Olšák} \pdfinfo{/Author \infoauthor}.
\pdfunidef does more things than only converting to hexadecimal PDF string. The ⟨text⟩ can be
scanned in verbatim mode (it is true becuase _Xtoc reads the ⟨text⟩ in verbatim mode). First \edef
do _scantextokens\unexpanded and second \edef expands the parameter according to current values
on selected macros from _regoul. Then _removeoutmath converts ..x^2.. to ..x^2.., i.e removes
dollars. Then _removeoutbraces converts ..{x}.. to ..x... Finally, the ⟨text⟩ is detokenized, spaces
are preprocessed using \replstring and then the _pdfunidefB is repeated on each character. It calls
the \directlua chunk to print hexadecimal numbers in the macro _hexprint.
Characters for quotes (and separators for quotes) are activated by first _scatextokens and they are
defined as the same non-active characters. But _regoul can change this definition.

```
44  \_def\_pdfunidef#1#2{%
45     \_begingroup
46        \_catcodetable\_optexcatcodes \_adef"{"}\_adef'{'}%
47        \_the\_regoul \_relax % \_regmacro alternatives of logos etc.
48        \_ifx\_savedttchar\_undefined \_def#1{\_scantextokens{\_unexpanded{#2}}}%
49        \_else \_lccode`\;=\_savedttchar \_lowercase{\_prepinverb#1;}{#2}\fi
50        \_edef#1{#1}%
51        \_escapechar=-1
52        \_edef#1{#1\_empty}%
53        \_escapechar=`\\
54        \_ea\_edef \_ea#1\_ea{\_ea\_removeoutmath   #1$\_end$}%  $x$ -> x
55        \_ea\_edef \_ea#1\_ea{\_ea\_removeoutbraces #1{\_end}}%  {x} -> x
56        \_edef#1{\_detokenize\_ea{#1}}%
57        \_replstring#1{ }{{ }}%  text text -> text{ }text
58        \_catcode`\\=12 \_let\\=\_bslash
59        \_edef\_out{<FEFF}
60        \_ea\_pdfunidefB#1^%  text -> \_out in octal
61        \_ea
62     \_endgroup
63     \_ea\_def\_ea#1\_ea{\_out>}
64  }
65  \_def\_pdfunidefB#1{%
66     \_ifx^#1\_else
67        \_edef\_out{\_out \_hexprint `#1}
68     \_ea\_pdfunidefB \_fi
69  }
70
71  \_def\_removeoutbraces #1#{#1\_removeoutbracesA}
72  \_def\_removeoutbracesA #1{\_ifx\_end#1\_else #1\_ea\_removeoutbraces\_fi}
73  \_def\_removeoutmath #1$#2${#1\_ifx\_end#2\_else #2\_ea\_removeoutmath\_fi}
```

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

```
84  \_def\_prepinverb#1#2#3{\_def#1{}%
85     \_def\_dotmpb ##1#2##2{\_addto#1{\_scantextokens{\_unexpanded{##1}}}%
86        \_ifx\_end##2\_else\_ea\_dotmpbA\_ea##2\_fi}%
87     \_def\_dotmpbA ##1#2{\_addto#1{##1}\_dotmpb}%
88     \_dotmpb#3#2\_end
89  }
```

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

```
97  \_regmacro {}{}{\_let\em=\_empty \_let\rm=\_empty \_let\bf=\_empty
98     \_let\it=\_empty \_let\bi=\_empty \_let\tt=\_empty \_let\/=\_empty
```

124

2.26 Chapters, sections, subsections

```
3 \_codedecl \chap {Titles, chapters, sections, subsections <2022-08-02>} % preloaded in format
```

We are using scaled fonts for titles **_titfont**, **_chapfont**, **_secfont** and **_seccfont**. They are scaled from main fonts size of the document, which is declared by first **\typosize[**⟨*fo-size*⟩/⟨*b-size*⟩**]** command.

```
13 \_def \_titfont  {\_scalemain\_typoscale[\_magstep4/\_magstep5]\_boldify}
14 \_def \_chapfont {\_scalemain\_typoscale[\_magstep3/\_magstep3]\_boldify}
15 \_def \_secfont  {\_scalemain\_typoscale[\_magstep2/\_magstep2]\_boldify}
16 \_def \_seccfont {\_scalemain\_typoscale[\_magstep1/\_magstep1]\_boldify}
```

The **\tit** macro is defined using **\scantoeol** and **_printtit**. It means that the parameter is separated by end of line and inline verbatim is allowed. The same principle is used in the **\chap**, **\sec**, and **\secc** macros.

```
25 \_def\_printtit #1{\_vglue\_titskip
26   {\_leftskip=0pt plus1fill \_rightskip=\_leftskip % centering
27    \_titfont \_noindent \_scantextokens{#1}\_par}%
28    \_nobreak\_bigskip
29 }
30 \_def\_tit{\_scantoeol\_printtit}
31
32 \_public \tit ;
```

You can re-define **_printchap**, **_printsec** or **_printsecc** macros if another design of section titles is needed. These macros get the ⟨*title*⟩ text in its parameter. The common recommendations for these macros are:

- Use **_abovetitle**{⟨*penaltyA*⟩}{⟨*skipA*⟩} and **_belowtitle**{⟨*skipB*⟩} for inserting vertical material above and below the section title. The arguments of these macros are normally used, i. e. **_abovetitle** inserts ⟨*penaltyA*⟩⟨*skipA*⟩ and **_belowtitle** inserts ⟨*skipB*⟩. But there is an exception: if **_belowtitle**{⟨*skipB*⟩} is immediately followed by **_abovetitle**{⟨*penaltyA*⟩}{⟨*skipA*⟩} (for example section title is immediately followed by subsection title), then only ⟨*skipA*⟩ is generated, i. e. ⟨*skipB*⟩⟨*penaltyA*⟩⟨*skipA*⟩ is reduced only to ⟨*skipA*⟩. The reason for such behavior: we don't want to duplicate vertical skip and we don't want to use the negative penalty in such cases. Moreover, **_abovetitle**{⟨*penaltyA*⟩}{⟨*skipA*⟩} takes previous whatever vertical skip (other than from **_belowtitle**) and generates only greater from this pair of skips. It means that ⟨*whatever-skip*⟩⟨*penaltyA*⟩⟨*skipA*⟩ is transformed to ⟨*penaltyA*⟩max(⟨*whatever-skip*⟩⟨*skipA*⟩). The reason for such behavior: we don't want to duplicate vertical skips (from **_belowlistskip**, for example) above the title.
- Use **_printrefnum**[⟨*pre*⟩@⟨*post*⟩] in horizontal mode. It prints ⟨*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 the label (if **\label**[⟨*label*⟩] precedes) and saves references for the table of contents (if **\maketoc** is used).
- Use **\nbpar** for closing the paragraph for printing title. This command inserts **_nobreak** between each line of such paragraph, so the title cannot be broken into more pages.
- You can use **_firstnoindent** in order to the first paragraph after the title is not indented.

```
72 \_def\_printchap #1{\_vfill\_supereject \_prevdepth=0pt
73    \_vglue\_medskipamount % shifted by topkip+\medskipamount
74    {\_chapfont \_noindent \_mtext{chap} \_printrefnum[@]\_par
75     \_nobreak\_smallskip
76     \_noindent \_raggedright #1\_nbpar}\_mark{}%
77    \_nobreak \_belowtitle{\_bigskip}%
78    \_firstnoindent
```

125

```
79  }
80  \_def\_printsec#1{\_par
81     \_abovetitle{\_penalty-151}\_bigskip
82     {\_secfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbpar}\_insertmark{#1}%
83     \_nobreak \_belowtitle{\_medskip}%
84     \_firstnoindent
85  }
86  \_def\_printsecc#1{\_par
87     \_abovetitle{\_penalty-101}{\_medskip\_smallskip}
88     {\_seccfont \_noindent \_raggedright \_printrefnum[@\_quad]#1\_nbpar}%
89     \_nobreak \_belowtitle{\_medskip}%
90     \_firstnoindent
91  }
```

The `\_sectionlevel` is the level of the printed section:

- `\_sectionlevel=0` – reserved for parts of the book (unused by default)
- `\_sectionlevel=1` – chapters (used in `\chap`)
- `\_sectionlevel=2` – sections (used in `\sec`)
- `\_sectionlevel=3` – subsections (used in `\secc`)
- `\_sectionlevel=4` – subsubsections (unused by default, see the OpTeX trick 0033)

```
                                                                    sections.opm
105  \_newcount\_sectionlevel
106  \_def \_secinfo {\_ifcase \_sectionlevel
107     part\_or chap\_or sec\_or secc\_or seccc\_fi
108  }
```

The `\_chapx` initializes counters used in chapters, the `\_secx` initializes counters in sections and `\_seccx` initializes counters in subsections. If you have more types of numbered objects in your document then you can declare appropriate counters and do `\addto\_chapx{\yourcounter=0 }` for example. If you have another concept of numbering objects used in your document, you can re-define these macros. All settings here are global because it is used by `{\_globaldefs=1 \_chapx}`.

Default concept: Tables, figures, and display maths are numbered from one in each section – subsections don't reset these counters. Footnotes declared by `\fnotenumchapters` are numbered in each chapter from one.

The `\_the*` macros `\_thechapnum`, `\_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
136  \_newcount \_chapnum   % chapters
137  \_newcount \_secnum    % sections
138  \_newcount \_seccnum   % subsections
139  \_newcount \_tnum      % table numbers
140  \_newcount \_fnum      % figure numbers
141  \_newcount \_dnum      % numbered display maths
142
143  \_def \_chapx {\_secx  \_secnum=0  \_lfnotenum=0 }
144  \_def \_secx  {\_seccx \_seccnum=0 \_tnum=0 \_fnum=0 \_dnum=0 \_resetABCDE }
145  \_def \_seccx {}
146
147  \_def \_thechapnum {\_the\_chapnum}
148  \_def \_thesecnum  {\_othe\_chapnum.\_the\_secnum}
149  \_def \_theseccnum {\_othe\_chapnum.\_the\_secnum.\_the\_seccnum}
150  \_def \_thetnum    {\_othe\_chapnum.\_othe\_secnum.\_the\_tnum}
151  \_def \_thefnum    {\_othe\_chapnum.\_othe\_secnum.\_the\_fnum}
152  \_def \_thednum    {(\_the\_dnum)}
153
154  \_def\_othe #1.{\_ifnum#1>0 \_the#1.\_fi}
```

The `\notoc` and `\nonum` prefixes are implemented by internal `\_ifnotoc` and `\_ifnonum`. They are reset after each chapter/section/subsection by the `\_resetnonumnotoc` macro.

```
                                                                    sections.opm
162  \_newifi \_ifnotoc  \_notocfalse  \_def\_notoc {\_global\_notoctrue}
163  \_newifi \_ifnonum  \_nonumfalse  \_def\_nonum {\_global\_nonumtrue}
164  \_def \_resetnonumnotoc{\_global\_notocfalse \_global\_nonumfalse}
165  \_public \_notoc \_nonum ;
```

126

The `\chap`, `\sec`, and `\secc` macros are implemented here. The `\_inchap`, `\_insec` and `\_insecc` macros do the real work, First, we read the optional parameter [⟨*label*⟩], if it exists. The `\chap`, `\sec` and `\secc` macro reads its parameter using `\scantoeol`. This causes that they cannot be used inside other macros. Use `\_inchap`, `\_insec`, and `\_insecc` macros directly in such case.

```
176 \_optdef\_chap[]{\_trylabel \_scantoeol\_inchap}
177 \_optdef\_sec []{\_trylabel \_scantoeol\_insec}
178 \_optdef\_secc[]{\_trylabel \_scantoeol\_insecc}
179 \_def\_trylabel{\_istoksempty\_opt\_iffalse \_label[\_the\_opt]\_fi}
180
181 \_def\_inchap #1{\_par \_sectionlevel=1
182    \_def \_savedtitle {#1}% saved to .ref file
183    \_ifnonum \_else {\_globaldefs=1 \_incr\_chapnum \_chapx}\_fi
184    \_edef \_therefnum {\_ifnonum \_space \_else \_thechapnum \_fi}%
185    \_printchap{\_scantextokens{#1}}%
186    \_resetnonumnotoc
187 }
188 \_def\_insec #1{\_par \_sectionlevel=2
189    \_def \_savedtitle {#1}% saved to .ref file
190    \_ifnonum \_else {\_globaldefs=1 \_incr\_secnum \_secx}\_fi
191    \_edef \_therefnum {\_ifnonum \_space \_else \_thesecnum \_fi}%
192    \_printsec{\_scantextokens{#1}}%
193    \_resetnonumnotoc
194 }
195 \_def\_insecc #1{\_par \_sectionlevel=3
196    \_def \_savedtitle {#1}% saved to .ref file
197    \_ifnonum \_else {\_globaldefs=1 \_incr\_seccnum \_seccx}\_fi
198    \_edef \_therefnum {\_ifnonum \_space \_else \_theseccnum \_fi}%
199    \_printsecc{\_scantextokens{#1}}%
200    \_resetnonumnotoc
201 }
202 \_public \chap \sec \secc ;
```

The `\_printrefnum`[⟨*pre*⟩@⟨*post*⟩] macro is used in `\_print*` macros.

Note that the ⟨*tite-text*⟩ is `\detokenize`d before `\_wref`, so the problem of "fragile macros" from old LaTeX never occurs. This fourth parameter is not delimited by {...} but by end of line. This gives possibility to have unbalanced braces in inline verbatim in titles.

```
213 \_def \_printrefnum [#1@#2]{\_leavevmode % we must be in horizontal mode
214    \_ifnonum \_else #1\_therefnum #2\_fi
215    \_wlabel \_therefnum  % references, if `\label[<label>]` is declared
216    \_ifnotoc \_else \_incr \_tocrefnum
217       \_dest[toc:\_the\_tocrefnum]%
218       \_ewref\_Xtoc{{\_the\_sectionlevel}{\_secinfo}%
219                  {\_therefnum}{\_theoutline}\_detokenize\_ea{\_savedtitle}}%
220    \_fi
221    \_gdef\_theoutline{}%
222 }
```

`\thisoutline`{⟨*text*⟩} saves text to the `\_theoutline` macro. `\_printrefnum` uses it and removes it.

```
229 \_def\_theoutline{}
230 \_def\_thisoutline#1{\_gdef\_theoutline{#1}}
231 \_public \thisoutline ;
```

The `\_abovetitle`{⟨*penaltyA*⟩}{⟨*skipA*⟩} and `\_belowtitle`{⟨*skipB*⟩} pair communicates using a special penalty 11333 in vertical mode. The `\_belowtitle` puts the vertical skip (its value is saved in `\_savedtitleskip`) followed by this special penalty. The `\_abovetitle` reads `\lastpenalty` and if it has this special value then it removes the skip used before and doesn't use the parameter. The `\abovetitle` creates ⟨*skipA*⟩ only if whatever previous skip is less or equal than ⟨*skipA*⟩. We must save ⟨*whatever-skip*⟩, remove it, create ⟨*penaltyA*⟩ (if `\_belowtitle` does not precede) and create ⟨*whatever-skip*⟩ or ⟨*skipA*⟩ depending on what is greater. The amount of ⟨*skipA*⟩ is measured using `\setbox0=\vbox`.

```
247 \_newskip \_savedtitleskip
248 \_newskip \_savedlastskip
249 \_def\_abovetitle #1#2{\_savedlastskip=\_lastskip % <whatever-skip>
250    \_ifdim\_lastskip>\_zo \_vskip-\_lastskip \_fi
```

```
251    \_ifnum\_lastpenalty=11333 \_vskip-\_savedtitleskip \_else #1\_fi
252    \_ifdim\_savedlastskip>\_zo \_setbox0=\_vbox{#2\_global\_tmpdim=\_lastskip}%
253    \_else \_tmpdim=\_maxdimen \_fi
254    \_ifdim\_savedlastskip>\_tmpdim \vskip\_savedlastskip \_else #2\_fi
255 }
256 \_def\_belowtitle #1{#1\_global\_savedtitleskip=\_lastskip \_penalty11333 }
```

\nbpar sets \interlinepenaty value. \nl is "new line" in the text (or titles), but space in toc or headlines or outlines.

```
263 \_def\_nbpar{{\_interlinepenalty=10000\_endgraf}}
264
265 \_protected\_def\_nl{\_unskip\_hfil\_break}
266 \_regmacro {\_def\_nl{\_unskip\_space}} {\_def\_nl{\_unskip\_space}} {\_def\_nl{ }}
267 \_regmacro {\_def\nl{\_unskip\_space}}  {\_def\nl{\_unskip\_space}}  {\_def\nl{ }}
268
269 \_public \nbpar \nl ;
```

_firstnoindent puts a material to \everypar in order to next paragraph will be without indentation. It is useful after titles. If you dislike this feature then you can say \let_firtnoindent=\relax. The _wipeepar removes the material from \everypar.

```
278 \_def \_firstnoindent {\_global\_everypar={\_wipeepar \_setbox7=\_lastbox}}
279 \_def \_wipeepar {\_global\_everypar={}}
```

The \mark (for running heads) is used in _printsection only. We suppose that chapters will be printed after \vfil\break, so users can implement chapter titles for running headers directly by macros, no \mark mechanism is needed. But sections need \marks. And they can be mixed with chapter's running heads, of course.

The _insertmark{⟨*title text*⟩} saves \mark in the format {⟨*title-num*⟩} {⟨*title-text*⟩}, so it can be printed "as is" in \headline (see the space between them), or you can define a formating macro with two parameters for processing these data, if you need it.

```
294 \_def\_insertmark#1{\_mark{{\_ifnonum\_else\_therefnum\_fi} {\_unexpanded{#1}}}}
```

OpTeX sets \headline={} by default, so no running headings are printed. You can activate the running headings by following code, for example. See also issue 100.

```
\addto\_chapx {\globaldefs=0 \vfil\break % headline of previous chapter is printed
    \xdef\_runningchap {\_thechapnum: \unexpanded\_ea{\_savedtitle}}}
\def \formathead #1#2{\isempty{#1}\iffalse #1: #2\fi}
\headline = {%
    \ifodd \pageno
        \hfil \ea\formathead\firstmark{}{}%
    \else
        \ifx\_runningchap\_undefined \else Chapter \_runningchap \fi \hfil
    \fi
}
```

The \secl⟨*number*⟩ ⟨*title-text*⟩⟨*eol*⟩ should be used for various levels of sections (for example, when converting from Markdown to OpTeX). \secl1 is \chap, \secl2 is \sec, \secl3 is \secc and all more levels (for ⟨*number*⟩> 3) are printed by the common _seclp macro. It declares only a simple design. If there is a requirement to use such more levels then the book designer can define something different here.

```
322 \_def\_secl{\_afterassignment\_secla \_sectionlevel=}
323 \_def\_secla{\_ifcase\_sectionlevel
324    \_or\_ea\_chap\_or\_ea\_sec\_or\_ea\_secc\_else\_ea\_seclp\_fi}
325 \_eoldef\_seclp#1{\_par \_ifnum\_lastpenalty=0 \_removelastskip\_medskip\_fi
326    \_noindent{\_bf #1}\_vadjust{\_nobreak}\_nl\_ignorepars}
327 \_def\_ignorepars{\_isnextchar\_par{\_ignoresecond\_ignorepars}{}}
328
329 \_public \secl ;
```

The \caption/⟨*letter*⟩ increases _⟨*letter*⟩num counter, edefines _thecapnum as _the⟨*letter*⟩num and defines _thecaptitle as language-dependent word using _mtext, runs the _everycaption⟨*letter*⟩ tokens register. The group opened by \caption is finalized by first _par from an empty line or from

`\vskip` or from `\endinsert`. The `\_printcaption`⟨*letter*⟩ is called, it starts with printing of the caption. The `\cskip` macro inserts nonbreakable vertical space between the caption and the object.

```
344  \_def\_caption/#1{\_def\_tmpa{#1}\_nospaceafter \_capA}
345  \_optdef\_capA []{\_trylabel \_incaption}
346  \_def\_incaption {\_bgroup
347     \_ifcsname _\_tmpa num\_endcsname \_ea\_incr \_csname _\_tmpa num\_endcsname
348     \_else \_opwarning{Unknown caption /\_tmpa}\_fi
349     \_edef\_thecapnum {\_csname _the\_tmpa num\_endcsname}%
350     \_edef\_thecaptitle{\_mtext{\_tmpa}}%
351     \_ea\_the \_csname _everycaption\_tmpa\_endcsname
352     \_def\_par{\_nbpar\_egroup}%
353     \_cs{_printcaption\_tmpa}%
354  }
355  \_def \_cskip {\_par\_nobreak\_medskip} % space between caption and the object
356
357  \_public \caption \cskip ;
```

The `\_printcaptiont` and `\_printcaptionf` macros start in vertical mode. They switch to horizontal mode and use `\_wlabel\_thecapnum` (in order to make reference and hyperlink destination) a they can use:

- `\_thecaptitle` ... expands to the word Table or Figure (depending on the current language).
- `\_thecapnum` ... expands to `\the`⟨*letter*⟩`num` (caption number).

The `\_captionsep` inserts a separator between auto-generated caption number and the following caption text. Default separator is `\_enspace` but if the caption text starts with dot or colon, then the space is not inserted. A user can wite `\caption/t: My table` and "**Table 1.1:** My table" is printed. You can re-define the `\_captionsep` macro if you want to use another separator.

```
376  \_def \_printcaptiont {%
377     \_noindent \_wlabel\_thecapnum {\_bf\_thecaptitle~\_thecapnum}%
378     \_narrowlastlinecentered\_iindent \_futurelet\_next\_captionsep
379  }
380  \_def\_captionsep{\_ifx\_next.\_ea\_bfnext \_else\_ifx\_next:\_ea\_ea\_ea\_bfnext
381     \_else \_enspace \_fi\_fi}
382  \_def\_bfnext#1{{\_bf#1}}
383  \_let \_printcaptionf = \_printcaptiont % caption of figures = caption of tables
```

If you want to declare a new type of `\caption` with independent counter, you can use following lines, where `\caption/a` for Algorithms are declared:

```
\let\_printcaptiona = \_printcaptionf  \let\_everycaptiona = \_everycaptionf
\newcount\_anum  \addto\_secx {\_anum=0 }
\def\_theanum {\_othe\_chapnum.\_the\_secnum.\_the\_anum}
\sdef{_mt:a:en}{Algorithm}  \sdef{_mt:a:cs}{Algoritmus} % + your language...
```

The default format of `\caption` text is a paragraph in block narrower by `\_iindent` and with the last line is centered. This setting is done by the `\_narrowlastlinecentered` macro.

```
400  \_def\_narrowlastlinecentered#1{%
401     \_leftskip=#1plus1fil
402     \_rightskip=#1plus-1fil
403     \_parfillskip=0pt plus2fil\_relax
404  }
```

`\eqmark` is processed in display mode (we add `\eqno` primitive) or in internal mode when `\eqaligno` is used (we don't add `\eqno`).

```
411  \_optdef\_eqmark []{\_trylabel \_ineqmark}
412  \_def\_ineqmark{\_incr\_dnum
413     \_ifinner\_else\_eqno \_fi
414     \_wlabel\_thednum \_hbox{\_thednum}%
415  }
416  \_public \eqmark ;
```

The `\numberedpar` ⟨*letter*⟩{⟨*name*⟩} is implemented here.

```
422 \_newcount\_counterA \_newcount\_counterB \_newcount\_counterC
423 \_newcount\_counterD \_newcount\_counterE
424
425 \_def\_resetABCDE {\_counterA=0 \_counterB=0 \_counterC=0 \_counterD=0 \_counterE=0 }
426
427 \_def \_theAnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterA}
428 \_def \_theBnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterB}
429 \_def \_theCnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterC}
430 \_def \_theDnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterD}
431 \_def \_theEnum {\_othe\_chapnum.\_othe\_secnum.\_the\_counterE}
432
433 \_def\_numberedpar#1#2{\_ea \_incr \_csname _counter#1\_endcsname
434    \_def\_tmpa{#1}\_def\_tmpb{#2}\_numberedparparam}
435 \_optdef\_numberedparparam[]{%
436    \_ea \_printnumberedpar \_csname _the\_tmpa num\_ea\_endcsname\_ea{\_tmpb}}
437
438 \_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
```

```
456 \_def \_printnumberedpar #1#2{\_par
457    \_noindent\_wlabel #1%
458    {\_bf #2 #1\_istoksempty\_opt\_iffalse \_space \_the\_opt \_fi.}\_space
459    \_ignorespaces
460 }
```

2.27   Lists, items

```
3 \_codedecl \begitems {Lists: begitems, enditems <2021-03-10>} % preloaded in format
```

_aboveliskip is used above the list of items,
_belowliskip is used below the list of items and
_interliskip is used between items.
_listskipA is used as **\listskipamount** at level 1 of items.
_listskipB is used as **\listskipamount** at other levels.
_setlistskip sets the skip dependent on the current level of items

```
14 \_def\_aboveliskip {\_removelastskip \_penalty-100 \_vskip\_listskipamount}
15 \_def\_belowliskip {\_penalty-200 \_vskip\_listskipamount}
16 \_def\_interliskip {}
17 \_def\_listskipA {\_medskipamount}
18 \_def\_listskipB {0pt plus.5\_smallskipamount}
19
20 \_def\_setlistskip {%
21    \_ifnum \_ilevel = 1 \_listskipamount = \_listskipA \_relax
22    \_else  \_listskipamount = \_listskipB \_relax
23    \_fi}
```

The **\itemnum** is locally reset to zero in each group declared by **\begitems**. So nested lists are numbered independently. Users can set initial value of **\itemnum** to another value after **\beitems** if they want.
Each level of nested lists is indented by the new **\iindent** from left. The default item mark is **_printitem**.
The **\begitems** runs **_aboveliskip** only if we are not near below a title, where a vertical skip is placed already and where the **\penalty** 11333 is. It activates * and defines it as **_startitem**.
The **\enditems** runs _isnextchar_par{}{_noindent} thus the next paragraph is without indentation if there is no empty line between the list and this paragraph (it is similar behavior as after display math).

```
42 \_newcount\_itemnum  \_itemnum=0
43 \_newtoks\_printitem
44
45 \_def\_begitems{\_par
46    \_bgroup
47    \_advance \_ilevel by1
48    \_setlistskip
49    \_ifnum\_lastpenalty<10000 \_aboveliskip \_fi
50    \_itemnum=0 \_adef*{\_relax\_ifmmode*\_else\_ea\_startitem\_fi}
51    \_advance\_leftskip by\_iindent
52    \_printitem=\_defaultitem
53    \_the\_everylist \_relax
54 }
55 \_def\_enditems{\_par\_belowliskip\_egroup \_isnextchar\_par{}{\_noindent}}
56
57 \_def\_startitem{\_par \_ifnum\_itemnum>0 \_interliskip \_fi
58    \_advance\_itemnum by1
59    \_the\_everyitem \_noindent\_llap{\_the\_printitem}\_ignorespaces
60 }
61 \_public \begitems \enditems \itemnum ;
```

\novspaces sets **\listskipamount** to 0pt.

```
67 \_def\_novspaces {\_removelastskip \_listskipamount=0pt \_relax}
68 \_public \novspaces ;
```

Various item marks are saved in **_item:**⟨*letter*⟩ macros. You can re-define 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**.

```
79 \_def\_style#1{%
80    \_ifcsname _item:#1\_endcsname \_printitem=\ea{\_csname _item:#1\_endcsname}%
81    \_else \_printitem=\_defaultitem \_fi
82 }
83 \_sdef{_item:o}{\_raise.4ex\_hbox{$\_scriptscriptstyle\_bullet$} }
84 \_sdef{_item:-}{- }
85 \_sdef{_item:n}{\_the\_itemnum. }
86 \_sdef{_item:N}{\_the\_itemnum) }
87 \_sdef{_item:i}{(\_romannumeral\_itemnum) }
88 \_sdef{_item:I}{\_uppercase\_ea{\_romannumeral\_itemnum}\_kern.5em}
89 \_sdef{_item:a}{\_athe\_itemnum) }
90 \_sdef{_item:A}{\_uppercase\_ea{\_athe\_itemnum}) }
91 \_sdef{_item:x}{\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
92 \_sdef{_item:X}{\_raise.2ex\_fullrectangle{1ex}\_kern.5em}
```

_athe{⟨*num*⟩} returns the ⟨*num*⟩s lowercase letter from the alphabet.
_fullrectangle{⟨*dimen*⟩} prints full rectangle with given ⟨*dimen*⟩.

```
99  \_def\_fullrectangle#1{\_hbox{\_vrule height#1 width#1}}
100
101 \_def\_athe#1{\_ifcase#1?\_or a\_or b\_or c\_or d\_or e\_or f\_or g\_or h\_or
102    i\_or j\_or k\_or l\_or m\_or n\_or o\_or p\_or q\_or r\_or s\_or t\_or
103    u\_or v\_or w\_or x\_or y\_or z\_else ?\_fi
104 }
105 \_public \style ;
```

The **\begblock** macro selects fonts from footnotes **_fnset** and opens new indentation in a group. **\endblock** closes the group. This is implemented as an counterpart of Markdown's Blockquotes. Redefine these macros if you want to declare different design. The OpTeX trick 0031 shows how to create blocks with grey background splittable to more pages.

```
118 \_def\_begblock{\_bgroup\_fnset \_medskip \_advance\_leftskip by\_iindent \_firstnoindent}
119 \_def\_endblock{\_par\_medskip\_egroup\_isnextchar\_par{}{\_noindent}}
120
121 \_public \begblock \endblock ;
```

2.28 Verbatim, listings

2.28.1 Inline and "display" verbatim

```
3  \_codedecl \begtt {Verbatim <2022-04-23>} % preloaded in format
```

The internal parameters _ttskip, _ttpenalty, _viline, _vifile and _ttfont for verbatim macros are set.

```
11  \_def\_ttskip{\_medskip}              % space above and below \begtt, \verbinput
12  \_mathchardef\_ttpenalty=100          % penalty between lines in \begtt, \verbinput
13  \_newcount\_viline                    % last line number in \verbinput
14  \_newread\_vifile                     % file given by \verinput
15  \_def\_ttfont{\_tt}                    % default tt font
```

\code{⟨text⟩} expands to \detokenize{⟨text⟩} when \escapechar=-1. In order to do it more robust when it is used in \write then it expands as noexpanded \code⟨space⟩ (followed by space in its csname). This macro does the real work.

The _printinverbatim{⟨text⟩} macro is used for \code{⟨text⟩} printing and for `⟨text⟩` printing. It is defined as \hbox, so the in-verbatim ⟨text⟩ will be never broken. But you can re-define this macro.

When \code occurs in PDF outlines then it does the same as \detokenize. The macro for preparing outlines sets \escapechar to −1 and uses _regoul token list before \edef.

The \code is not \proteced because we want it expands to \unexpanded{\code⟨space⟩{⟨text⟩}} in \write parameters. This protect the expansions of the \code parameter (like \\, \^ etc.).

```
36  \_def\_code#1{\_unexpanded\_ea{\_csname _code \_endcsname{#1}}}
37  \_protected\_sdef{_code }#1{{\_escapechar=-1 \_ttfont \_the\_everyintt \_relax
38    \_ea\_printinverbatim\_ea{\_detokenize{#1}}}}
39  \_def\_printinverbatim#1{\_leavevmode\_hbox{#1}}
40
41  \_regmacro {}{}{\_let\code=\_detokenize \_let\_code=\_detokenize}
42  \_public \code ;
```

The _setverb macro sets all catcodes to "verbatim mode". It should be used only in a group, so we prepare a new catcode table with "verbatim" catcodes and we define it as _catcodetable_verbatimcatcodes. After the group is finished then original catcode table is restored.

```
51  \_newcatcodetable \_verbatimcatcodes
52  \_def\_setverb{\_begingroup
53    \_def\do##1{\_catcode`##1=12 }
54    \_dospecials
55    \_savecatcodetable\_verbatimcatcodes % all characters are normal
56    \_endgroup
57  }
58  \_setverb
59  \_def\_setverb{\_catcodetable\_verbatimcatcodes }%
```

\verbchar⟨char⟩ saves original catcode of previously declared ⟨char⟩ (if such character was declared) using _savedttchar and _savedttcharc values. Then new such values are stored. The declared character is activated by _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 \verbchar" is used. Then the following work is schematically done:

```
\_def "{\_begingroup \_setverb ... \_readverb}
\_def \_readverb #1"{\_printinverbatim{#1}\_endgroup}
```

Note that the second occurrence of " is not active because _setverb deactivates it.

```
78  \_def\_verbchar#1{%
79    \_ifx\_savedttchar\_undefined\_else \_catcode\_savedttchar=\_savedttcharc \_fi
80    \_chardef\_savedttchar=`#1
81    \_chardef\_savedttcharc=\_catcode`#1
82    \_adef{#1}{\_begingroup \_setverb \_adef{ }{\_dsp}\_ttfont \_the\_everyintt\_relax \_readverb}%
83    \_def\_readverb ##1#1{\_printinverbatim{##1}\_endgroup}%
84  }
85  \_let \_activettchar=\_verbchar % for backward compatibility
86  \_public \verbchar \activettchar ;
```

`\begtt` is defined only as public. We don't need a private `\_begtt` variant. This macro opens a group and sets `%` as an active character (temporary). This will allow it to be used as the comment character at the same line after `\begtt`. Then `\_begtti` is run. It is defined by `\eoldef`, so users can put a parameter at the same line where `\begtt` is. This `#1` parameter is used after `\everytt` parameters settings, so users can change them locally.

The `\_begtti` macro does `\_setverb` and another preprocessing, sets `\endlinechar` to `^^J` and reads the following text in verbatim mode until `\endtt` occurs. This scanning is done by `\_startverb` macro which is defined as:

    _def_startverb #1\endtt #2^^J{...}

We must to ensure that the backslash in `\endtt` has category 12 (this is a reason of the `\ea` chain in real code). The `#2` is something between `\endtt` and the end of the same line and it is simply ignored.

The `\_startverb` puts the scanned data to `\_prepareverbdata`. It sets the data to `\_tmpb` without changes by default, but you should re-define it in order to do special changes if you want. (For example, `\hisyntax` redefines this macro.) The scanned data have `^^J` at each end of line and all spaces are active characters (defined as `\␣`). Other characters have normal category 11 or 12.

The `^^J` is appended to verbatim data because we need to be sure that the data are finished by this character. When `\endtt` is preceded by spaces then we need to close these spaces by `^^J` and such line is not printed due to a trick used in `\_printverb`.

When `\_prepareverbdata` finishes then `\_startverb` runs `\_printverb` loop over each line of the data and does a final work: last skip plus `\noindent` in the next paragraph.

```
126 \_def\begtt{\_par \_begingroup \_adef\%##1\_relax{\_relax}\_begtti}
127 \_eoldef \_begtti#1{\_wipeepar \_setxhsize
128    \_vskip\_parskip \_ttskip
129    \_setverb
130    \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
131    \_adef{ }{\_dsp}\_adef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
132    \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
133    \_protrudechars=0 % disable protrusion
134    \_the\_everytt \_relax #1\_relax \_ttfont
135    \_def\_testcommentchars##1\_iftrue{\_iffalse}\_let\_hicomments=\_relax
136    \_savemathsb \_endlinechar=`^^J
137    \_startverb
138 }
139 \_ea\_def\_ea\_startverb \_ea#\_ea1\_csstring\\endtt#2^^J{%
140    \_prepareverbdata\_tmpb{#1^^J}%
141    \_ea\_printverb \_tmpb\_end
142    \_par \_restoremathsb
143    \_endgroup \_ttskip
144    \_isnextchar\_par{}{\_noindent}%
145 }
146 \_def\_prepareverbdata#1#2{\_def#1{#2}}
```

The `\_printverb` macro calls `\_printverbline{⟨line⟩}` repeatedly to each scanned line of verbatim text. The `\_printverb` is used from `\begtt...\endtt` and from `\verbinput` too.

The `\_testcommentchars` replaces the following `\_iftrue` to `\_iffalse` by default unless the `\commentchars` are set. So, the main body of the loop is written in the `\_else` part of the `\_iftrue` condition. The `\_printverbline{⟨line⟩}` is called here.

The `\_printverbline{⟨line⟩}` expects that it starts in vertical mode and it must do `\par` to return the vertical mode. The `\_printverblinenum` is used here: it does nothing when `\_ttline<0` else it prints the line number using `\_llap`.

`\_putttpenalty` puts `\_ttpenalty` before second and next lines, but not before first line in each `\begtt...\endtt` environment.

The `\_ttline` is increased here in the `\_printverb` macro because of comments-blocks: the `\_prinverbline` is not processed in comments-blocks but we need to count the `\_ttline`.

```
171 \_def\_printverb #1^^J#2{%
172    \_ifx\_printverblinenum\_relax \_else \_incr\_ttline \_fi
173    \_testcommentchars #1\_relax\_relax\_relax
174    \_iftrue
175       \_ifx\_end#2\_printcomments\_fi
176    \_else
```

```
177        \_ifx\_vcomments\_empty\_else  \_printcomments \_def\_vcomments{}\_fi
178        \_ifx\_end#2%
179            \_bgroup \_adef{ }{}\_def\t{}% if the last line is emtpy, we don't print it
180            \_ifcat&#1&\_egroup \_ifx\_printverblinenum\_relax \_else \_decr\_ttline \_fi
181            \_else\_egroup \_printverbline{#1}\_fi
182        \_else
183            \_printverbline{#1}%
184        \_fi
185    \_fi
186    \_unless\_ifx\_end#2\_afterfi{\_printverb#2}\_fi
187 }
188 \_def\_printverbline#1{\_putttpenalty \_indent \_printverblinenum \_kern\_ttshift #1\_par}
189 \_def\_initverblinenum{\_tenrm \_thefontscale[700]\_ea\_let\_ea\_sevenrm\_the\_font}
190 \_def\_printverblinenum{\_llap{\_sevenrm \_the\_ttline\_kern.9em}}
191 \_def\_putttpenalty{\_def\_putttpenalty{\_penalty\_ttpenalty}}
```

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 the 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. A similar preparation is done as in \begtt after the group is opened. Then we skip **_vinolines** lines in a loop a and we read **_vidolines** lines. The read data is accumulated into **_tmpb** macro. The next steps are equal to the steps done in **_startverb** macro: data are processed via **_prepareverbdata** and printed via **_printverb** loop.

```
207 \_def\_verbinput #1(#2) #3 {\_par \_def\_tmpa{#3}%
208    \_def\_tmpb{#1}%  cmds used in local group
209    \_ifx\_vifilename\_tmpa \_else
210        \_openin\_vifile={#3}%
211        \_global\_viline=0 \_global\_let\_vifilename=\_tmpa
212        \_ifeof\_vifile
213            \_opwarning{\_string\verbinput: file "#3" unable to read}
214            \_ea\_ea\_ea\_skiptorelax
215        \_fi
216    \_fi
217    \_viscanparameter #2+\_relax
218 }
219 \_def\_skiptorelax#1\_relax{}
220
221 \_def \_viscanparameter #1+#2\_relax{%
222    \_if$#2$\_viscanminus(#1)\_else \_viscanplus(#1+#2)\_fi
223 }
224 \_def\_viscanplus(#1+#2+){%
225    \_if$#1$\_tmpnum=\_viline
226    \_else \_ifnum#1<0 \_tmpnum=\_viline \_advance\_tmpnum by-#1
227        \_else \_tmpnum=#1
228            \_advance\_tmpnum by-1
229            \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi % (0+13) = (1+13)
230    \_fi \_fi
231    \_edef\_vinolines{\_the\_tmpnum}%
232    \_if$#2$\_def\_vidolines{0}\_else\_edef\_vidolines{#2}\_fi
233    \_doverbinput
234 }
235 \_def\_viscanminus(#1-#2){%
236    \_if$#1$\_tmpnum=0
237        \_else \_tmpnum=#1 \_advance\_tmpnum by-1 \_fi
238    \_ifnum\_tmpnum<0 \_tmpnum=0 \_fi  % (0-13) = (1-13)
239    \_edef\_vinolines{\_the\_tmpnum}%
240    \_if$#2$\_tmpnum=0
241        \_else \_tmpnum=#2 \_advance\_tmpnum by-\_vinolines \_fi
242    \_edef\_vidolines{\_the\_tmpnum}%
243    \_doverbinput
244 }
245 \_def\_doverbinput{%
246    \_tmpnum=\_vinolines
247    \_advance\_tmpnum by-\_viline
248    \_ifnum\_tmpnum<0
249        \_openin\_vifile={\_vifilename}%
```

134

```
250        \_global\_viline=0
251     \_else
252        \_edef\_vinolines{\_the\_tmpnum}%
253     \_fi
254     \_vskip\_parskip \_ttskip \_wipeepar \_setxhsize
255     \_begingroup
256     \_ifnum\_ttline<-1 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
257     \_setverb \_adef{ }{\_dsp}\_adef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
258     \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
259     \_protrudechars=0 % disable protrusion
260     \_the\_everytt\_relax \_tmpb\_relax \_ttfont
261     \_savemathsb \_endlinechar=`^^J \_tmpnum=0
262     \_loop \_ifeof\_vifile \_tmpnum=\_vinolines\_space \_fi
263         \_ifnum\_tmpnum<\_vinolines\_space
264         \_vireadline \_advance\_tmpnum by1 \_repeat     %% skip lines
265     \_edef\_ttlinesave{\_global\_ttline=\_the\_ttline}%
266     \_ifnum\_ttline=-1 \_ttline=\_viline \_else \_let\_ttlinesave=\_relax \_fi
267     \_tmpnum=0 \_def\_tmpb{}%
268     \_ifnum\_vidolines=0 \_tmpnum=-1 \_fi
269     \_ifeof\_vifile \_tmpnum=\_vidolines\_space \_fi
270     \_loop \_ifnum\_tmpnum<\_vidolines\_space
271         \_vireadline
272         \_ifnum\_vidolines=0 \_else\_advance\_tmpnum by1 \_fi
273         \_ifeof\_vifile \_tmpnum=\_vidolines\_space \_else \_visaveline \_fi %% save line
274         \_repeat
275     \_ea\_prepareverbdata \_ea \_tmpb\_ea{\_tmpb^^J}%
276     \_catcode`\ =10 \_catcode`\%=9 % used in \commentchars comments
277     \_ea\_printverb \_tmpb\_end
278     \_ttlinesave
279     \_par \_restoremathsb
280     \_endgroup
281     \_ttskip
282     \_isnextchar\_par{}{\_noindent}%
283 }
284 \_def\_vireadline{\_read\_vifile to \_tmp \_incr\_viline }
285 \_def\_visaveline{\_ea\_addto\_ea\_tmpb\_ea{\_tmp}}
286
287 \_public \verbinput ;
```

_savemathsb, _restoremathsb pair is used in \begtt...\endtt or in \verbinput to temporary suppress the \mathsbon because we don't need to print \int _a in verbatim mode if \int_a is really written. The _restoremathsb is defined locally as \mathsbon only if it is needed.

```
297 \_def\_savemathsb{\_ifmathsb \_mathsboff \_def\_restoremathsb{\_mathsbon}\_fi}
298 \_def\_restoremathsb{}
```

If the language of your code printed by \verbinput supports the format of comments started by two characters from the beginning of the line then you can set these characters by \commentchars⟨first⟩⟨second⟩. Such comments are printed in the non-verbatim mode without these two characters and they look like the verbatim printing is interrupted at the places where such comments are. See the section 2.39 for good illustration. The file optex.lua is read by a single command \verbinput (4-) optex.lua here and the \commentchars -- was set before it.

If you need to set a special character by \commentchars then you must to set the catcode to 12 (and space to 13). Examples:

```
\commentchars //        % C++ comments
\commentchars --        % Lua comments
{\catcode`\%=12 \_ea}\commentchars %%              % TeX comments
{\catcode`\#=12 \catcode`\ =13 \_ea}\commentchars#{ }  % bash comments
```

There is one limitation when TeX interprets the comments declared by \commentchars. Each block of comments is accumulated to one line and then it is re-interpreted by TeX. So, the ends of lines in the comments block are lost. You cannot use macros which need to scan end of lines, for example \begtt...\endtt inside the comments. The character % is ignored in comments but you can use \% for printing or % alone for de-activating _endpar from empty comment lines.

Implementation: The \commentchars⟨first⟩⟨second⟩ redefines the _testcommentchars used in _printverb in order to it removes the following _iftrue and returns _iftrue or _iffalse depend-

ing on the fact that the comment characters are or aren't present at the beginning of tested line. If it is true (\ifnum expands to \ifnum 10>0) then the rest of the line is added to the _vcomments macro.

The _hicomments is \relax by default but it is redefined by \commentchars in order to keep no-colorized comments if we need to use feature from \commentchars.

The accumulated comments are printed whenever the non-comment line occurs. This is done by _printcomments macro. You can re-define it, but the main idea must be kept: it is printed in the group, _reloding _rm initializes normal font, \catcodetable0 returns to normal catcode table used before \verbinput is started, and the text accumulated in _vcomments must be printed by _scantextokens primitive.

```
350  \_def\_vcomments{}
351  \_let\_hicomments=\_relax
352
353  \_def\_commentchars#1#2{%
354      \_def\_testcommentchars ##1##2##3\_relax ##4\_iftrue{\_ifnum % not closed in this macro
355          \_ifx #1##1\_ifx#2##21\_fi\_fi 0>0
356          \_ifx\_relax##3\_relax \_addto\_vcomments{\_endgraf}% empty comment=\enfgraf
357          \_else \_addto\_vcomments{##3 }\_fi}%
358      \_def\_hicomments{\_replfromto{\b\n#1#2}{^^J}{\w{#1#2####1}^^J}}% used in \hisyntax
359  }
360  \_def\_testcommentchars #1\_iftrue{\_iffalse} % default value of \_testcommentchar
361  \_def\_printcomments{\_ttskip
362      {\_catcodetable0 \_rm \_everypar={}%
363       \_noindent \_ignorespaces \_scantextokens\_ea{\_vcomments}\_par}%
364      \_ttskip
365  }
366  \_public \commentchars ;
```

The \visiblesp sets spaces as visible characters ␣. It redefines the _dsp, so it is useful for verbatim modes only.

The _dsp is equivalent to \␣ primitive. It is used in all verbatim environments: spaces are active and defined as _dsp here.

```
377  \_def \_visiblesp{\_ifx\_initunifonts\_relax \_def\_dsp{\_char9251 }%
378                   \_else \_def\_dsp{\_char32 }\_fi}
379  \_let\_dsp=\  % primitive "direct space"
380
381  \_public \visiblesp ;
```

2.28.2 Listings with syntax highlighting

The user can write

```
\begtt \hisyntax{C}
...
\endtt
```

to colorize the code using C syntax. The user can also write \everytt={\hisyntax{C}} to have all verbatim listings colorized.

\hisyntax{⟨name⟩} reads the file hisyntax-⟨name⟩.opm where the colorization is declared. The parameter ⟨name⟩ is case insensitive and the file name must include it in lowercase letters. For example, the file hisyntax-c.opm looks like this:

```
3   \_codedecl \_hisyntaxc {Syntax highlighting for C sources <2020-04-03>}
4
5   \_newtoks \_hisyntaxc  \_newtoks \_hicolorsc
6
7   \_global\_hicolorsc={%     colors for C language
8       \_hicolor K \Red      % Keywords
9       \_hicolor S \Magenta  % Strings
10      \_hicolor C \Green    % Comments
11      \_hicolor N \Cyan     % Numbers
12      \_hicolor P \Blue     % Preprocessor
13      \_hicolor O \Blue     % Non-letters
14  }
15  \_global\_hisyntaxc={%
```

```
16     \_the\_hicolorsc
17     \_let\c=\_relax \_let\e=\_relax \_let\o=\_relax
18     \_replfromto {/*}{*/}        {\x C{/*#1*/}}%   /*...*/
19     \_replfromto {//}{^^J}       {\z C{//#1}^^J}%  //...
20     \_replfromto {\_string#}{^^J} {\z P{\##1}^^J}%  #include ...
21     \_replthis   {\_string\"}    {{\_string\"}}%   \" protected inside strings
22     \_replfromto {"}{"}          {\x S{"#1"}}%     "..."
23     %
24     \_edef\_tmpa {()\_string{\_string}+-*/=[]<>,:;\_pcent\_string&\_string^|!?}% non-letters
25     \_ea \_foreach \_tmpa
26       \_do {\_replthis{#1}{\n\o#1\n}}
27     \_foreach                                              % keywords
28       {auto}{break}{case}{char}{continue}{default}{do}{double}%
29       {else}{entry}{enum}{extern}{float}{for}{goto}{if}{int}{long}{register}%
30       {return}{short}{sizeof}{static}{struct}{switch}{typedef}{union}%
31       {unsigned}{void}{while}
32       \_do {\_replthis{\n#1\n}{\z K{#1}}}
33     \_replthis{.}{\n.\n}                                   % numbers
34     \_foreach 0123456789
35       \_do {\_replfromto{\n#1}{\n}{\c#1##1\e}}
36     \_replthis{\e.\c}{.}
37     \_replthis{\e.\n}{.\e}
38     \_replthis{\n.\c}{\c.}
39     \_replthis{e\e\o+\c}{e+}\_replthis{e\e\o-\c}{e-}
40     \_replthis{E\e\o+\c}{E+}\_replthis{E\e\o-\c}{E-}
41     \_def\o#1{\z O{#1}}
42     \_def\c#1\e{\z N{#1}}
43   }
```

OpTEX provides `hisyntax-{c,python,tex,html}.opm` files. You can take inspiration from these files and declare more languages.

Users can re-declare default colors by `\hicolors`={⟨*list of color declarations*⟩}. This value has precedence over `\_hicolors`⟨*name*⟩ values declared in the `hicolors-`⟨*name*⟩`.opm` file. For example `\hicolors`={`\hicolor S \Brown`} causes all strings in brown color.

Another way to set non-default colors is to declare `\newtoks\hicolors`⟨*name*⟩ (without the `_` prefix) and set the color palette there. It has precedence before `\_hicolors`⟨*name*⟩ (with the `_` prefix) declared in the `hicolors-`⟨*name*⟩`.opm` file. You must re-declare all colors used in the corresponding `hisyntax-`⟨*name*⟩`.opm` file.

Notes for hi-syntax macro writers

The file `hisyntax-`⟨*name*⟩`.opm` is read only once and in a TEX group. If there are definitions then they must be declared as global.

The file `hisyntax-`⟨*name*⟩`.opm` must (globally) declare `\_hisyntax`⟨*name*⟩ token list where the action over verbatim text is declared typically by using the `\replfromto` or `\replthis` macros.

The verbatim text is prepared by the *pre-processing phase*, then `\_hisyntax`⟨*name*⟩ is applied and then the *post-processing phase* does final corrections. Finally, the verbatim text is printed line by line.

The pre-processing phase does:

- Each space is replaced by `\n\ \n`, so `\n`⟨*word*⟩`\n` is the pattern for matching whole words (no subwords). The `\n` control sequence is removed in the post-processing phase.
- Each end of line is represented by `\n^^J\n`.
- The `\_start` control sequence is added before the verbatim text and the `\_end` control sequence is appended to the end of the verbatim text. Both are removed in the post-processing phase.

Special macros are working only in a group when processing the verbatim text.

- `\n` represents nothing but it should be used as a boundary of words as mentioned above.
- `\t` represents a tabulator. It is prepared as `\n\t\n` because it can be at the boundary word boundary.
- `\x` ⟨*letter*⟩{⟨*text*⟩} can be used as replacing text. Consider the example

        \replfromto{/*}{*/}{\x C{/*#1*/}}

This replaces all C comments `/*...*/` by `\x C{/*...*/}`. But C comments may span multiple lines, i.e. the `^^J` should be inside it.

The macro `\x` ⟨*letter*⟩{⟨*text*⟩} is replaced by one or more occurrences of `\z` ⟨*letter*⟩{⟨*text*⟩} in the post-processing phase, each parameter ⟨*text*⟩ of `\z` is from from a single line. Parameters not

crossing line boundary are represented by `\x C{`⟨*text*⟩`}` and replaced by `\z C{`⟨*text*⟩`}` without any change. But:

> `\x C{`⟨*text1*⟩`^^J`⟨*text2*⟩`^^J`⟨*text3*⟩`}`

is replaced by

> `\z C{`⟨*text1*⟩`}^^J\z C{`⟨*text2*⟩`}^^J\z C{`⟨*text3*⟩`}`

`\z` ⟨*letter*⟩`{`⟨*text*⟩`}` is expanded to `\_z:`⟨*letter*⟩`{`⟨*text*⟩`}` and if `\hicolor` ⟨*letter*⟩ ⟨*color*⟩ is declared then `\_z:`⟨*letter*⟩`{`⟨*text*⟩`}` expands to `{`⟨*color*⟩⟨*text*⟩`}`. So, required color is activated for each line separately (e.g. for C comments spanning multiple lines).

- `\y {`⟨*text*⟩`}` is replaced by `\`⟨*text*⟩ in the post-processing phase. It should be used for macros without a parameters. You cannot use unprotected macros as replacement text before the post-processing phase, because the post-processing phase is based on the expansion of the whole verbatim text.

```
3 \_codedecl \hisyntax {Syntax highlighting of verbatim listings <2022-04-04>} % preloaded in format
```

The macros `\replfromto` and `\replthis` manipulate the verbatim text that is already stored in the `\_tmpb` macro.

`\replfromto` `{`⟨*from*⟩`}{`⟨*to*⟩`}{`⟨*replacement*⟩`}` finds the first occurrence of ⟨*from*⟩ and the first occurrence of ⟨*to*⟩ following it. The ⟨*text*⟩ between them is packed into `#1` and available to ⟨*replacement*⟩ which ultimately replaces ⟨*text*⟩.

`\replfromto` continues by finding next ⟨*from*⟩, then, next ⟨*to*⟩ repeatedly over the whole verbatim text. If the verbatim text ends with opening ⟨*from*⟩ but has no closing ⟨*to*⟩, then ⟨*to*⟩ is appended to the verbatim text automatically and the last part of the verbatim text is replaced too.

The first two parameters are expanded before use of `\replfromto`. You can use `\csstring\%` or something else here.

```
23 \_def\_replfromto #1#2{\_edef\_tmpa{{#1}{#2}}\_ea\_replfromtoE\_tmpa}
24 \_def\_replfromtoE#1#2#3{% #1=from #2=to #3=replacement
25    \_def\_replfrom##1#1##2{\_addto\_tmpb{##1}%
26        \_ifx\_end##2\_ea\_replstop \_else \_afterfi{\_replto##2}\_fi}%
27    \_def\_replto##1#2##2{%
28        \_ifx\_end##2\_afterfi{\_replfin##1}\_else
29          \_addto\_tmpb{#3}%
30          \_afterfi{\_replfrom##2}\_fi}%
31    \_def\_replfin##1#1\_end{\_addto\_tmpb{#3}\_replstop}%
32    \_edef\_tmpb{\_ea}\_ea\_replfrom\_tmpb#1\_end#2\_end\_end\_relax
33 }
34 \_def\_replstop#1\_end\_relax{}
35 \_def\_finrepl{}
```

The `\replthis` `{`⟨*pattern*⟩`}{`⟨*replacement*⟩`}` replaces each ⟨*pattern*⟩ by ⟨*replacement*⟩. Both parameters of `\replthis` are expanded first.

```
43 \_def\_replthis#1#2{\_edef\_tmpa{{#1}{#2}}\_ea\_replstring\_ea\_tmpb \_tmpa}
44
45 \_public \replfromto \replthis ;
```

The patterns ⟨*from*⟩, ⟨*to*⟩ and ⟨*pattern*⟩ are not found when they are hidden in braces `{...}`. E.g.

> `\replfromto{/*}{*/}{\x C{/*#1*/}}`

replaces all C comments by `\x C{...}`. The patterns inside `{...}` are not used by next usage of `\replfromto` or `\replthis` macros.

The `\_xscan` macro replaces occurrences of `\x` by `\z` in the post-processing phase. The construct `\x` ⟨*letter*⟩`{`⟨*text*⟩`}` expands to `\_xscan {`⟨*letter*⟩`}`⟨*text*⟩`^^J^`. If `#3` is `\_end` then it signals that something wrong happens, the ⟨*from*⟩ was not terminated by legal ⟨*to*⟩ when `\replfromto` did work. We must to fix this by using the `\_xscanR` macro.

```
63 \_def\_xscan#1#2^^J#3{\_ifx\_end#3 \_ea\_xscanR\_fi
64    \z{#1}{#2}%
65    \_ifx^#3\_else ^^J\_afterfi{\_xscan{#1}#3}\_fi}
66 \_def\_xscanR#1\_fi#2^{^^J}
```

The `\hicolor` ⟨*letter*⟩ ⟨*color*⟩ defines `\_z:`⟨*letter*⟩`{`⟨*text*⟩`}` as `{`⟨*color*⟩⟨*text*⟩`}`. It should be used in the context of `\x` ⟨*letter*⟩`{`⟨*text*⟩`}` macros.

```
74 \_def\_hicolor #1#2{\_sdef{_z:#1}##1{{#2##1}}}
```

`\hisyntax{`⟨*name*⟩`}` re-defines default `\_prepareverbdata`⟨*macro*⟩⟨*verbtext*⟩, but in order to do it does more things: It saves ⟨*verbtext*⟩ to `\_tmpb`, appends `\n` around spaces and `^^J` characters in pre-processing phase, opens `hisyntax-`⟨*name*⟩`.opm` file if `\_hisyntax`⟨*name*⟩ is not defined. Then `\_the\_hisyntax`⟨*name*⟩ is processed. Finally, the post-processing phase is realized by setting appropriate values to the `\x` and `\y` macros and doing `\_edef\_tmpb{\_tmpb}`.

```
87  \_def\_hisyntax#1{\_def\_prepareverbdata##1##2{%
88     \_let\n=\_relax \_let\b=\_relax \_def\t{\n\_noexpand\t\n}\_let\_start=\_relax
89     \_adef{ }{\n\_noexpand\ \n}\_edef\_tmpb{\_start^^J##2\_end}%
90     \_replthis{^^J}{\n^^J\b\n}\_replthis{\b\n\_end}{\_end}%
91     \_let\x=\_relax  \_let\y=\_relax \_let\z=\_relax \_let\t=\_relax
92     \_hicomments % keeps comments declared by \commentchars
93     \_endlinechar=`\^^M
94     \_lowercase{\_def\_tmpa{#1}}%
95     \_ifcsname _hialias:\_tmpa\_endcsname \_edef\_tmpa{\_cs{_hialias:\_tmpa}}\_fi
96     \_ifx\_tmpa\_empty \_else
97        \_unless \_ifcsname _hisyntax\_tmpa\_endcsname
98           \_isfile{hisyntax-\_tmpa.opm}\_iftrue \_opinput {hisyntax-\_tmpa.opm} \_fi\_fi
99        \_ifcsname _hisyntax\_tmpa\_endcsname
100          \_ifcsname hicolors\_tmpa\_endcsname
101             \_cs{hicolors\_tmpa}=\_cs{hicolors\_tmpa}%
102          \_fi
103          \_ea\_the \_csname _hisyntax\_tmpa\_endcsname % \_the\_hisyntax<name>
104          \_the\_hicolors  % colors which have precedece
105       \_else\_opwarning{Syntax "\_tmpa" undeclared (no file hisyntax-\_tmpa.opm)}
106    \_fi\_fi
107    \_replthis{\_start\n^^J}{}\_replthis{^^J\_end}{^^J}%
108    \_def\n{}\_def\b{}\_adef{ }{\_dsp}%
109    \_bgroup \_lccode`\~=`\ \_lowercase{\_egroup\_def\ {\_noexpand~}}%
110    \_def\w####1{####1}\_def\x####1####2{\_xscan{####1}####2^^J^}%
111    \_def\y####1{\_ea \_noexpand \_csname ####1\_endcsname}%
112    \_edef\_tmpb{\_tmpb}%
113    \_def\z####1{\_cs{_z:####1}}%
114    \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
115    \_localcolor
116 }}
117 \_public \hisyntax \hicolor ;
```

Aliases for languages can be declared like this. When `\hisyntax{xml}` is used then this is the same as `\hisyntax{html}`.

```
124 \_sdef{_hialias:xml}{html}
125 \_sdef{_hialias:json}{c}
```

2.29   Graphics

The `\inspic` is defined by `\pdfximage` and `\pdfrefximage` primitives. If you want to use one picture more than once in your document, then the following code is recommended:

```
\newbox\mypic
\setbox\mypic = \hbox{\picw=3cm \inspic{⟨picture⟩}}

My picture: \copy\mypic, again my picture: \copy\mypic, etc.
```

This code downloads the picture data to the PFD output only once (when `\setbox` is processed). Each usage of `\copy\mypic` puts only a pointer to the picture data in the PDF.

If you want to copy the same picture in different sizes, then choose a "basic size" used in `\setbox` and all different sizes can be realized by the `\transformbox{`⟨*transformation*⟩`}{\copy\mypic}`.

```
3  \_codedecl \inspic {Graphics <2022-03-05>} % 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.

```
17 \_def\_inspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inspicB\_inspicA}
18 \_def\_inspicA #1 {\_inspicB {#1}}
19 \_def\_inspicB #1{%
20     \_pdfximage \_ifdim\_picwidth=\_zo  \_else width\_picwidth\_fi
21                 \_ifdim\_picheight=\_zo \_else height\_picheight\_fi
22                 \_picparams {\_the\_picdir#1}%
23     \_pdfrefximage\_pdflastximage\_egroup}
24
25 \_def\_picparams{}
26
27 \_public \inspic ;
```

Inkscape can 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 after `*.pdf` is included in order to place labels of this picture in the same font as the document is printed. We need to read this LaTeX file by plain TeX macros when `\inkinspic` is used. These macros are stored in the `\_inkdefs` tokens list and it is used locally in the group. The solution is borrowed from OPmac trick 0032.

```
39 \_def\_inkinspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}
40 \_def\_inkinspicA #1 {\_inkinspicB {#1}}
41 \_def\_inkinspicB #1{%
42   \_ifdim\_picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
43   \_tmptoks={#1}%
44   \_the\_inkdefs
45   \_opinput {\_the\_picdir #1_tex}% file with labels
46   \_egroup}
47
48 \_newtoks\_inkdefs  \_inkdefs={%
49   \_def\makeatletter#1\makeatother{}%
50   \_def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{\_the\_tmptoks}\_hss}%
51   \_def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
52   \_def\put(#1,#2)#3{\_nointerlineskip\_vbox to\_zo{\_vss\_hbox to\_zo{\_kern#1\_picwidth
53     \_pdfsave\_hbox to\_zo{#3}\_pdfrestore\_hss}\_kern#2\_picwidth}}%
54   \_def\begin#1{\_csname _begin#1\_endcsname}%
55   \_def\_beginpicture(#1,#2){\_vbox\_bgroup
56     \_hbox to\_picwidth{}\_kern#2\_picwidth \_def\end##1{\_egroup}}%
57   \_def\_begintabular[#1]#2#3\end#4{%
58     \_vtop{\_def\\{\_cr}\_tabiteml{}\_tabitemr{}\_table{#2}{#3}}}%
59   \_def\color[#1]#2{\_scancolor #2,}%
60   \_def\_scancolor#1,#2,#3,{\_pdfliteral{#1 #2 #3 rg}}%
61   \_def\makebox(#1)[#2]#3{\_hbox to\_zo{\_csname _mbx:#2\_endcsname{#3}}}%
62   \_sdef{_mbx:lb}#1{#1\_hss}\_sdef{_mbx:rb}#1{\_hss#1}\_sdef{_mbx:b}#1{\_hss#1\_hss}%
63   \_sdef{_mbx:lt}#1{#1\_hss}\_sdef{_mbx:rt}#1{\_hss#1}\_sdef{_mbx:t}#1{\_hss#1\_hss}%
64   \_def\rotatebox#1#2{\_pdfrotate{#1}#2}%
65   \_def\lineheight#1{}%
66   \_def\setlength#1#2{}%
67 }
68 \_public \inkinspic ;
```

`\pdfscale{`⟨*x-scale*⟩`}{`⟨*y-scale*⟩`}` and `\pdfrotate{`⟨*degrees*⟩`}` macros are implemented by `\pdfsetmatrix` primitive. We need to know the values of sin, cos function in the `\pdfrotate`. We use Lua code for this.

```
77 \_def\_pdfscale#1#2{\_pdfsetmatrix{#1 0 0 #2}}
78
79 \_def\_gonfunc#1#2{%
80     \_directlua{tex.print(string.format('\_pcent.4f',math.#1(3.14159265*(#2)/180)))}%
81 }
82 \_def\_sin{\_gonfunc{sin}}
83 \_def\_cos{\_gonfunc{cos}}
84
85 \_def\_pdfrotate#1{\_pdfsetmatrix{\_cos{#1} \_sin{#1} \_sin{(#1)-180} \_cos{#1}}}
86
87 \_public \pdfscale \pdfrotate ;
```

The `\transformbox{⟨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.

```
101 \_def\_multiplyMxV #1 #2 #3 #4 {% matrix * (vvalX, vvalY)
102     \_tmpdim = #1\_vvalX \_advance\_tmpdim by #3\_vvalY
103     \_vvalY  = #4\_vvalY \_advance\_vvalY  by #2\_vvalX
104     \_vvalX = \_tmpdim
105 }
106 \_def\_multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
107     \_vvalX=#1pt \_vvalY=#2pt \_ea\_multiplyMxV \_currmatrix
108     \_edef\_tmpb{\_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY}%
109     \_vvalX=#3pt \_vvalY=#4pt \_ea\_multiplyMxV \_currmatrix
110     \_edef\_currmatrix{\_tmpb\_space
111         \_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY\_space}%
112 }
113 \_def\_transformbox#1#2{\_hbox{\_setbox0=\_hbox{{#2}}%
114     \_dimendef\_vvalX 11 \_dimendef\_vvalY 12 % we use these variables
115     \_dimendef\_newHt 13 \_dimendef\_newDp 14 % only in this group
116     \_dimendef\_newLt 15 \_dimendef\_newRt 16
117     \_preptransform{#1}%
118     \_kern-\_newLt \_vrule height\_newHt depth\_newDp width\_zo
119     \_setbox0=\_hbox{\_box0}\_ht0=\_zo \_dp0=\_zo
120     \_pdfsave#1\_rlap{\_box0}\_pdfrestore \_kern\_newRt}%
121 }
122 \_def\_preptransform #1{\_def\_currmatrix{1 0 0 1 }%
123     \_def\_pdfsetmatrix##1{\_edef\_tmpb{##1 }\_ea\_multiplyMxM \_tmpb\_unskip}%
124     \_let\pdfsetmatrix=\_pdfsetmatrix #1%
125     \_setnewHtDp 0pt  \_ht0  \_setnewHtDp 0pt  -\_dp0
126     \_setnewHtDp \_wd0 \_ht0  \_setnewHtDp \_wd0 -\_dp0
127     \_protected\_def \_pdfsetmatrix {\_pdfextension setmatrix}%
128     \_let\pdfsetmatrix=\_pdfsetmatrix
129 }
130 \_def\_setnewHtDp #1 #2 {%
131     \_vvalX=#1\_relax \_vvalY=#2\_relax \_ea\_multiplyMxV \_currmatrix
132     \_ifdim\_vvalX<\_newLt \_newLt=\_vvalX \_fi \_ifdim\_vvalX>\_newRt \_newRt=\_vvalX \_fi
133     \_ifdim\_vvalY>\_newHt \_newHt=\_vvalY \_fi \_ifdim-\_vvalY>\_newDp \_newDp=-\_vvalY \_fi
134 }
135
136 \_def\_rotbox#1#2{%
137     \_isequal{90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_ht0 \_tmpdim=\_dp0}{\_vfill}{#2}%
138     \_else \_isequal{-90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_dp0 \_tmpdim=\_ht0}{}{#2}%
139     \_else \_transformbox{\_pdfrotate{#1}}{#2}%
140     \_fi \_fi
141 }
142 \_def\_rotboxA #1#2#3#4{\_hbox{\_setbox0=\_hbox{{#4}}#2%
143     \_vbox to\_wd0{#3\_wd0=\_zo \_dp0=\_zo \_ht0=\_zo
144                 \_pdfsave\_pdfrotate{#1}\_box0\_pdfrestore\vfil}%
145     \_kern\_tmpdim
146 }}
147 \_public \transformbox \rotbox ;
```

`\_scantwodimens` scans two objects with the syntactic rule ⟨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 a natural dimension.

`\backgroundpic{⟨image-file⟩}` puts the image to the background of each page. It is used in the `\slides` style, for example.

```
166 \_def\_scantwodimens{%
167     \_directlua{tex.print(string.format('{\_pcent d}{\_pcent d}',
168             token.scan_dimen(),token.scan_dimen()))}%
```

```
169  }
170
171  \_def\_puttext{\_ea\_ea\_ea\_puttextA\_scantwodimens}
172  \_def\_puttextA#1#2#3{{\_setbox0=\_hbox{{#3}}\_dimen1=#1sp \_dimen2=#2sp \_puttextB}}
173  \_def\_puttextB{%
174     \_ifvmode
175        \_ifdim\_prevdepth>\_zo \_vskip-\_prevdepth \_relax \_fi
176        \_nointerlineskip
177     \_fi
178     \_wd0=\_zo \_ht0=\_zo \_dp0=\_zo
179     \_vbox to\_zo{\_kern-\_dimen2 \_hbox to\_zo{\_kern\_dimen1 \_box0\_hss}\_vss}}
180
181  \_def\_putpic{\_ea\_ea\_ea\_putpicA\_scantwodimens}
182  \_def\_putpicA#1#2{\_dimen1=#1sp \_dimen2=#2sp \_ea\_ea\_ea\_putpicB\_scantwodimens}
183  \_def\_putpicB#1#2#3{{\_setbox0=\_hbox{\_picwidth=#1sp \_picheight=#2sp \_inspic{#3}}\_puttextB}}
184
185  \_newbox\_bgbox
186  \_def\_backgroundpic#1{%
187     \_setbox\_bgbox=\_hbox{\_picwidth=\_pdfpagewidth \_picheight=\_pdfpageheight \_inspic{#1}}%
188     \_pgbackground={\_copy\_bgbox}
189  }
190  \_def\nospec{0pt}
191  \_public \puttext \putpic \backgroundpic ;
```

_circle{⟨x⟩}{⟨y⟩} creates an ellipse with ⟨x⟩ axis and ⟨y⟩ axis. The origin is in the center.
_oval{⟨x⟩}{⟨y⟩}{⟨roundness⟩} creates an oval with ⟨x⟩, ⟨y⟩ size and with the 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⟩, creates the ⟨curve⟩ and returns the current point back. All these macros are fully expandable and they can be used in the \pdfliteral argument.

graphics.opm
```
207  \def\_circle#1#2{\_expr{.5*(#1)} 0 m
208     \_expr{.5*(#1)} \_expr{.276*(#2)} \_expr{.276*(#1)} \_expr{.5*(#2)} 0 \_expr{.5*(#2)} c
209     \_expr{-.276*(#1)} \_expr{.5*(#2)} \_expr{-.5*(#1)} \_expr{.276*(#2)} \_expr{-.5*(#1)} 0 c
210     \_expr{-.5*(#1)} \_expr{-.276*(#2)} \_expr{-.276*(#1)} \_expr{-.5*(#2)} 0 \_expr{-.5*(#2)} c
211     \_expr{.276*(#1)} \_expr{-.5*(#2)} \_expr{.5*(#1)} \_expr{-.276*(#2)} \_expr{.5*(#1)} 0 c h}
212
213  \def\_oval#1#2#3{0 \_expr{-(#3)} m \_expr{#1} \_expr{-(#3)} l
214     \_expr{(#1)+.552*(#3)} \_expr{-(#3)} \_expr{(#1)+(#3)} \_expr{-.552*(#3)}
215                                          \_expr{(#1)+(#3)} 0 c
216     \_expr{(#1)+(#3)} \_expr{#2} l
217     \_expr{(#1)+(#3)} \_expr{(#2)+.552*(#3)} \_expr{(#1)+.552*(#3)} \_expr{(#2)+(#3)}
218                                          \_expr{#1} \_expr{(#2)+(#3)} c
219     0 \_expr{(#2)+(#3)} l
220     \_expr{-.552*(#3)} \_expr{(#2)+(#3)} \_expr{-(#3)} \_expr{(#2)+.552*(#3)}
221                                          \_expr{-(#3)} \_expr{#2} c
222     \_expr{-(#3)} 0 l
223     \_expr{-(#3)} \_expr{-.552*(#3)} \_expr{-.552*(#3)} \_expr{-(#3)}  0 \_expr{-(#3)} c h}
224
225  \def\_mv#1#2#3{1 0 0 1 \_expr{#1} \_expr{#2} cm #3 1 0 0 1 \_expr{-(#1)} \_expr{-(#2)} cm}
```

The \inoval{⟨text⟩} is an example of _oval usage.
The \incircle{⟨text⟩} is an example of _circle usage.
The \ratio, \lwidth, \fcolor, \lcolor, \shadow and \overlapmargins are parameters, they can be set by user in optional brackets [...]. For example \fcolor=\Red does _let_fcolorvalue=\Red and it means filling color.
The _setflcolors uses the _setcolor macro to separate filling (non-stroking) color and stroking color. The _coc macro means "create oval or circle" and it expands to the stroking primitve S or filling primitive f or boh B. Only boundary stroking is performed after \fcolor=\relax. You cannot combine \fcolor=\relax with \shadow=Y.

graphics.opm
```
242  \_newdimen \_lwidth
243  \_def\_fcolor{\_let\_fcolorvalue}
244  \_def\_lcolor{\_let\_lcolorvalue}
245  \_def\_shadow{\_let\_shadowvalue}
246  \_def\_overlapmargins{\_let\_overlapmarginsvalue}
247  \_def\ratio{\_isnextchar ={\_ratioA}{\_ratioA=}}
248  \_def\_ratioA =#1 {\_def\_ratiovalue{#1}}
```

```
249  \_def\_touppervalue#1{\_ifx#1n\_let#1=N\_fi}

250

251  \_def\_setflcolors#1{% use only in a group
252      \_def\_setcolor##1##2##3{##1 ##2}%
253      \_edef#1{\_fcolorvalue}%
254      \_def\_setcolor##1##2##3{##1 ##3}%
255      \_edef#1{#1\_space\_lcolorvalue\_space}%
256  }
257  \_optdef\_inoval[]{\_vbox\_bgroup
258      \_roundness=2pt \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
259      \_shadow=N \_overlapmargins=N \_hhkern=0pt \_vvkern=0pt
260      \_the\_ovalparams \_relax \_the\_opt \_relax
261      \_touppervalue\_overlapmarginsvalue \_touppervalue\_shadowvalue
262      \_ifx\_overlapmarginsvalue N%
263          \_advance\_hsize by-2\_hhkern \_advance\_hsize by-2\_roundness \_fi
264      \_setbox0=\_hbox\_bgroup\_bgroup \_aftergroup\_inovalA \_kern\_hhkern \_let\_next=%
265  }
266  \_def\_inovalA{\_egroup % of \setbox0=\hbox\bgroup
267      \_ifdim\_vvkern=\_zo \_else \_ht0=\_dimexpr\_ht0+\_vvkern \_relax
268                          \_dp0=\_dimexpr\_dp0+\_vvkern \_relax \_fi
269      \_ifdim\_hhkern=\_zo \_else \_wd0=\_dimexpr\_wd0+\_hhkern \_relax \_fi
270      \_ifx\_overlapmarginsvalue N\_dimen0=\_roundness \_dimen1=\_roundness
271      \_else                   \_dimen0=-\_hhkern   \_dimen1=-\_vvkern \_fi
272      \_setflcolors\_tmp
273      \_hbox{\_kern\_dimen0
274          \_vbox to\_zo{\_kern\_dp0
275              \_ifx\_shadowvalue N\_else
276                  \_edef\_tmpb{{\_bp{\_wd0+\_lwidth}}{\_bp{\_ht0+\_dp0+\_lwidth}}{\_bp{\_roundness}}}%
277                  \_doshadow\_oval
278              \_fi
279              \_pdfliteral{q \_bp{\_lwidth} w \_tmp
280                  \_oval{\_bp{\_wd0}}{\_bp{\_ht0+\_dp0}}{\_bp{\_roundness}} \_coc\_space Q}\_vss%
281          \_ht0=\_dimexpr\_ht0+\_dimen1 \_relax \_dp0=\_dimexpr\_dp0+\_dimen1 \_relax
282          \_box0
283          \_kern\_dimen0}%
284      \_egroup % of \vbox\bgroup
285  }
286  \_optdef\_incircle[]{\_vbox\_bgroup
287      \_ratio=1 \_fcolor=\Yellow \_lcolor=\Red \_lwidth=.5bp
288      \_shadow=N \_overlapmargins=N \_hhkern=3pt \_vvkern=3pt
289      \_ea\_the \_ea\_circleparams \_space \_relax
290      \_ea\_the \_ea\_opt \_space \_relax
291      \_touppervalue\_overlapmarginsvalue \_touppervalue\_shadowvalue
292      \_setbox0=\_hbox\_bgroup\_bgroup \_aftergroup\_incircleA \_kern\_hhkern \_let\_next=%
293  }
294  \_def\_incircleA {\_egroup % of \setbox0=\hbox\bgroup
295      \_wd0=\_dimexpr \_wd0+\_hhkern \_relax
296      \_ht0=\_dimexpr \_ht0+\_vvkern \_relax \_dp0=\_dimexpr \_dp0+\_vvkern \_relax
297      \_ifdim \_ratiovalue\_dimexpr \_ht0+\_dp0 > \_wd0
298          \_dimen3=\_dimexpr \_ht0+\_dp0 \_relax  \_dimen2=\_ratiovalue\_dimen3
299      \_else \_dimen2=\_wd0 \_dimen3=\_expr{1/\_ratiovalue}\_dimen2 \_fi
300      \_setflcolors\_tmp
301      \_ifx\_overlapmarginsvalue N\_dimen0=\_zo \_dimen1=\_zo
302      \_else \_dimen0=-\_hhkern \_dimen1=-\_vvkern \_fi
303      \_hbox{\_kern\_dimen0
304          \_ifx\_shadowvalue N\_else
305              \_edef\_tmpb{{\_bp{\_dimen2+\_lwidth}}{\_bp{\_dimen3+\_lwidth}}{}}%
306              \_doshadow\_circlet
307          \_fi
308          \_pdfliteral{q \_bp{\_lwidth} w \_tmp \_mv{\_bp{.5\_wd0}}{\_bp{(\_ht0-\_dp0)/2}}
309                                  {\_circle{\_bp{\_dimen2}}{\_bp{\_dimen3}} \_coc} Q}%
310          \_ifdim\_dimen1=\_zo \_else
311              \_ht0=\_dimexpr \_ht0+\_dimen1 \_relax \_dp0=\_dimexpr \_dp0+\_dimen1 \_relax \_fi
312          \_box0
313          \_kern\_dimen0}
314      \_egroup % of \vbox\bgroup
315  }
316  \_def\_circlet#1#2#3{\_circle{#1}{#2}}
317  \_def\_coc{\_ifx\_fcolorvalue\_relax S\_else \_ifdim\_lwidth=0pt f\_else B\_fi\_fi}
```

```
318
319  \_public \inoval \incircle \ratio \lwidth \fcolor \lcolor \shadow \overlapmargins ;
```

Just before defining shadows, which require special graphics states, we define means for managing these graphics states and other PDF page resources (graphics states, patterns, shadings, etc.). Our mechanism, defined mostly in Lua (see 2.39.4, uses single dictionary for each PDF page resource type (extgstate, etc.) for all pages (`\pdfpageresources` just points to it).

The macro `\addextgstate`{⟨*PDF name*⟩}{⟨*PDF dictionary*⟩} is a use of that general mechanism and shall be used for adding more graphics states. It must be used *after* `\dump`. It's general variant defined in Lua is `\_addpageresource` {⟨*resource type*⟩}{⟨*PDF name*⟩}{⟨*PDF dictionary*⟩}. You can use `\pageresources` or `\_pageresources` if you need to insert resource entries to manually created PDF XObjects.

```
                                                                         graphics.opm
337  \_def\_addextgstate{\_addpageresource{ExtGState}}
338
339  \_public \addextgstate ;
340  \_def\pageresources{\_pageresources}
341  \_def\addpageresource{\_addpageresource}
```

A shadow effect is implemented here. The shadow is equal to the silhouette of the given path in a gray-transparent color shifted by `\_shadowmoveto` vector and with blurred boundary. A waistline with the width 2*`\_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
352  \_def\_shadowlevels{9}          % number of layers for blurr effect
353  \_def\_shadowdarknessA{0.025}   % transparency of first shadowlevels/2 layers
354  \_def\_shadowdarknessB{0.07}    % transparency of second half of layers
355  \_def\_shadowmoveto{1.8 -2.5}   % vector defines shifting layer (in bp)
356  \_def\_shadowb{1}               % 2*shadowb = blurring area thickness
357
358  \_def\_insertshadowresources{%
359      \_addextgstate{op1}{<</ca \_shadowdarknessA>>}%
360      \_addextgstate{op2}{<</ca \_shadowdarknessB>>}%
361      \_glet\_insertshadowresources=\_relax
362  }
```

The `\_doshadow`{⟨*curve*⟩} does the shadow effect.

```
                                                                         graphics.opm
368  \_def\_doshadow#1{\_vbox{%
369      \_insertshadowresources
370      \_tmpnum=\_numexpr (\_shadowlevels-1)/2 \_relax
371      \_edef\_tmpfin{\_the\_tmpnum}%
372      \_ifnum\_tmpfin=0 \_def\_shadowb{0}\_def\_shadowstep{0}%
373      \_else \_edef\_shadowstep{\_expr{\_shadowb/\_tmpfin}}\_fi
374      \_def\_tmpa##1##2##3{\_def\_tmpb
375          {#1{##1+2*\_the\_tmpnum*\_shadowstep}{##2+2*\_the\_tmpnum*\_shadowstep}{##3}}}%
376      \_ea \_tmpa \_tmpb
377      \_def\_shadowlayer{%
378          \_ifnum\_tmpnum=0 /op2 gs \_fi
379          \_tmpb\_space f
380          \_immediateassignment\_advance\_tmpnum by-1
381          \_ifnum-\_tmpfin<\_tmpnum
382              \_ifx#1\_oval 1 0 0 1 \_shadowstep\_space \_shadowstep\_space cm \_fi
383              \_ea \_shadowlayer \_fi
384      }%
385      \_pdfliteral{q /op1 gs 0 g 1 0 0 1 \_shadowmoveto\_space cm
386          \_ifx#1\_circlet 1 0 0 1 \_expr{\_bp{.5\_wd0}} \_expr{\_bp{(\_ht0-\_dp0)/2}} cm
387          \_else  1 0 0 1 -\_shadowb\_space -\_shadowb\_space cm \_fi
388          \_shadowlayer Q}
389  }}
```

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.

```
400 \_def\_clipinpath#1#2#3#4{% #1=x-pos[bp], #2=y-pos[bp], #3=curve, #4=text
401     \_hbox{\_setbox0=\_hbox{{#4}}%
402         \_tmpdim=\_wd0 \_wd0=\_zo
403         \_pdfliteral{q \_mv{#1}{#2}{#3 W n}}%
404         \_box0\_pdfliteral{Q}\_kern\_tmpdim
405     }%
406 }
407
408 \_def\_clipinoval {\_ea\_ea\_ea\_clipinovalA\_scantwodimens}
409 \_def\_clipinovalA #1#2{%
410     \_def\_tmp{{#1/65781.76}{#2/65781.76}}%
411     \_ea\_ea\_ea\_clipinovalB\_scantwodimens
412 }
413 \_def\_clipinovalB{\_ea\_clipinovalC\_tmp}
414 \_def\_clipinovalC#1#2#3#4{%
415     \_ea\_clipinpath{#1-(#3/131563.52)+(\_bp{\_roundness})}{#2-(#4/131563.52)+(\_bp{\_roundness})}%
416     {\_oval{#3/65781.76-(\_bp{2\_roundness})}{#4/65781.76-(\_bp{2\_roundness})}{\_bp{\_roundness}}}%
417 }
418 \_def\_clipincircle {\_ea\_ea\_ea\_clipincircleA\_scantwodimens}
419 \_def\_clipincircleA #1#2{%
420     \_def\_tmp{{#1/65781.76}{#2/65781.76}}%
421     \_ea\_ea\_ea\_clipincircleB\_scantwodimens
422 }
423 \_def\_clipincircleB#1#2{%
424     \_ea\_clipinpath\_tmp{\_circle{#1/65781.76}{#2/65781.76}}%
425 }
426 \_public \clipinoval \clipincircle ;
```

2.30 The `\table` macro, tables and rules

2.30.1 The boundary declarator :

The ⟨*declaration*⟩ part of `\table{`⟨*declaration*⟩`}{`⟨*data*⟩`}` includes column declarators (letters) and other material: the `|` or (⟨*cmd*⟩). If the boundary declarator : is not used then the boundaries of columns are just before each column declarator with exception of the first one. For example, the declaration `{|c||c(xx)(yy)c}` should be written more exactly using the boundary declarator : by `{|c||:c(xx)(yy):c}`. But you can set these boundaries to other places using the boundary declarator : explicitly, for example `{|c:||c(xx):(yy)c}`. The boundary declarator : can be used only once between each pair of column declarators.

Each table item has its group. The (⟨*cmd*⟩) are parts of the given table item (depending on the boundary declarator position). If you want to apply a special setting for a given column, you can do this by (⟨*setting*⟩) followed by column declarator. But if the column is not first, you must use :(⟨*setting*⟩). Example. We have three centered columns, the second one have to be in bold font and the third one have to be in red: `\table{c:(\bf)c:(\Red)c}{`⟨*data*⟩`}`

2.30.2 Usage of the `\tabskip` primitive

The value of `\tabskip` primitive is used between all columns of the table. It is glue-type, so it can be stretchable or shrinkable, see next section 2.30.3.

By default, `\tabskip` is 0 pt. It means that only `\tabiteml`, `\tabitemr` and (⟨*cmds*⟩) can generate visual spaces between columns. But they are not real spaces between columns because they are in fact the part of the total column width.

The `\tabskip` value declared before the `\table` macro (or in `\everytable` or in `\thistable`) is used between all columns in the table. This value is equal to all spaces between columns. But you can set each such space individually if you use (`\tabskip=`⟨*value*⟩) in the ⟨*declaration*⟩ immediately before boundary character. The boundary character represents the column pair for which the `\tabskip` has individual value. For example `c(\tabskip=5pt):r` gives `\tabskip` value between `c` and `r` columns. You need not use boundary character explicitly, so `c(\tabskip=5pt)r` gives the same result.

Space before the first column is given by the `\tabskipl` and space after the last column is equal to `\tabskipr`. Default values are 0 pt.

Use nonzero `\tabskip` only in special applications. If `\tabskip` is nonzero then horizontal lines generated by `\crli`, `\crlli` and `\crlp` have another behavior than you probably expected: they are interrupted in each `\tabskip` space.

2.30.3 Tables to given width

There are two possibilities how to create tables to given width:

- `\table to`⟨*size*⟩`{`⟨*declaration*⟩`}{`⟨*data*⟩`}` uses stretchability or shrinkability of all spaces between columns generated by `\tabskip` value and eventually by `\tabskipl`, `\tabskipr` values. See example below.
- `\table pxto`⟨*size*⟩`{`⟨*declaration*⟩`}{`⟨*data*⟩`}` expands the columns declared by `p{`⟨*size*⟩`}`, if the ⟨*size*⟩ is given by a virtual `\tsize` unit. See the example below.

Example of `\table to`⟨*size*⟩:

```
\thistable{\tabskip=0pt plus1fil minus1fil}
\table to\hsize {lr}{⟨data⟩}
```

This table has its width `\hsize`. The first column starts at the left boundary of this table and it is justified left (to the boundary). The second column ends at the right boundary of the table and it is justified right (to the boundary). The space between them is stretchable and shrinkable to reach the given width `\hsize`.

Example of `\table pxto`⟨*size*⟩ (means "**p**aragraphs e**x**panded **to**"):

```
\table pxto\hsize {|c|p{\tsize}|}{\crl
 aaa          & Ddkas jd dsjds ds cgha sfgs dd fddzf dfhz xxz
                 dras ffg hksd kds d sdjds h sd jd dsjds ds cgha
                 sfgs dd fddzf dfhz xxz. \crl
 bb ddd ggg   & Dsjds ds cgha sfgs dd fddzf dfhz xxz
                 ddkas jd dsjds ds cgha sfgs dd fddzf. \crl }
```

aaa	Ddkas jd dsjds ds cgha sfgs dd fddzf dfhz xxz dras ffg hksd kds d sdjds h sd jd dsjds ds cgha sfgs dd fddzf dfhz xxz.
bb ddd ggg	Dsjds ds cgha sfgs dd fddzf dfhz xxz ddkas jd dsjds ds cgha sfgs dd fddzf.

The first `c` column is variable width (it gets the width of the most wide item) and the resting space to given `\hsize` is filled by the `p` column.

You can declare more than one `p{`⟨*coefficient*⟩`\tsize}` columns in the table when `pxto` keyword is used.

```
\table pxto13cm {r p{3.5\tsize} p{2\tsize} p{\tsize} l}{⟨data⟩}
```

This gives the ratio of widths of individual paragraphs in the table $3.5:2:1$.

2.30.4 \eqbox: boxes with equal width across the whole document

The `\eqbox` `[`⟨*label*⟩`]{`⟨*text*⟩`}` behaves like `\hbox{`⟨*text*⟩`}` in the first run of TeX. But the widths of all boxes with the same label are saved to `.ref` file and the maximum box width for each label is calculated at the beginning of the next TeX run. Then `\eqbox` `[`⟨*label*⟩`]{`⟨*text*⟩`}` behaves like `\hbox to` ⟨*dim:label*⟩ `{\hss` ⟨*text*⟩`\hss}`, where ⟨*dim:label*⟩ is the maximum width of all boxes labeled by the same `[`⟨*label*⟩`]`. The documentation of the LaTeX package `eqparbox` includes more information and tips.

The `\eqboxsize` `[`⟨*label*⟩`]{`⟨*dimen*⟩`}` expands to ⟨*dim:label*⟩ if this value is known, else it expands to the given ⟨*dimen*⟩.

The optional parameter `r` or `l` can be written before `[`⟨*label*⟩`]` (for example `\eqbox r[label]{text}`) if you want to put the text to the right or to the left side of the box width.

Try the following example and watch what happens after first TeX run and after the second one.

```
\def\leftitem#1{\par
   \noindent \hangindent=\eqboxsize[items]{2em}\hangafter=1
   \eqbox r[items]{#1 }\ignorespaces}

\leftitem {\bf first}      \lorem[1]
\leftitem {\bf second one} \lorem[2]
\leftitem {\bf final}      \lorem[3]
```

2.30.5 Implemetation of the `\table` macro and friends

```
3  \_codedecl \table {Basic macros for OpTeX <2022-05-03>} % 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`.

```
11  \_let\_tablebox=\_vbox
```

We save the to⟨*size*⟩ or pxto⟨*size*⟩ to #1 and `\_tableW` sets the to⟨*size*⟩ to the `\_tablew` macro. If pxto⟨*size*⟩ is used then `\_tablew` is empty and `\_tmpdim` includes given ⟨*size*⟩. The `\_ifpxto` returns true in this case.

The `\table` continues by reading {⟨*declaration*⟩} in the `\_tableA` macro. Catcodes (for example the | character) have to be normal when reading `\table` parameters. This is the reason why we use `\catcodetable` here.

```
24  \_newifi \_ifpxto
25  \_def\_table#1#{\_tablebox\_bgroup \_tableW#1\_empty\_end
26      \_bgroup \_catcodetable\_optexcatcodes \_tableA}
27  \_def\_tableW#1#2\_end{\_pxtofalse
28      \_ifx#1\_empty \_def\_tablew{}\_else
29      \_ifx#1p \_def\_tablew{}\_tableWx#2\_end \_else \_def\_tablew{#1#2}\_fi\_fi}
30  \_def\_tableWx xto#1\_end{\_tmpdim=#1\_relax \_pxtotrue}
31  \_public \table ;
```

The `\tablinespace` is implemented by enlarging given `\tabstrut` by desired dimension (height and depth too) and by setting `\_lineskip=-2\_tablinespace`. Normal table rows (where no `\hrule` is between them) have normal baseline distance.

The `\_tableA`{⟨*declaration*⟩} macro scans the ⟨*declaration*⟩ by `\_scantabdata#1\_relax` and continues by processing {⟨*data*⟩} by `\_tableB`. The trick `\_tmptoks={⟨data⟩}\_edef\_tmpb{\_the\_tmptoks}` is used here in order to keep the hash marks in the ⟨*data*⟩ unchanged.

```
44  \_def\_tableA#1{\_egroup
45      \_the\_thistable \_global\_thistable={}%
46      \_ea\_ifx\_ea^\_the\_tabstrut^\_setbox\_tstrutbox=\_null
47      \_else \_setbox\_tstrutbox=\_hbox{\_the\_tabstrut}%
48          \_setbox\_tstrutbox=\_hbox{\_vrule width\_zo
49              height\_dimexpr\_ht\_tstrutbox+\_tablinespace
50              depth\_dimexpr\_dp\_tstrutbox+\_tablinespace}%
51          \_offinterlineskip
52          \_lineskip=-2\_tablinespace
53      \_fi
54      \_colnum=0 \_let\_addtabitem=\_addtabitemx
55      \_def\_tmpa{}\_tabdata={\_colnum1\_relax}\_scantabdata#1\_relax
56      \_the\_everytable \_bgroup \_catcode`\#=12 \_tableB
57  }
```

The `\_tableB` saves ⟨*data*⟩ to `\_tmpb` and does `\replstring`s to prefix each macro `\crl` (etc.) by `\_crcr`. See `\_tabreplstrings`. The whole `\_tableB` macro is hidden in {...} in order to there may be `\table` in `\table` and we want to manipulate with & and `\cr` as with normal tokens in the `\_tabreplstrings`, not as the item delimiters of an outer `\table`.

The `\tabskip` value is saved for places between columns into the `\_tabskipmid` macro. Then it runs

> `\tabskip=\tabskipl \halign{`⟨*converted declaration*⟩`\tabskip=\tabskipr \cr` ⟨*data*⟩`\crcr}`

This sets the desired boundary values of `\tabskip`. The "between-columns" values are set as `\tabskip=\_tabskipmid` in the ⟨*converted declaration*⟩ immediately after each column declarator.

If pxto keyword was used, then we set the virtual unit `\tsize` to `-\hsize` first. Then the first attempt of the table is created in box 0. All collums where p{..\tsize} is used, are created as empty in this first pass. So, the `\wd0` is the width of all other columns. The `\_tsizesum` includes the sum of `\tsize`'s in `\hsize` units after firts pass. The desired table width is stored in the `\_tmpdim`, so `\_tmpdim-\_wd0` is the rest which have to be filled by `\tsize`s. Then the `\tsize` is re-calculated and the real table is printed by `\halign` in the second pass.

147

If no `pxto` keyword was used, then we print the table using `\halign` directly. The `\_tablew` macro is nonempty if the `to` keyword was used.

The ⟨*data*⟩ are re-tokenized by `\_scantextokens` in order to be more robust to catcode changing inside the ⟨*data*⟩. But inline verbatim cannot work in special cases here like `` `{` `` for example.

```
 95 \_long\_def\_tableB #1{\_egroup
 96    {\_def\_tmpb{#1}\_tablereplstrings
 97     \_edef\_tabskipmid{\_the\_tabskip}\_tabskip=\_tabskipl
 98     \_ifpxto
 99       \_edef\_tsizes{\_global\_tsizesum=\_the\_tsizesum \_gdef\_noexpand\_tsizelast{\_tsizelast}}%
100       \_tsizesum=\_zo \_def\_tsizelast{0}%
101       \_tsize=-\_hsize \_setbox0=\_vbox{\_tablepxpreset \_halign \_tableC}%
102       \_advance\_tmpdim by-\_wd0
103       \_ifdim \_tmpdim >\_zo \_else \_tsizesum=\_zo \_fi
104       \_ifdim \_tsizesum >\_zo \_tsize =\_expr{\_number\_hsize/\_number\_tsizesum}\_tmpdim
105       \_else \_tsize=\_zo \_fi
106       \_tsizes % retoring values if there is a \table pxto inside a \table pxto.
107       \_setbox0=\_null \_halign \_tableC
108     \_else
109       \_halign\_tablew \_tableC
110     \_fi
111   }\_egroup % \_tablebox\_bgroup is in the \_table macro
112 }
113 \_def\_tableC{\_ea{\_the\_tabdata\_tabskip=\_tabskipr\_cr \_scantextokens\_ea{\_tmpb\_crcr}}}
```

`\_tabreplstrings` replaces each `\crl` etc. to `\crcr\crl`. The reason is: we want to use macros that scan its parameter to a delimiter written in the right part of the table item declaration. The `\crcr` cannot be hidden in another macro in this case.

```
122 \_def\_tablereplstrings{%
123    \_replstring\_tmpb{\crl}{\_crcr\crl}\_replstring\_tmpb{\crll}{\_crcr\crll}%
124    \_replstring\_tmpb{\crli}{\_crcr\crli}\_replstring\_tmpb{\crlli}{\_crcr\crlli}%
125    \_replstring\_tmpb{\crlp}{\_crcr\crlp}%
126 }
127
128 \_def\_tablepxpreset{} % can be used to de-activate references to .ref file
129 \_newbox\_tstrutbox    % strut used in table rows
130 \_newtoks\_tabdata     % the \halign declaration line
```

The `\_scantabdata` macro converts `\table`'s ⟨*declaration*⟩ to `\halign` ⟨*converted declaration*⟩. The result is stored into `\_tabdata` tokens list. For example, the following result is generated when ⟨*declaration*⟩=`|cr||cl|`.

```
    tabdata: \_vrule\_the\_tabiteml{\_hfil#\_unsskip\_hfil}\_the\_tabitemr\_tabstrutA
       &\_the\_tabiteml{\_hfil#\_unsskip}\_the\_tabitemr
                                        \_vrule\_kern\_vvkern\_vrule\_tabstrutA
       &\_the\_tabiteml{\_hfil#\_unsskip\_hfil}\_the\_tabitemr\_tabstrutA
       &\_the\_tabiteml{\_relax#\_unsskip\_hfil}\_the\_tabitemr\_vrule\_tabstrutA
    ddlinedata: &\_dditem &\_dditem\_vvitem &\_dditem &\_dditem
```

The second result in the `\_ddlinedata` macro is a template of one row of the table used by `\crli` macro.

```
150 \_def\_scantabdata#1{\_let\_next=\_scantabdata
151    \_ifx\_relax#1\_let\_next=\_relax
152    \_else\_ifx|#1\_addtabvrule
153       \_else\_ifx(#1\_def\_next{\_scantabdataE}%
154          \_else\_ifx:#1\_def\_next{\_scantabdataF}%
155             \_else\_isinlist{123456789}#1\_iftrue \_def\_next{\_scantabdataC#1}%
156                \_else \_ea\_ifx\_csname _tabdeclare#1\_endcsname \_relax
157                   \_ea\_ifx\_csname _paramtabdeclare#1\_endcsname \_relax
158                      \_opwarning{tab-declarator "#1" unknown, ignored}%
159                   \_else
160                      \_def\_next{\_ea\_scantabdataB\_csname _paramtabdeclare#1\_endcsname}\_fi
161                   \_else \_def\_next{\_ea\_scantabdataA\_csname _tabdeclare#1\_endcsname}%
162    \_fi\_fi\_fi\_fi\_fi\_fi \_next
163 }
164 \_def\_scantabdataA#1{\_addtabitem
165    \_ea\_addtabdata\_ea{#1\_tabstrutA \_tabskip\_tabskipmid\_relax}\_scantabdata}
```

```
166  \_def\_scantabdataB#1#2{\_addtabitem
167      \_ea\_addtabdata\_ea{#1{#2}\_tabstrutA \_tabskip\_tabskipmid\_relax}\_scantabdata}
168  \_def\_scantabdataC {\_def\_tmpb{}\_afterassignment\_scantabdataD \_tmpnum=}
169  \_def\_scantabdataD#1{\_loop \_ifnum\_tmpnum>0 \_advance\_tmpnum by-1 \_addto\_tmpb{#1}\_repeat
170      \_ea\_scantabdata\_tmpb}
171  \_def\_scantabdataE#1){\_addtabdata{#1}\_scantabdata}
172  \_def\_scantabdataF {\_addtabitem\_def\_addtabitem{\_let\_addtabitem=\_addtabitemx}\_scantabdata}
```

The _addtabitemx adds the boundary code (used between columns) to the ⟨*converted declaration*⟩. This code is \egroup &\bgroup \colnum=⟨*value*⟩\relax. You can get the current number of column from the \colnum register, but you cannot write \the\colnum as the first object in a ⟨*data*⟩ item because \halign first expands the front of the item and the left part of the declaration is processed after this. Use \relax\the\colnum instead. Or you can write:

```
\def\showcolnum{\ea\def\ea\totcolnum\ea{\the\colnum}\the\colnum/\totcolnum}
\table{ccc}{\showcolnum & \showcolnum & \showcolnum}
```

This example prints 1/3 2/3 3/3, because the value of the \colnum is equal to the total number of columns before left part of the column declaration is processed.

```
192  \_newcount\_colnum      % number of current column in the table
193  \_public \colnum ;
194
195  \_def\_addtabitemx{\_ifnum\_colnum>0
196      \_addtabdata{&}\_addto\_ddlinedata{&\_dditem}\_fi
197      \_advance\_colnum by1 \_let\_tmpa=\_relax
198      \_ifnum\_colnum>1 \_ea\_addtabdata\_ea{\_ea\_colnum\_the\_colnum\_relax}\_fi}
199  \_def\_addtabdata#1{\_tabdata\_ea{\_the\_tabdata#1}}
```

This code converts || or | from \table ⟨*declaration*⟩ to the ⟨*converted declaration*⟩.

```
205  \_def\_addtabvrule{%
206      \_ifx\_tmpa\_vrule \_addtabdata{\_kern\_vvkern}%
207          \_ifnum\_colnum=0 \_addto\_vvleft{\_vvitem}\_else\_addto\_ddlinedata{\_vvitem}\_fi
208      \_else \_ifnum\_colnum=0 \_addto\_vvleft{\_vvitemA}\_else\_addto\_ddlinedata{\_vvitemA}\_fi\_fi
209      \_let\_tmpa=\_vrule \_addtabdata{\_vrule}%
210  }
211  \_def\_tabstrutA{\_copy\_tstrutbox}
212  \_def\_vvleft{}
213  \_def\_ddlinedata{}
```

The default "declaration letters" c, l, r and p are declared by setting _tabdeclarec, _tabdeclarel, _tabdeclarer and _paramtabdeclarep macros. In general, define \def_tabdeclare⟨*letter*⟩{...} for a non-parametric letter and \def_paramtabdeclare⟨*letter*⟩{...} for a letter with a parameter. The double hash ## must be in the definition, it is replaced by a real table item data. You can declare more such "declaration letters" if you want.

   Note, that the ## with fills are in group. The reason can be explained by following example:

```
\table{|c|c|}{\crl \Red A & B \crl}
```

We don't want vertical line after red A to be in red.

```
232  \_def\_tabdeclarec{\_the\_tabiteml \_begingroup\_hfil##\_unsskip\_hfil\_endgroup \_the\_tabitemr}
233  \_def\_tabdeclarel{\_the\_tabiteml \_begingroup ##\_unsskip\_hfil\_endgroup \_the\_tabitemr}
234  \_def\_tabdeclarer{\_the\_tabiteml \_begingroup\_hfil##\_unsskip\_endgroup \_the\_tabitemr}
```

The _paramtabdeclarep{⟨*data*⟩} is invoked when p{⟨*data*⟩} declarator is used. First, it saves the \hsize value and then it runs _tablepar. The _tablepar macro behaves like _tableparbox (which is \vtop) in normal cases. But there is a special case: if the first pass of pxto table is processed then \hsize is negative. We print nothing in this case, i.e. _tableparbox is \ignoreit and we advance the _tsizesum. The auxiliary macro _tsizelast is used to do advancing only in the first row of the table. _tsizesum and _tsizelast are initialized in the _tableB macro.

```
249  \_def\_paramtabdeclarep#1{\_hsize=#1\_relax
250      \_the\_tabiteml \_tablepar{\_tableparB ##\_tableparC}\_the\_tabitemr
251  }
252  \_def\_tablepar{%
253      \_ifdim\_hsize<0pt
```

```
254        \_ifnum\_tsizelast<\_colnum \_global\_advance\_tsizesum by-\_hsize
255            \_xdef\_tsizelast{\_the\_colnum}\_fi
256        \_let\_tableparbox=\_ignoreit
257    \_fi
258    \_tableparA \_tableparbox
259 }
260 \_let \_tableparbox=\_vtop
261 \_let \_tableparA=\_empty
262 \_newdimen \_tsizesum
263 \_def \_tsizelast{0}
```

The `\_tableparB` initializes the paragraphs inside the table item and `\_tableparC` closes them. They are used in the `\_paramtabdeclarep` macro. The first paragraph is no indented.

```
271 \_def\_tableparB{%
272    \_baselineskip=\_normalbaselineskip \_lineskiplimit=\_zo \_noindent
273    \_raise\_ht\_tstrutbox\_null \_hskip\_zo \_relax
274 }
275 \_def\_tableparC{%
276    \_unsskip
277    \_ifvmode\_vskip\_dp\_tstrutbox \_else\_lower\_dp\_tstrutbox\_null\_fi
278 }
```

Users put optional spaces around the table item typically, i.e. they write `& text &` instead `&text&`. The left space is ignored by the internal TeX algorithm but the right space must be removed by macros. This is a reason why we recommend to use `\_unsskip` after each `##` in your definition of "declaration letters". This macro isn't only the primitive `\unskip` because we allow usage of plain TeX `\hideskip` macro: `&\hideskip text\hideskip&`.

```
289 \_def\_unsskip{\_ifmmode\_else\_ifdim\_lastskip>\_zo \_unskip\_fi\_fi}
```

The `\fL`, `\fR`, `\fC` and `\fX` macros only do special parameters settings for paragraph building algorithm.

```
296 \_let\_fL=\_raggedright
297 \_def\_fR{\_leftskip=0pt plus 1fill \_relax}
298 \_def\_fC{\_leftskip=0pt plus1fill \_rightskip=0pt plus 1fill \_relax}
299 \_def\_fX{\_leftskip=0pt plus1fil  \_rightskip=0pt plus-1fil \_parfillskip=0pt plus2fil \_relax}
300 \_public \fL \fR \fC \fX ;
```

The `\fS` macro is more tricky. The `\_tableparbox` isn't printed immediatelly, but `\setbox2=` is prefixed by the macro `\_tableparA`, which is empty by default (used in `\_tablepar`). The `\_tableparD` is processed after the box is set: it checks if there is only one line and prints `\hbox to\hsize{\hfil`⟨*this line*⟩`\hfil}` in this case. In other cases, the box2 is printed.

```
311 \_def\_fS{\_relax
312    \_ifdim\_hsize<0pt \_else \_def\_tableparA{\_setbox2=}\_fi
313    \_addto\_tableparC{\_aftergroup\_tableparD}%
314 }
315 \_def\_tableparD{\_setbox0=\_vbox{\_unvcopy2 \_unskip \_global\_setbox1=\_lastbox}%
316    \_ifdim\_ht0>0pt \_box2 \_setbox0=\_box1
317    \_else \_hbox to\_hsize{\_hfil \_unhbox1\_unskip\_unskip\_hfil}\_setbox0=\_box2 \_fi
318 }
319 \_public \fS ;
```

The family of `\_cr*` macros `\crl`, `\crll`, `\crli`, `\crlli`, `\crlp` and `\tskip` ⟨*dimen*⟩ is implemented here. The `\_zerotabrule` is used to suppress the negative `\lineskip` declared by `\tablinespace`.

```
329 \_def\_crl{\_crcr\_noalign{\_hrule}}
330 \_def\_crll{\_crcr\_noalign{\_hrule\_kern\_hhkern\_hrule}}
331 \_def\_zerotabrule {\_noalign{\_hrule height\_zo width\_zo depth\_zo}}
332
333 \_def\_crli{\_crcr \_zerotabrule \_omit
334    \_gdef\_dditem{\_omit\_tablinefil}\_gdef\_vvitem{\_kern\_vvkern\_vrule}\_gdef\_vvitemA{\_vrule}%
335    \_vvleft\_tablinefil\_ddlinedata\_crcr \_zerotabrule}
336 \_def\_crlli{\_crli\_noalign{\_kern\_hhkern}\_crli}
337 \_def\_tablinefil{\_leaders\_hrule\_hfil}
338
339 \_def\_crlp#1{\_crcr \_zerotabrule \_noalign{\_kern-\_drulewidth}%
340    \_omit \_xdef\_crlplist{#1}\_xdef\_crlplist{,\_ea}\_ea\_crlpA\_crlplist,\_end,%
```

```
341    \_global\_tmpnum=0 \_gdef\_dditem{\_omit\_crlpD}%
342    \_gdef\_vvitem{\_kern\_vvkern\_kern\_drulewidth}\_gdef\_vvitemA{\_kern\_drulewidth}%
343    \_vvleft\_crlpD\_ddlinedata \_global\_tmpnum=0 \_crcr \_zerotabrule}
344 \_def\_crlpA#1,{\_ifx\_end#1\_else \_crlpB#1-\_end,\_ea\_crlpA\_fi}
345 \_def\_crlpB#1#2-#3,{\_ifx\_end#3\_xdef\_crlplist{\_crlplist#1#2,}\_else\_crlpC#1#2-#3,\_fi}
346 \_def\_crlpC#1-#2-#3,{\_tmpnum=#1\_relax
347    \_loop \_xdef\_crlplist{\_crlplist\_the\_tmpnum,}\_ifnum\_tmpnum<#2\_advance\_tmpnum by1 \_repeat}
348 \_def\_crlpD{\_incr\_tmpnum \_edef\_tmpa{\_noexpand\_isinlist\_noexpand\_crlplist{,\_the\_tmpnum,}}%
349    \_tmpa\_iftrue \_kern-\_drulewidth \_tablinefil \_kern-\_drulewidth\_else \_hfil \_fi}
350
351 \_def\_tskip{\_afterassignment\_tskipA \_tmpdim}
352 \_def\_tskipA{\_gdef\_dditem{}\_gdef\_vvitem{}\_gdef\_vvitemA{}\_gdef\_tabstrutA{}%
353    \_vbox to\_tmpdim{}\_ddlinedata \_crcr
354    \_zerotabrule \_noalign{\_gdef\_tabstrutA{\_copy\_tstrutbox}}}
355
356 \_public \crl \crll \crli \crlli \crlp \tskip ;
```

The `\mspan{⟨number⟩}[⟨declaration⟩]{⟨text⟩}` macro generates similar `\omit\span\omit\span` sequence as plain TeX macro `\multispan`. Moreover, it uses `\_scantabdata` to convert ⟨declaration⟩ from `\table` syntax to `\halign` syntax.

```
364 \_def\_mspan{\_omit \_afterassignment\_mspanA \_mscount=}
365 \_def\_mspanA[#1]#2{\_loop \_ifnum\_mscount>1 \_cs{_span}\_omit \_advance\_mscount-1 \_repeat
366    \_count1=\_colnum \_colnum=0 \_def\_tmpa{}\_tabdata={}\_scantabdata#1\_relax
367    \_colnum=\_count1 \_setbox0=\_vbox{\_halign\_ea{\_the\_tabdata\_cr#2\_cr}%
368    \_global\_setbox8=\_lastbox}%
369    \_setbox0=\_hbox{\_unhbox8 \_unskip \_global\_setbox8=\_lastbox}%
370    \_unhbox8 \_ignorespaces}
371 \_public \mspan ;
```

The `\vspan⟨number⟩{⟨text⟩}` implementation is here. We need to lower the box by

(⟨number⟩-1)*(\ht+\dp of \tabstrut) / 2.

The `#1` parameter must be a one-digit number. If you want to set more digits then use braces.

```
383 \_def\_vspan#1#2{\_vspanA{#1#2}}
384 \_def\_vspanA#1#2{\_vtop to\_zo{\_hbox{\_lower \_dimexpr
385    #1\_dimexpr(\_ht\_tstrutbox+\_dp\_tstrutbox)/2\_relax
386    -\_dimexpr(\_ht\_tstrutbox+\_dp\_tstrutbox)/2\_relax \_hbox{#2}}\_vss}}
387 \_public \vspan ;
```

The parameters of primitive `\vrule` and `\hrule` keeps the rule "last wins". If we re-define `\hrule` to `\_orihrule height1pt` then each usage of redefined `\hrule` uses `1pt` height if this parameter isn't overwritten by another following `height` parameter. This principle is used for settings another default rule thickness than 0.4 pt by the macro `\rulewidth`.

```
398 \_newdimen\_drulewidth  \_drulewidth=0.4pt
399 \_let\_orihrule=\_hrule  \_let\_orivrule=\_vrule
400 \_def\_rulewidth{\_afterassignment\_rulewidthA \_drulewidth}
401 \_def\_rulewidthA{\_edef\_hrule{\_orihrule height\_drulewidth}%
402                   \_edef\_vrule{\_orivrule width\_drulewidth}%
403                   \_let\_rulewidth=\_drulewidth
404                   \_public \vrule \hrule \rulewidth;}
405 \_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.

```
415 \_long\_def\_frame#1{%
416    \_hbox{\_vrule\_vtop{\_vbox{\_hrule\_kern\_vvkern
417       \_hbox{\_kern\_hhkern\_relax#1\_kern\_hhkern}%
418    }\_kern\_vvkern\_hrule}\_vrule}}
419 \_public \frame ;
```

`\eqbox` and `\eqboxsize` are implemented here. The widths of all `\eqbox`es are saved to the `.ref` file in the format `\_Xeqbox{⟨label⟩}{⟨size⟩}`. The `.ref` file is read again and maximum box width for each ⟨label⟩ is saved to `\_eqb:⟨label⟩`.

```
428  \_def\_Xeqbox#1#2{%
429     \_ifcsname _eqb:#1\_endcsname
430        \_ifdim #2>\_cs{_eqb:#1}\_relax \_sdef{_eqb:#1}{#2}\_fi
431     \_else \_sdef{_eqb:#1}{#2}\_fi
432  }
433  \_def\_eqbox #1[#2]#3{\_setbox0=\_hbox{{#3}}%
434     \_openref \_immediate\_wref \_Xeqbox{{#2}{\_the\_wd0}}%
435     \_ifcsname _eqb:#2\_endcsname
436        \_hbox to\_cs{_eqb:#2}{\_ifx r#1\_hfill\_fi\_hss\_unhbox0\_hss\_ifx l#1\_hfill\_fi}%
437     \_else \_box0 \_fi
438  }
439  \_def\_eqboxsize [#1]#2{\_trycs{_eqb:#1}{#2}}
440
441  \public \eqbox \eqboxsize ;
```

2.31 Balanced multi-columns

```
3  \_codedecl \begmulti {Balanced columns <2022-05-05>} % preloaded in format
```

`\_betweencolumns` or `\_leftofcolumns` or `\_rightofcolumns` include a material printed between columns or left of all columns or right of all columns respectivelly. The `\_betweencolumns` must include a stretchability or a material with exactly `\colsep` width. You can redefine these macros. For example the rule between colums can be reached by `\_def\_betweencolumns{\hss\vrule\hss}`. `\_multiskip` puts its material at the start and at the end of `\begmulti`...`\endmulti`.

```
16  \_def\_betweencolumns{\_hss} \_def\_leftofcolumns{} \_def\_rightofcolumns{}
17  \_def\_multiskip{\_medskip}   % space above and below \begmulti...\endmulti
```

The code used here is documented in detail in the "TEXbook naruby", pages 244–246, free available, http://petr.olsak.net/tbn.html, but in Czech. Roughly speaking, macros complete all material between `\begmulti`⟨*num-columns*⟩ and `\endmulti` into one `\vbox` 6. Then the macro measures the amount of free space at the current page using `\pagegoal` and `\pagtotal` and does `\vsplit` of `\vbox` 6 to columns with a height of such free space. This is done only if we have enough amount of material in `\vbox` 6 to fill the full page by columns. This is repeated in a loop until we have less amount of material in `\vbox` 6. Then we run `\_balancecolumns` which balances the last part of the columns. Each part of printed material is distributed to the main vertical list as `\hbox{`⟨*columns*⟩`}` and we need not do any change in the output routine.

If you have paragraphs in `\begmulti`... `\endmulti` environment then you may say `\raggedright` inside this environment and you can re-assign `\widowpenalty` and `\clubppenalty` (they are set to 10000 in OpTEX).

```
38  \_newcount\_mullines
39
40  \_def\_begmulti #1 {\_par\_bgroup\_wipeepar \_multiskip \_def\_Ncols{#1}
41     \_setbox6=\_vbox\_bgroup\_bgroup \_let\_setxhsize=\_relax \_penalty-99
42     %% \hsize := column width = (\hsize+\colsep) / n - \colsep
43     \_setbox0=\_hbox{\_leftofcolumns\_rightofcolumns}%
44     \_advance\_hsize by-\_wd0  \_advance\_hsize by\_colsep
45     \_divide\_hsize by\_Ncols  \_advance\_hsize by-\_colsep
46     \_mullines=0
47     \_def\_par{\_ifhmode\_endgraf\_global\_advance\_mullines by\_prevgraf\_fi}%
48  }
49  \_def\_endmulti{\_vskip-\_prevdepth\_vfil
50     \_ea\_egroup\_ea\_egroup\_ea\_baselineskip\_the\_baselineskip\_relax
51     \_dimen0=.8\_maxdimen \_tmpnum=\_dimen0 \_divide\_tmpnum by\_baselineskip
52     \_splittopskip=\_baselineskip
53     \_setbox1=\_vsplit6 to0pt % initialize first \splittopskip in \box6
54     %% \dimen1 := the free space on the page
55     \_penalty0 % initialize \_pageoal
56     \_ifdim\_pagegoal=\_maxdimen \_setcolsize\_vsize
57     \_else \_setcolsize{\_dimexpr\_pagegoal-\_pagetotal}\_fi
58     \_ifdim \_dimen1<2\_baselineskip
59       \_vfil\_break \_setcolsize\_vsize \_fi
60     \_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
61     \_divide\_dimen0 by\_Ncols \_relax
```

```
62    %% split the material to more pages?
63    \_ifdim \_dimen0>\_dimen1 \_splitpart
64    \_else \_balancecolumns \_fi  % only balancing
65    \_multiskip \_egroup
66 }
```

Splitting columns...

```
72 \_def\_makecolumns{\_bgroup % full page, destination height: \dimen1
73    \_vbadness=20000 \_dimen6=\_wd6
74    \_createcolumns
75    \_printcolumns
76    \_dimen0=\_dimen1 \_divide\_dimen0 by\_baselineskip \_multiply\_dimen0 by\_Ncols
77    \_global\_advance \_mullines by-\_dimen0
78    \_egroup
79 }
80 \_def\_splitpart{%
81    \_makecolumns % full page
82    \_vskip 0pt plus 1fil minus\_baselineskip \_break
83    \_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
84    \_divide\_dimen0 by\_Ncols \_relax
85    \_ifx\_balancecolumns\_flushcolumns \_advance\_dimen0 by-.5\_vsize \_fi
86    \_setcolsize\_vsize \_dimen2=\_dimen1
87    \_advance\_dimen2 by-\_baselineskip
88    %% split the material to more pages?
89    \_ifvoid6 \_else
90       \_ifdim \_dimen0>\_dimen2 \_ea\_ea\_ea \_splitpart
91       \_else \_balancecolumns % last balancing
92    \_fi \_fi
93 }
```

Final balancing of the columns.

```
99  \_def\_balancecolumns{\_bgroup \_setbox7=\_copy6 % destination height: \dimen0
100    \_ifdim\_dimen0>\_baselineskip \_else \_dimen0=\_baselineskip \_fi
101    \_vbadness=20000 \_dimen6=\_wd6 \_dimen1=\_dimen0
102    \_def\_tmp{\_createcolumns
103       \_ifvoid6 \_else
104          \_advance \_dimen1 by.2\_baselineskip
105          \_setbox6=\_copy7
106          \_ea \_tmp \_fi}\_tmp
107    \_printcolumns
108    \_egroup
109 }
```

_setcolsize⟨*dimen*⟩ sets initial value \dimen1=⟨*size*⟩ which is used as height of columns at given page. The correction \splittopskip−\topskip is done if the columns start at the top of the page.
_createcolumns prepares columns with given height \dimen1 side by side to the \box1.
_printcolumns prints the columns prepared in \box1. The first \hbox{} moves typesetting point to the next baseline. Next negative skip ensures that the first line from splitted columns is at this position.

```
124 \_def\_setcolsize #1{\_dimen1=#1\_relax
125    \_ifdim\_dimen1=\_vsize
126       \_advance \_dimen1 by \_splittopskip \_advance \_dimen1 by-\_topskip \_fi
127 }
128 \_def\_createcolumns{%
129    \_setbox1=\_hbox{\_leftofcolumns}\_tmpnum=0
130    \_loop \_ifnum\_Ncols>\_tmpnum
131       \_advance\_tmpnum by1
132       \_setbox1=\_hbox{\_unhbox1
133          \_ifvoid6 \_hbox to\_dimen6{\_hss}\_else \_vsplit6 to\_dimen1 \_fi
134          \_ifnum\_Ncols=\_tmpnum \_rightofcolumns \_else \_betweencolumns \_fi}%
135    \_repeat
136 }
137 \_def\_printcolumns{%
138    \_hbox{}\_nobreak\_vskip-\_splittopskip \_nointerlineskip
139    \_hbox to\_hsize{\_unhbox1}%
140 }
141 \_public \begmulti \endmulti ;
```

2.32 Citations, bibliography

2.32.1 Macros for citations and bibliography preloaded in the format

```
3  \_codedecl \cite {Cite, Biblioraphy <2021-04-13>} % preloaded in format
```

Registers used by \cite, \bib macros are declared here. The \bibnum counts the bibliography items from one. The \bibmark is used when \nonumcitations is set.

```
11  \_newcount\_bibnum                        % the bibitem counter
12  \_newtoks\_bibmark                        % the bibmark used if \nonumcitations
13  \_newcount\_lastcitenum \_lastcitenum=0   % for \shortcitations
14  \_public \bibnum \bibmark ;
```

_bibp expands to \bibpart/. By default, \bibpart is empty, so internal links are in the form cite:/⟨number⟩. If \bibpart is set to ⟨bibpart⟩, then internal links are cite:⟨bibpart⟩/⟨number⟩.

```
23  \_def\_bibp{\_the\_bibpart/}             % unique name for each bibliography list
```

\cite [⟨label⟩,⟨label⟩,...,⟨label⟩] manages ⟨labes⟩ using _citeA and prints [⟨bib-marks⟩] using _printsavedcites.
\nocite [⟨label⟩,⟨label⟩,...,⟨label⟩] only manages ⟨labels⟩ but prints nothing.
\rcite [⟨label⟩,⟨label⟩,...,⟨label⟩] behaves like \cite but prints ⟨bib-marks⟩ without brackets.
\ecite [⟨label⟩]{⟨text⟩} behaves like \rcite [⟨label⟩] but prints ⟨text⟩ instead ⟨bib-mark⟩. The ⟨text⟩ is hyperlinked like ⟨bib-marks⟩ when \cite or \rcite is used. The empty internal macro _savedcites will include the ⟨bib-marks⟩ list to be printed. This list is set by _citeA inside a group and it is used by _printsavedcites in the same group. Each \cite/\rcite/\ecite macro starts from empty list of ⟨bib-marks⟩ because new group is opened.

```
43  \_def\_cite[#1]{{\_citeA#1,,,[\_printsavedcites]}}
44  \_def\_nocite[#1]{{\_citeA#1,,,}}
45  \_def\_rcite[#1]{{\_citeA#1,,,\_printsavedcites}}
46  \_def\_ecite[#1]{\_bgroup\_citeA#1,,,\_ea\_eciteB\_savedcites;}
47  \_def\_eciteB#1,#2;#3{\_if?#1\_relax #3\_else \_ilink[cite:\_bibp#1]{#3}\_fi\_egroup}
48  \_def\_savedcites{}
49
50  \_public \cite \nocite \rcite \ecite ;
```

⟨bib-marks⟩ may be numbers or a special text related to cited bib-entry. It depends on \nonumcitations and on used bib-style. The mapping from ⟨label⟩ to ⟨bib-mark⟩ is done when \bib or \usebib is processed. These macros store the information to _Xbib{⟨bibpart⟩}{⟨label⟩}{⟨number⟩}{⟨nonumber⟩} where ⟨number⟩ and ⟨nonumber⟩ are two variants of ⟨bib-mark⟩ (numbered or text-like). This information is read from .ref file and it is saved to macros _bib:⟨bibpart⟩/⟨label⟩ and _bim:⟨bibpart⟩/⟨number⟩. First one includes ⟨number⟩ and second one includes ⟨nonumber⟩. The _lastbn:⟨bibpart⟩ macro includes last number of bib-entry used in the document with given ⟨bibpart⟩. A designer can use it to set appropriate indentation when printing the list of all bib-entries.

```
69  \_def\_Xbib#1#2#3#4{\_sxdef{_bib:#1/#2}{\_bibnn{#3}&}%
70     \_if^#4^\_else\_sxdef{_bim:#1/#3}{#4}\_fi\_sxdef{_lastbn:#1}{#3}}
```

_citeA ⟨label⟩, processes one label from the list of labels given in the parameter of \cite, \nocite, \rcite or \ecite macros. It adds the ⟨label⟩ to a global list _ctlst:⟨bibpart⟩/ which will be used by \usebib (it must know what ⟨labels⟩ are used in the document to pick-up only relevant bib-entries from the database. Because we want to save space and to avoid duplications of ⟨label⟩ in the _ctlst:⟨bibpart⟩/, we distinguish four cases:

- ⟨label⟩ was not declared by _Xbib before and it is first such a ⟨label⟩ in the document: Then _bib:⟨bibpart⟩/⟨label⟩ is undefined and we save label using _addcitelist, write warning on the terminal and define _bib:⟨bibpart⟩/⟨label⟩ as empty.
- ⟨label⟩ was not declared by _Xbib before but it was used previously in the document: Then _bib:⟨bibpart⟩/⟨label⟩ is empty and we do nothing (only data to _savedcites are saved).
- ⟨label⟩ was declared by _Xbib before and it is first such ⟨label⟩ used in the document: Then _bib:⟨bibpart⟩/⟨label⟩ includes _bibnn{⟨number⟩}& and we test this case by the command \if &_bibnn{⟨number⟩}&. This is true when _bibnn{⟨number⟩} expands to empty. The ⟨label⟩ is saved by _addcitelist and _bib:⟨bibpart⟩/⟨label⟩ is re-defined directly as ⟨number⟩.
- ⟨label⟩ was declared by _Xbib and it was used previously in the document. Then we do nothing (only data to _savedcites are saved).

The `\_citeA` macro runs repeatedly over the whole list of ⟨*labels*⟩.

```
 99  \_def\_citeA #1#2,{\_if#1,\_else
100      \_if *#1\_addcitelist{*}\_ea\_skiptorelax \_fi
101      \_ifcsname _bib:\_bibp#1#2\_endcsname \_else
102         \_addcitelist{#1#2}%
103         \_opwarning{{\_the\_bibpart} \_noexpand\cite [#1#2] unknown. Try to TeX me again}\_openref
104         \_incr\_unresolvedrefs
105         \_addto\_savedcites{?,}\_def\_sortcitesA{}\_lastcitenum=0
106         \_ea\_gdef \_csname _bib:\_bibp#1#2\_endcsname {}%
107         \_ea\_skiptorelax \_fi
108      \_ea\_ifx \_csname _bib:\_bibp#1#2\_endcsname \_empty
109         \_addto\_savedcites{?,}\_def\_sortcitesA{}\_lastcitenum=0
110         \_ea\_skiptorelax \_fi
111      \_def\_bibnn##1{}%
112      \_if &\_csname _bib:\_bibp#1#2\_endcsname
113         \_def\_bibnn##1##2{##1}%
114         \_addcitelist{#1#2}%
115         \_sxdef{_bib:\_bibp#1#2}{\_csname _bib:\_bibp#1#2\_endcsname}%
116      \_fi
117      \_edef\_savedcites{\_savedcites \_csname _bib:\_bibp#1#2\_endcsname,}%
118      \_relax
119      \_ea\_citeA\_fi
120  }
121  \_let\_bibnn=\_relax
```

Because we implement possibility of more independent bibliography lists distinguished by ⟨*bibpart*⟩, the `\_addcitelist{`⟨*label*⟩`}` macro must add the ⟨*label*⟩ to given `\_ctlst:`⟨*bibpart*⟩`/`.
When `\_addcitelist` is processed before `\usebib`, then `\_citeI[`⟨*label*⟩`]` is added. `\usebib` will use this list for selecting right records from `.bib` file. Then `\usebib` sets `\_ctlst:`⟨*bibpart*⟩`/` to `\_write`.
If `\_addcitelist` is processed after `\usebib`, then `\_Xcite{`⟨*bibpart*⟩`/}{`⟨*label*⟩`}` is saved to the `.ref` file. The `\_Xcite` creates `\_ctlstB:`⟨*bibpart*⟩`/` as a list of saved `\_citeI[`⟨*label*⟩`]`. Finally, `\usebib` concats boths lists `\_ctlst:`⟨*bibpart*⟩`/` and `\_ctlstB:`⟨*bibpart*⟩`/` in the second TeX run.

```
138  \_def\_addcitelist#1{%
139      \_unless \_ifcsname _ctlst:\_bibp\_endcsname \_sxdef{_ctlst:\_bibp}{}\_fi
140      \_ea \_ifx \_csname _ctlst:\_bibp\_endcsname \_write
141         \_openref \_immediate\_wref\_Xcite{{\_bibp}{#1}}%
142      \_else \_global \_ea\_addto \_csname _ctlst:\_bibp\_endcsname {\_citeI[#1]}\_fi
143  }
144  \_def\_Xcite#1#2{%
145      \_unless \_ifcsname _ctlstB:#1\_endcsname \_sxdef{_ctlstB:#1}{}\_fi
146      \_global \_ea\_addto \_csname _ctlstB:#1\_endcsname {\_citeI[#2]}%
147  }
```

The ⟨*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 the 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).

```
163  \_def\_printsavedcites{\_sortcitesA
164      \_chardef\_tmpb=0 \_ea\_citeB\_savedcites,%
165      \_ifnum\_tmpb>0 \_printdashcite{\_the\_tmpb}\_fi
166  }
167  \_def\_sortcitesA{}
168  \_def\_sortcitations{%
169    \_def\_sortcitesA{\_edef\_savedcites{300000,\_ea}\_ea\_sortcitesB\_savedcites,%
170                    \_def\_tmpa####1300000,{\_def\_savedcites{####1}}\_ea\_tmpa\_savedcites}%
171  }
172  \_def\_sortcitesB #1,{\_if $#1$
173    \_else
174       \_mathchardef\_tmpa=#1
175       \_edef\_savedcites{\_ea}\_ea\_sortcitesC \_savedcites\_end
176       \_ea\_sortcitesB
```

155

```
177    \_fi
178 }
179 \_def\_sortcitesC#1,{\_ifnum\_tmpa<#1\_edef\_tmpa{\_the\_tmpa,#1}\_ea\_sortcitesD
180                   \_else\_edef\_savedcites{\_savedcites#1,}\_ea\_sortcitesC\_fi}
181 \_def\_sortcitesD#1\_end{\_edef\_savedcites{\_savedcites\_tmpa,#1}}
182
183 \_def\_citeB#1,{\_if$#1$\_else
184    \_if?#1\_relax??%
185       \_else
186       \_ifnum\_lastcitenum=0    % only comma separated list
187          \_printcite{#1}%
188       \_else
189          \_ifx\_citesep\_empty  % first cite item
190             \_lastcitenum=#1\_relax
191             \_printcite{#1}%
192          \_else              % next cite item
193             \_advance\_lastcitenum by1
194             \_ifnum\_lastcitenum=#1\_relax % cosecutive cite item
195               \_mathchardef\_tmpb=\_lastcitenum
196            \_else  % there is a gap between cite items
197               \_lastcitenum=#1\_relax
198               \_ifnum\_tmpb=0 % previous items were printed
199                  \_printcite{#1}%
200               \_else
201                  \_printdashcite{\_the\_tmpb}\_printcite{#1}\_chardef\_tmpb=0
202    \_fi\_fi\_fi\_fi
203    \_ea\_citeB\_fi
204 }
205 \_def\_shortcitations{\_lastcitenum=1 }
206
207 \_def\_printcite#1{\_citesep
208    \_ilink[cite:\_bibp#1]{\_citelinkA{#1}}\_def\_citesep{,\_hskip.2em\_relax}}
209 \_def\_printdashcite#1{\_ifmmode-\_else\_hbox{--}\_fi\_ilink[cite:\_bibp#1]{\_citelinkA{#1}}}
210 \_def\_citesep{}
211
212 \_def\_nonumcitations{\_lastcitenum=0\_def\_sortcitesA{}\_def\_etalchar##1{$^{##1}$}%
213    \_def\_citelinkA##1{\_trycs{_bim:\_bibp##1}
214       {##1\_opwarning{\_noexpand\nonumcitations + empty bibmark. Maybe bad bib-style}}}%
215 }
216 \_def\_citelinkA{}
217
218 \_public \nonumcitations \sortcitations \shortcitations ;
```

The **\bib** [⟨*label*⟩] or \bib [⟨*label*⟩] ={⟨*bib-mark*⟩} prints one bib-entry without reading any database. The bib-entry follows after this command. This command counts the used \bibs from one by \bibnum counter and saves _Xbib{⟨*bibpart*⟩}{⟨*label*⟩}{⟨*number*⟩}{⟨*nonumber*⟩} into .ref file immediately using _wbib{⟨*label*⟩}{⟨*number*⟩}{⟨*nonumber*⟩}. This is the core of creation of mapping from ⟨*labels*⟩ to ⟨*number*⟩ and ⟨*nonumber*⟩.

_bibA and _bibB implement the scanner of the optional argument with the \bibmark.

_bibgl is \relax by default but \slides do \let_bibgl=_global.

_dbib{⟨*label*⟩} creates destination for hyperlinks.

```
234 \_def\_bib#1{\_def\_tmp{\_isnextchar={\_bibA[#1]}{\_bibmark={}\_bibB[#1]}}%
235    \_ea\_tmp\_romannumeral-`\.} % ignore optional space
236 \_def\_bibA[#1]=#2{\_bibmark={#2}\_bibB[#1]}
237 \_def\_bibB[#1]{\_par \_bibskip
238    \_bibgl\_advance\_bibnum by1
239    \_noindent \_def\_tmpb{#1}\_dbib{#1}\_wbib{#1}{\_the\_bibnum}{\_the\_bibmark}%
240    \_printbib \_ignorespaces
241 }
242 \_def\_dbib#1{\_dest[cite:\_bibp\_the\_bibnum]\_printlabel{#1}}
243 \_def\_wbib#1#2#3{%
244    \_ifx\_wref\_wrefrelax\_else \_immediate\_wref\_Xbib{{\_the\_bibpart}{#1}{#2}{#3}}\_fi
245    \_unless \_ifcsname bib:\_bibp#1\_endcsname \_Xbib{\_the\_bibpart}{#1}{#2}{#3}\_fi
246 }
247 \_let\_bibgl=\_relax
248
249 \_public \bib ;
```

The `\_printbib` prints the bib-entry itself. You can re-define it if you want a different design. The `\_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.

```
266  \_def \_printbib {\_hangindent=\_iindent
267     \_ifx\_citelinkA\_empty \_hskip\_iindent \_llap{[\_the\_bibnum] }\_fi
268  }
269  \_def \_bibskip {\_ifnum\_bibnum>0 \_smallskip \_fi}
```

The `\usebib` command is implemented in `usebib.opm` file which is loaded when the `\usebib` command is used first. The `usebib.opm` file loads the `librarian.tex` for scanning the `.bib` files. See the section 2.32.2, where the file `usebib.opm` is documented.

```
279  \_def\_usebib{\_par \_opinput {usebib.opm} \_usebib}
280  \_def\usebib{\_usebib}
```

`\nobibwarning` [⟨*list of bib-labels*⟩] declares a list of bib labels which are not fully declared in `.bib` file but we want to suppress the warning about it. List of bib labels are comma-separated case sensitive list without spaces.

```
290  \_def\_nobibwarnlist{,}
291  \_def\_nobibwarning[#1]{\_global\_addto\_nobibwarnlist{#1,}}
292  \_public \nobibwarning ;
```

2.32.2 The \usebib command

The file `usebib.opm` implements the command `\usebib/`⟨*sorttype*⟩ `(`⟨*style*⟩`)` ⟨*bibfiles*⟩ where ⟨*sorttype*⟩ is one letter `c` (references ordered by citation order in the text) or `s` (references ordered by key in the style file), ⟨*style*⟩ is the part of the name `bib-`⟨*style*⟩`.opm` of the style file and ⟨*bibfiles*⟩ are one or more `.bib` file names without suffix separated by comma without space. Example:

    \usebib/s (simple) mybase,yourbase

This command reads the ⟨*bibfiles*⟩ directly and creates the list of bibliographic references (only those declared by `\cite[]` or `\nocite[]` in the text). The formatting of such references is defined in the style file.

The principle "first entry wins" is used. Suppose `\usebib/s (simple) local,global`. If an entry with the same label is declared in `local.bib` and in `global.bib` too then the first wins. So, you can set exceptions in your `local.bib` file for your document.

The `bib-`⟨*style*⟩`.opm` declares entry types (like `@BOOK`, `@ARTICLE`) and declares their mandatory and optional fields (like `author`, `title`). When a mandatory field is missing in an entry in the `.bib` file then a warning is printed on the terminal about it. You can suppress such warnings by command `\nobibwarning` [⟨*bib-labels*⟩], where ⟨*bib-labels*⟩ is a comma-separated list of labels (without spaces) where missing mandatory fields will be no warned.

Old `.bib` files may use the obscure notation for accents like `{\"o}`. Recommendation: convert such old files to Unicode encoding. If you are unable to do this then you can set `\bibtexhook={\oldaccents}`.

2.32.3 Notes for bib-style writers

The `.bib` files include records in the format:

    @⟨*entry-type*⟩{⟨*label*⟩,
        ⟨*field-name*⟩ = "⟨*field-data*⟩",
        ⟨*field-name*⟩ = "⟨*field-data*⟩",
        ...etc
    }

see the file `demo/op-biblist.bib` for a real example. The ⟨*entry-types*⟩ and ⟨*field-names*⟩ are case insensitive.

Ancient BibTEX has read such files and has generated files appropriate for reading by LATEX. It has worked with a set of ⟨entry-types⟩, see the www page http://en.wikipedia.org/wiki/BibTeX. The set of entry types listed on this www page is de facto the BibTEX standard. The OpTEX bib style writer must "declare" all such entry types and more non-standard entry types can be declared too if there is a good reason for doing it. The word "declare" used in the previous sentence means that a bib-style writer must define the printing rules for each ⟨entry-type⟩. The printing rules for ⟨entry-type⟩ include: which fields will be printed, in what order, by what format they will be printed on (italic, caps, etc.), which fields are mandatory, which are optional, and which are ignored in .bib records.

The style writer can be inspired by two styles already done: `bib-simple.opm` and `bib-iso690.opm`. The second one is documented in detail in section 2.32.5.

The printing rules for each ⟨entry-type⟩ must be declared by `\_sdef{_print:⟨entry-type⟩}` in `bib-⟨style⟩.opm` file. The ⟨entry-type⟩ has to be lowercase here. OpTEX supports following macros for a more comfortable setting of printing rules:

- `\_bprinta` [⟨field-name⟩] {⟨if defined⟩} {⟨if not defined⟩}. The part ⟨if defined⟩ is executed if ⟨field-name⟩ is declared in .bib file for the entry which is currently processed. Else the part ⟨if not defined⟩ is processed. The part ⟨if defined⟩ can include the * parameter which is replaced by the value of the ⟨field-name⟩.
- The part ⟨if not defined⟩ can include the `\_bibwarning` command if the ⟨field-name⟩ is mandatory.
- `\_bprintb` [⟨field-name⟩] {⟨if defined⟩} {⟨if not defined⟩}. The same as `\_bprinta`, but the ##1 parameter is used instead *. Differences: ##1 parameter can be used more than once and can be enclosed in nested braces. The * parameter can be used at most once and cannot be enclosed in braces. Warning: if the `\_bprintb` commands are nested (`\_bprintb` in `\_bprintb`), then you need to write the ####1 parameter for internal `\_bprintb`. But if `\_bprinta` commands are nested then the parameter is not duplicated.
- `\_bprintc` \macro {⟨if non-empty⟩}. The ⟨if non-empty⟩ part is executed if \macro is non-empty. The * parameter can be used, it is replaced by the \macro.
- `\_bprintv` [⟨field1⟩,⟨field2⟩,...] {⟨if defined⟩} {⟨if not defined⟩}. The part ⟨if defined⟩ is executed if ⟨field1⟩ or ⟨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 parameters.

There are two special field-names: `!author` and `!editor`. The processed list of authors or editors are printed here instead of raw data, see the commands `\_authorname` and `\_editorname` below.

The bib-style writer can define `_print:BEGIN` and/or `_print:END`. They are executed at the beginning or end of each ⟨entry-type⟩. The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by `\_printbib` macro used in OpTEX (and may be redefined by the author or document designer).

The `\bibmark`={something} can be declared, for instance in the `_print:END` macro. Such "bibmark" is saved to the .ref file and used in next TEX run as \cite marks when \nonumcitations is set.

Moreover, the bib-style writer must declare the format of special fields `author` and `editor`. These fields include a list of names, each name is precessed individually in a loop. The `\_authorname` or `\_editorname` is called for each name on the list. The bib-style writer must define the `\_authorname` and `\_editorname` commands in order to declare the format of printing each individual name. The following control sequences can be used in these macros:

- `\_NameCount`: the number of the currently processed author in the list
- `\_namecont`: the total number of the authors in the list
- `\_Lastname`, `\_Firstname`, `\_Von`, `\_Junior`: the parts of the name.

The whole style file is read in the group during the \usebib command is executed before typesetting the reference list. Each definition or setting is local here.

The auto-generated phrases (dependent on current language) can be used in bib-style files by `\_mtext{bib.⟨identifier⟩}`, where ⟨ident⟩ is an identifier of the phrase and the phrase itself is defined by `\_sdef{_mt:bib.⟨identifier⟩:⟨language⟩}{⟨phrase⟩}`. See section 2.37.2 for more detail. Phrases for ⟨identifiers⟩: and, etal, edition, citedate, volume, number, prepages, postpages, editor, editors, available, availablealso, bachthesis, masthesis, phdthesis are defined already, see the end of section 2.37.2.

If you are using non-standard field-names in .bib database and bib-style, you have to declare them by `\_CreateField` {⟨fieldname⟩}.

You can declare `\_SortingOrder` in the manner documented by librarian package.

User or author of the bib-style can create the hidden field which has a precedence while sorting names. Example:

```
\CreateField {sortedby}
\SpecialSort {sortedby}
```

Suppose that the .bib file includes:

```
...
author   = "Jan Chadima",
sortedby = "Hzzadima Jan",
...
```

Now, this author is sorted between H and I, because the Ch digraph in this name has to be sorted by this rule.

If you need (for example) to place the auto-citations before other citations, then you can mark your entries in `.bib` file by `sortedby = "@"`, because this character is sorted before `A`.

2.32.4 The `usebib.opm` macro file loaded when `\usebib` is used

```
  3  \_codedecl \MakeReference {Reading bib databases <2022-02-04>} % loaded on demand by \usebib
```

Loading the `librarian.tex` macro package. See `texdoc librarian` for more information about it.

We want to ignore `\errmessage` and we want not to create `\jobname.lbr` file.

```
 13  \_def\errmessage#1{}
 14  \_def\newwrite#1{\_csname lb@restoreat\_endcsname \_endinput}
 15  \_def\_tmpb{\_catcode`\_=12 \_input librarian \_catcode`\_=11 }\_tmpb
 16  \_let\errmessage=\_errmessage
 17  \_let\newwrite=\_newwrite
 18
 19  \_private \BibFile \ReadList \SortList \SortingOrder \NameCount \AbbreviateFirstname
 20     \CreateField \RetrieveFieldInFor \RetrieveFieldIn \RetrieveField ;
```

The `\usebib` command.

```
 26  \_def\_usebib/#1 (#2) #3 {%
 27    \_let\_citeI=\_relax \_xdef\_citelist{\_trycs{_ctlst:\_bibp}{}\_trycs{_ctlstB:\_bibp}{}}%
 28    \_global \_ea\_let \_csname _ctlst:\_bibp\_endcsname =\_write
 29    \_ifx\_citelist\_empty
 30      \_opwarning{No cited items. \_noexpand\usebib ignored}%
 31    \_else
 32      \_bgroup \_par
 33        \_emergencystretch=.3\_hsize
 34        \_def\_optexbibstyle{#2}%
 35        \_setctable\_optexcatcodes
 36        \_input bib-#2.opm
 37        \_the \_bibtexhook
 38        \_ifcsname _mt:bib.and:\_cs{_lan:\_the\_language}\_endcsname \_else
 39          \_opwarning{\_string\usebib: No phrases for language
 40                    "\_cs{_lan:\_the\_language}" (using "en")}%
 41          \_language=0 \_chardef\_documentlanguage=0
 42        \_fi
 43        \_def\_tmp##1[*]##2\_relax{\_def\_tmp{##2}}\_ea\_tmp\_citelist[*]\_relax
 44        \_ifx\_tmp\_empty\_else % there was \nocite[*] used.
 45          \_setbox0=\_vbox{\_hsize=\_maxdimen \_def\_citelist{}\_adef@{\_readbibentry}%
 46          \_input #3.bib
 47          \_ea}\_ea\_def\_ea\_citelist\_ea{\_citelist}%
 48        \_fi
 49        \_def\_citeI[##1]{\_csname lb@cite\_endcsname{##1}{\_bibp}{}{}}\_citelist
 50        \_BibFile{#3}%
 51        \_if s#1\_SortList{\_bibp}\_fi
 52        \_ReadList{\_bibp}%
 53        \_restorectable
 54      \_egroup
 55    \_fi
 56  }
 57  \_def\_readbibentry#1#{\_readbibentryA}
 58  \_def\_readbibentryA#1{\_readbibentryB#1,,\_relax!.}
 59  \_def\_readbibentryB#1#2,#3\_relax!.{\_addto\_citelist{\_citeI[#1#2]}}
```

159

Corrections in librarian macros.

```
65  \_tmpnum=\_catcode`\@  \_catcode`\@=11
66  \_def\lb@checkmissingentries#1,{% we needn't \errmessage here, only \opmacwarning
67    \_def\lb@temp{#1}%
68    \_unless\_ifx\lb@temp\lb@eoe
69      \lb@ifcs{#1}{fields}%
70            {}%
71            {\_opwarning{\_string\usebib: entry [#1] isn't found in .bib}}%
72      \_ea\lb@checkmissingentries
73    \_fi
74  }
75  \_def\lb@readentry#1#2#3,{% space before key have to be ingnored
76    \_def\lb@temp{#2#3}%      we need case sensitive keys
77    \_def\lb@next{\_ea\lb@gotoat\lb@gobbletoeoe}%
78    \lb@ifcs\lb@temp{requested}%
79            {\_let\lb@entrykey\lb@temp
80             \lb@ifcs\lb@entrykey{fields}{}%
81                   {\lb@defcs\lb@entrykey{fields}{}%
82                    \_lowercase{\lb@addfield{entrytype}{#1}}%
83                    \_let\lb@next\lb@analyzeentry}}{}%
84    \lb@next
85  }
86  \_let\lb@compareA=\lb@compare
87  \_let\lb@preparesortA=\lb@preparesort
88  \_def\lb@compare#1\lb@eoe#2\lb@eoe{% SpecialSort:
89    \_ifx\lb@sorttype\lb@namestring
90      \_ifx\_sortfield\_undefined \lb@compareA#1\lb@eoe#2\lb@eoe
91      \_else
92        \_ea\_RetrieveFieldInFor\_ea{\_sortfield}\lb@entrykey\lb@temp
93        \_ifx\lb@temp\_empty \_toks1={#1\lb@eoe}\_else \_toks1=\_ea{\lb@temp\lb@eoe}\_fi
94        \_ea\_RetrieveFieldInFor\_ea{\_sortfield}\lb@currententry\lb@temp
95        \_ifx\lb@temp\_empty \_toks2={#2\lb@eoe}\_else \_toks2=\_ea{\lb@temp\lb@eoe}\_fi
96        \_edef\lb@temp{\_noexpand\lb@compareA\_space\_the\_toks1 \_space\_the\_toks2}\lb@temp
97      \_fi
98    \_else \lb@compareA#1\lb@eoe#2\lb@eoe \_fi
99  }
100 \_def\lb@preparesort#1#2\lb@eoe{%
101   \_if#1-%
102     \_def\lb@sorttype{#2}%
103   \_else
104     \_def\lb@sorttype{#1#2}%
105   \_fi
106   \lb@preparesortA#1#2\lb@eoe
107 }
108 \_def\_SpecialSort#1{\_def\_sortfield{#1}}
109 \_def\WriteImmediateInfo#1{}  % the existence of .lbr file bocks new reading of .bib
110 \_catcode`\@=\_tmpnum
```

Main action per each entry.

```
116 \_def\MakeReference{\_par \_bibskip
117   \_bibgl\_advance\_bibnum by1
118   \_isdefined{_bim:\_bibp\_the\_bibnum}\_iftrue
119     \_edef\_tmpb{\_csname _bim:\_bibp\_the\_bibnum\_endcsname}%
120     \_bibmark=\_ea{\_tmpb}%
121   \_else \_bibmark={}\_fi
122   \_edef\_tmpb{\EntryKey}%
123   \_noindent \_dbib\EntryKey
124   \_printbib
125   {%
126     \_RetrieveFieldIn{entrytype}\_entrytype
127     \_csname _print:BEGIN\_endcsname
128     \_isdefined{_print:\_entrytype}\_iftrue
129       \_csname _print:\_entrytype\_endcsname
130     \_else
131       \_ifx\_entrytype\_empty \_else
132         \_opwarning{Entrytype @\_entrytype\_space from [\EntryKey] undefined}%
133         \_csname _print:misc\_endcsname
134     \_fi\_fi
```

```
135      \_csname _print:END\_endcsname
136      \_wbib \EntryKey {\_the\_bibnum}{\_the\_bibmark}%
137    }\_par
138 }
```

The _bprinta, _bprintb, _bprintc, _bprintv commands used in the style files:

```
145 \_def\_bprinta {\_bprintb*}
146 \_def\_bprintb #1[#2#3]{%
147    \_def\_bibfieldname{#2#3}%
148    \_if!#2\_relax
149        \_def\_bibfieldname{#3}%
150        \_RetrieveFieldIn{#3}\_bibfield
151        \_ifx\_bibfield\_empty\_else
152            \_RetrieveFieldIn{#3number}\_namecount
153            \_def\_bibfield{\_csname _Read#3\_ea\_endcsname \_csname _pp:#3\_endcsname}%
154        \_fi
155    \_else
156        \_RetrieveFieldIn{#2#3}\_bibfield
157    \_fi
158    \_if^#1^%
159        \_ifx\_bibfield\_empty \_ea\_ea\_ea \_doemptyfield
160        \_else \_ea\_ea\_ea \_dofullfield \_fi
161    \_else \_ea \_bprintaA
162    \_fi
163 }
164 \_def\_dofullfield#1#2{\_def\_dofield##1{#1}\_ea\_dofield\_ea{\_bibfield}}
165 \_def\_doemptyfield#1#2{\_def\_dofield##1{#2}\_ea\_dofield\_ea{\_bibfield}}
166 \_let\_Readauthor=\ReadAuthor  \_let\_Readeditor=\ReadEditor
167 \_def\_bprintaA #1#2{\_ifx\_bibfield\_empty #2\_else\_bprintaB #1**\_eee\_fi}
168 \_def\_bprintaB #1*#2*#3\_eee{\_if^#3^#1\_else\_ea\_bprintaC\_ea{\_bibfield}{#1}{#2}\_fi}
169 \_def\_bprintaC #1#2#3{#2#1#3}
170 \_def\_bprintc#1#2{\_bprintcA#1#2**\_relax}
171 \_def\_bprintcA#1#2*#3*#4\_relax{\_ifx#1\_empty \_else \_if^#4^#2\_else#2#1#3\_fi\_fi}
172 \_def\_bprintv [#1]#2#3{\_def\_tmpa{#2}\_def\_tmpb{#3}\_bprintvA #1,,}
173 \_def\_bprintvA #1,{%
174    \_if^#1^\_tmpb\_else
175        \_RetrieveFieldIn{#1}\_tmp
176        \_ifx \_tmp\_empty
177        \_else \_tmpa \_def\_tmpb{}\_def\_tmpa{}%
178        \_fi
179    \_ea \_bprintvA
180    \_fi
181 }
182 \_sdef{_pp:author}{\_letNames\_authorname}
183 \_sdef{_pp:editor}{\_letNames\_editorname}
184 \_def\_letNames{\_let\_Firstname=\Firstname \_let\_Lastname=\Lastname
185    \_let\_Von=\Von \_let\_Junior=\Junior
186 }
```

Various macros + multilingual. Note that _nobibwarnlist is used in _bibwarning and it is set by \nobibwarning macro.

```
193 \_def\_bibwarning{%
194    \_ea\_isinlist \_ea\_nobibwarnlist\_ea{\_ea,\EntryKey,}\_iffalse
195        \_opwarning{Missing field "\_bibfieldname" in [\EntryKey]}\_fi}
```

2.32.5 Usage of the `bib-iso690` style

This is the iso690 bibliographic style used by OpTeX.

See `op-biblist.bib` for an example of the `.bib` input. You can try it by:

```
\fontfam[LMfonts]
\nocite[*]
\usebib/s (iso690) op-biblist
\end
```

161

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 badly interpreted.

The entry names and field names are case insensitive. If there exists 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 bibT_EX (author, title, editor, edition, etc., see http://en.wikipedia.org/wiki/BibTeX). The `iso690` style introduces several "non-standard" fields: ednote, numbering, isbn, issn, doi, url, citedate, key, bibmark. They are documented here.

Moreover, there are two optional special fields:

- lang = language of the entry. The hyphenation plus autogenerated phrases and abbreviations will be typeset by this language.
- option = options by which you can control a special printing of various fields.

There can be only one option field per each entry with (maybe) more options separated by spaces. You can declare the global option(s) in your document applied for each entry by `\biboptions={...}`.

The author field

All names in the author list have to be separated by " `and` ". Each author can be written in various formats (the `von` part is typically missing):

```
Firstname(s) von Lastname
or
von Lastname, Firstname(s)
or
von Lastname, After, Firstname(s)
```

Only the Lastname part is mandatory. Examples:

```
Petr Olšák
or
Olšák, Petr

Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
or
da Vinci, painter, Leonardo Piero
```

The separator " `and` " between authors will be converted to comma during printing, but between the semifinal and final author the word "and" (or something different depending on the current language) is printed.

The first author is printed in reverse order: "LASTNAME, Firstname(s) von, After" and the other authors are printed in normal order: "Firstname(s) von LASTNAME, After". This feature follows the ISO 690 norm. The Lastname is capitalized using uppercase letters. But if the `\caps` font modifier is defined, then it is used and printed `{\caps\_rm␣Lastname}`.

You can specify the option `aumax:`⟨*number*⟩. The ⟨*number*⟩ denotes the maximum authors to be printed. The rest of the authors are ignored and the `et~al.` is appended to the list of printed authors. This text is printed only if the `aumax` value is less than the real number of authors. If you have the same number of authors in the .bib file as you need to print but you want to append `et~al.` then you can use `auetal` option.

There is an `aumin:`⟨*number*⟩ option which denotes the definitive number of printed authors if the author list is not fully printed due to `aumax`. If `aumin` is unused then `aumax` authors are printed in this case.

All authors are printed if `aumax:`⟨*number*⟩ option isn't given. There is no internal limit. But you can set the global options in your document by setting the `\biboptions` tokens list. For example:

```
\biboptions={aumax:7 aumin:1}
% if there are 8 or more authors then only the first author is printed.
```

Examples:

```
    author = "John Green and Bob Brown and Alice Black",
```

output: GREEN, John, Bob BROWN, and Alice BLACK.

```
    author = "John Green and Bob Brown and Alice Black",
    option = "aumax:1",
```

output: GREEN, John et al.

```
    author = "John Green and Bob Brown and Alice Black",
    option = "aumax:2",
```

output: GREEN, John, Bob BROWN et al.

```
    author = "John Green and Bob Brown and Alice Black",
    option = "aumax:3",
```

output: GREEN, John, Bob BROWN, and Alice BLACK.

```
    author = "John Green and Bob Brown and Alice Black",
    option = "auetal",
```

output: GREEN, John, Bob BROWN, Alice BLACK et al.

If you need to add a text before or after the author's list, you can use the `auprint:{`⟨*value*⟩`}` option. The ⟨*value*⟩ will be printed instead of the authors list. The ⟨*value*⟩ can include `\AU` macro which expands to the authors list. Example:

```
    author = "Robert Calbraith",
    option = "auprint:{\AU\space [pseudonym of J. K. Rowling]}",
```

output: CALBRAITH Robert [pseudonym of J. K. Rowling].

You can use the `autrim:`⟨*number*⟩ option. All Firstnames of all authors are trimmed (i. e. reduced to initials) iff the number of authors in the author field is greater than or equal to ⟨*number*⟩. There is an exception: `autrim:0` means that no Firstnames are trimmed. This is the default behavior. Another example: `autrim:1` means that all Firstnames are trimmed.

```
    author = "John Green and Bob Brown and Alice Black",
    option = "auetal autrim:1",
```

output: GREEN, J., B. BROWN, A. BLACK et al.

If you need to write a team name or institution instead of authors, replace all spaces by `\␣` in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after the comma. Example:

```
      author = "Czech\ Technical\ University\ in\ Prague,
                Faculty\ of\ Electrical\ Engeneering",
```

output: CZECH TECHNICAL UNIVERSITY IN PRAGUE, Faculty of Electrical Engeneering.

The editor field
The editor field is used for the list of the authors of the collection. The analogous rules as in author field are used here. It means that the authors are separated by " `and` ", the Firstnames, Lastnames, etc. are interpreted and you can use the options `edmax:`⟨*number*⟩, `edmin:`⟨*number*⟩, `edetal`, `edtrim:`⟨*number*⟩ and `edprint:{`⟨*value*⟩`}` (with `\ED` macro). Example:

```
    editor = "Jan Tomek and Petr Karas",
    option = "edprint:{\ED, editors.} edtrim:1",
```

Output: J. TOMEK and P. KARAS, editors.

If `edprint` option is not set then `{\ED,␣eds.}` or `{\ED,␣ed.}` is used depending on the entry language and on the singular or plural of the editor(s).

The ednote field
The ednote field is used as the secondary authors and more 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 has to be used for Lastnames in the ednote field.

The title field

This is the title of the work. It will be printed (in common entry types) by italics. The ISO 690 norm declares, that the title plus optional subtitle are in italics and they are separated by a colon. Next, the optional secondary title has to be printed in an upright font. This can be added by `titlepost:{⟨value⟩}`. Example:

```
title = "The Simple Title of The Work",
or
title = "Main Title: Subtitle",
or
title  = "Main Title: Subtitle",
option = "titlepost:{Secondary title}",
```

The output of the last example: *Main Title: Subtitle.* Secondary title.

The edition field

This field is used only for second or more edition of cited work. Write only the number without the word "edition". The shortcut "ed." (or something else depending on the current language) is added automatically. Examples:

```
edition = "Second",
edition = "2nd",
edition = "2$^{\rm nd}$",
edition = "2.",
```

Output of the last example: 2. ed.

```
edition = "2."
lang    = "cs",
```

Output: 2. vyd.

Note, that the example `edition␣=␣"Second"` may cause problems. If you are using language "cs" then the output is bad: Second vyd. But you can use `editionprint:{⟨value⟩}` option. The the ⟨value⟩ is printed instead of edition field and shortcut. The edition field must be set. Example:

```
edition = "whatever",
option  = "editionprint:{Second full revised edition}",
```

Output: Second full revised edition.

You can use `\EDN` macro in `editionprint` value. This macro is expanded to the edition value. Example:

```
edition = "Second",
option  = "editionprint:{\EDN\space full revised edition}",
or
edition = "Second full revised edition",
option  = "editionprint:{\EDN}",
```

The address, publisher, year fields

This is an anachronism from ancient BibTEX (unfortunately no exclusive) that the address field includes only the city of the publisher's residence. No more data are here. The publisher field includes the name of the publisher.

```
address = "Berlin",
publisher = "Springer Verlag",
year = 2012,
```

Output: Berlin: Springer Verlag, 2012.

Note, that the year needn't to be inserted into quotes because it is pure numeric.

The letter a, b, etc. are appended to the year automatically if two or more subsequent entries in the bibliography list are not distinct by the first author and year fields. If you needn't this feature, you can use the `noautoletters` option.

You can use "yearprint:⟨*value*⟩" option. If it is set then the ⟨*value*⟩ is used for printing year instead the real field value. The reason: year is sort sensitive, maybe you need to print something else than only sorting key. Example:

```
year   = 2000,
option = "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 a different current language is used) before URL. It means, that the output of the previous example is:

Available from http://petr.olsak.net/opmac.html.

If the `cs` language is the current one than the output is:

Dostupné z: http://petr.olsak.net/opmac.html.

If the `urlalso` option is used, then the added text has the form "Available also from" or "Dostupné také z:" (if `cs` language is current).

The citedate field
This is the citation date. The field must be in the form year/month/day. It means, that the two slashes must be written here. The output depends on the current language. Example:

```
citedate = "2004/05/21",
```

Output when `en` is current: [cit. 2004-05-21].
Output when `cs` is current: [vid. 21. 5. 2004].

The howpublished field
This declares the available medium for the cited document if it is not in printed form. Alternatives: online, CD, DVD, etc. Example:

```
howpublished = "online",
```

Output: [online].

The volume, number, pages and numbering fields
The volume is the "big mark" of the journal issue and the number is the "small mark" of the journal issue and pages includes the page range of the cited article in the journal. The volume is prefixed by Vol. , the number by No. , and the pages by pp. . But these prefixes depends on the language of the entry.
Example:

```
volume = 31,
number = 3,
pages  = "37--42",
```

Output: Vol. 31, No. 3, pp. 37–42.

```
volume = 31,
number = 3,
pages  = "37--42",
lang   = "cs",
```

Output: ročník 31, č. 3, s. 37–42.

If you disagree with the default prefixes, you can use the numbering field. When it is set then it is used instead of volume, number, pages fields and instead of any mentioned prefixes. The numbering can include macros `\VOL`, `\NO`, `\PP`, which are expanded to the respective values of fields. Example:

```
    volume   = 31,
    number   = 3,
    pages    = "37--42"
    numbering = "Issue~\VOL/\NO, pages~\PP",
```

Output: Issue 31/3, pages 37–42

Note: The volume, numbers, and pages fields are printed without numbering filed only in the `@ARTICLE` entry. It means, that if you need to visible them in the `@INBOOK`, `@INPROCEEDINGS` etc. entries, then you must use the numbering field.

Common notes about entries
The order of the fields in the entry is irrelevant. We use the printed order in this manual. The exclamation mark (!) denotes the mandatory field. If the field is missing then a warning occurs during processing.

If the `unpublished` option is set then the fields address, publisher, year, isbn, and pages are not mandatory. If the `nowarn` option is set then no warnings about missing mandatory fields occur.

If the field is used but not mentioned in the entry documentation below then it is silently ignored.

• The `@BOOK` entry
  This is used for book-like entries.
  Fields: author(!), title(!), howpublished, edition, ednote, address(!), publisher(!), year(!), citedate, series, isbn(!), doi, url, note.
  The ednote field here means the secondary authors (illustrator, cover design etc.).

• The `@ARTICLE` entry
  This is used for articles published in a journal.
  Fields: author(!), title(!), journal(!), howpublished, address, publisher, month, year, [numbering or volume, number, pages(!)], citedate, issn, doi, url, note.
  If the numbering is used then it is used instead volume, number, pages.

• The `@INBOOK` entry
  This is used for the part of a book.
  Fields: author(!), title(!), booktitle(!), howpublished, edition, ednote, address(!), publisher(!), year(!), numbering, citedate, series, isbn or issn, doi, url, note.
  The author field is used for author(s) of the part, the editor field includes author(s) or editor(s) of the whole document. The pages field specifies the page range of the part. The series field can include more information about the part (chapter numbers etc.).
  The `@INPROCEEDINGS` and `@CONFERENCE` entries are equivalent to `@INBOOK` entry.

• The `@THESIS` entry
  This is used for the student's thesis.
  Fields: author(!), title(!), howpublished, address(!), school(!), month, year(!), citedate, type(!), ednote, doi, url, note.
  The type field must include the text "Master's Thesis" or something similar (depending on the language of the outer document).
  There are nearly equivalent entries: `@BACHELORSTHESIS`, `@MASTERSTHESIS` and `@PHDTHESIS`. These entries set the type field to an appropriate value automatically. The type field is optional in this case. If it is used then it has precedence before the default setting.

• The @MISC entry
  It is intended for various usage.
  Fields: author, title, howpublished, ednote, citedate, doi, url, note.
  You can use `\AU`, `\ED`, `\EDN`, `\VOL`, `\NO`, `\PP`, `\ADDR`, `\PUBL`, `\YEAR` macros in ednote field. These macros print authors list, editors list, edition, volume, number, pages, address, publisher, and year field values respectively.
  The reason for this entry is to give to you the possibility to set the format of entry by your own decision. The most of data are concentrated in the ednote field.

• The `@BOOKLET`, `@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-oriented bib-marks instead of numbers. This style file auto-generates these marks in the form "Lastname of the first author, comma, space, the year" if the bibmark field isn't declared. If you need to set an exception from this common format, then you can use bibmark field.

The OPmac trick `http://petr.olsak.net/opmac-tricks-e.html#bibmark` describes how to redefine the algorithm for bibmark auto-generating when you need the short form of the type [Au13].

Sorting

If `\usebib/c` is used then entries are sorted by citation order in the text. If `\usebib/s` is used then entries are sorted by "Lastname, Firstname(s)" of the first author and if more entries have this value equal, then the year is used (from older to newer). This feature follows the recommendation of the ISO 690 norm.

If you have the same authors and the same year, you can control the sorting by setting years like 2013, 2013a, 2013b, etc. You can print something different to the list using `yearprint{⟨value⟩}` option, see the section about address, publisher, and year above. The real value of year field (i.e. not yearprint value) is also used in the text-oriented bib-marks when `\nonumcitations` is set.

If you have some problems with name sorting, you can use the hidden field `key`, which is used for sorting instead of the "Lastname Firstname(s)" of authors. If the `key` field is unset then the "Lastname Firstname(s)" is used for sorting normally. Example:

```
author     = "Světla Čmejrková",
key        = "Czzmejrkova Svetla",
```

This entry is now sorted between C and D.

The norm recommends placing the auto-citations at the top of the list of references. You can do this by setting `key␣=␣"@"`, to each entry with your name because the `@` character is sorted before `A`.

Languages

There is the language of the outer document and the languages of each entry. The ISO 690 norm recommends that the technical notes (the prefix before URL, the media type, the "and" conjunction between the semifinal and final author) maybe printed in the language of the outer document. The data of the entry have to be printed in the entry language (edition ed./vyd., Vol./ročník, No./č. etc.). Finally, there are the phrases independent of the language (for example In:). Unfortunately, the bibTEX supposes that the entry data are not fully included in the fields so the automaton has to add some text during processing ("ed.", "Vol.", "see also", etc.). But what language has to be chosen?

The current value of the `\language` register at the start of the `.bib` processing is described as the language of the outer document. This language is used for technical notes regardless of the entry language. Moreover, each entry can have the `lang` field (short name of the language). This language is used for ed./vyd., vol./ročník, etc. and it is used for hyphenation too. If the `lang` is not set then the outer document language is used.

You can use `\_Mtext{bib.⟨identifier⟩}` if you want to use a phrase dependent on outer document language (no on entry language). Example:

```
howpublished = "\_Mtext{bib.blue-ray}"
```

Now, you can set the variants of `bib.blue-ray` phrase for various languages:

```
\_sdef{_mt:bib.blue-ray:en} {Blue-ray disc}
\_sdef{_mt:bib.blue-ray:cs} {Blue-ray disk}
```

Summary of non-standard fields

This style uses the following fields unknown by bibTEX:

```
option     ... options separated by spaces
lang       ... the language two-letter code of one entry
ednote     ... edition info (secondary authors etc.) or
               global data in @MISC-like entries
citedate   ... the date of the citation in year/month/day format
numbering  ... format for volume, number, pages
isbn       ... ISBN
issn       ... ISSN
```

```
doi       ... DOI
url       ... URL
```

Summary of options

```
aumax:⟨number⟩         ... maximum number of printed authors
aumin:⟨number⟩         ... number of printed authors if aumax exceeds
autrim:⟨number⟩        ... full Firstnames iff number of authors are less than this
auprint:{⟨value⟩}      ... text instead authors list (\AU macro may be used)
edmax, edmin, edtrim   ... similar as above for editors list
edprint:{⟨value⟩}      ... text instead editors list (\ED macro may be used)
titlepost:{⟨value⟩}    ... text after title
yearprint:{⟨value⟩}    ... text instead real year (\YEAR macro may be used)
editionprint:{⟨value⟩} .. text instead of real edition (\EDN macro may be used)
urlalso       ... the ``available also from'' is used instead ``available from''
unpublished   ... the publisher etc. fields are not mandatory
nowarn        ... no mandatory fields
```

Other options in the option field are silently ignored.

2.32.6   Implementation of the `bib-iso690` style

```
3  \_codedecl \_undefined {BIB style (iso690) <2022-05-10>} % loaded on demand by \usebib
4
5  \_ifx\_optexbibstyle\_undefined \_errmessage
6    {This file can be read by: \_string\usebib/? (iso690) bibfiles command only}
7    \_endinput \_fi
```

`\_maybedod` (alias `\:` in the style file group) does not put the second dot.

```
13  \_def\_maybedot{\_ifnum\_spacefactor=\_sfcode`\.\_relax\_else.\_fi}
14  \_tmpnum=\_sfcode`\. \_advance\_tmpnum by-2 \_sfcode`\.=\_tmpnum
15  \_sfcode`\?=\_tmpnum \_sfcode`\!=\_tmpnum
16  \_let\:=\_maybedot  % prevents from double periods
17  \_ifx\.\_undefined \_let\.=\_maybedot \_fi % for backward compatibility
```

Option field.

```
23  \_CreateField {option}
24  \_def\_isbiboption#1#2{\_edef\_tmp{\_noexpand\_isbiboptionA{#1}}\_tmp}
25  \_def\_isbiboptionA#1{\_def\_tmp##1 #1 ##2\_relax{%
26      \_if^##2^\_csname iffalse\_ea\_endcsname \_else\_csname iftrue\_ea\_endcsname \_fi}%
27    \_ea\_tmp\_biboptionsi #1 \_relax}
28  \_def\_bibopt[#1]#2#3{\_isbiboption{#1}\_iftrue\_def\_tmp{#2}\_else\_def\_tmp{#3}\_fi\_tmp}
29  \_def\_biboptionvalue#1#2{\_def\_tmp##1 #1:##2 ##3\_relax{\_def#2{##2}}%
30    \_ea\_tmp\_biboptionsi #1: \_relax}
31
32  \_def\_readbiboptions{%
33    \_RetrieveFieldIn{option}\_biboptionsi
34    \_toks1=\_ea{\_biboptionsi}%
35    \_edef\_biboptionsi{\_space \_the\_toks1 \_space \_the\_biboptions \_space}%
36  }
```

Formating of Author/Editor lists.

```
42  \_def\_firstauthorformat{%
43    \_upper{\_Lastname}\_bprintc\_Firstname{, *}\_bprintc\_Von{ *}\_bprintc\_Junior{, *}%
44  }
45  \_def\_otherauthorformat{%
46    \_bprintc\_Firstname{* }\_bprintc\_Von{* }\_upper{\_Lastname}\_bprintc\_Junior{, *}%
47  }
48  \_def\_commonname{%
49   \_ifnum\_NameCount=1
50      \_firstauthorformat
51      \_ifx\_dobibmark\_undefined \_edef\_dobibmark{\_Lastname}\_fi
52   \_else
53      \_ifnum0\_namecount=\_NameCount
```

```
54          \_ifx\_maybeetal\_empty \_bibconjunctionand\_else , \_fi
55        \_else , \_fi
56        \_otherauthorformat
57    \_fi
58 }
59 \_def\_authorname{%
60    \_ifnum\_NameCount>0\_namecount\_relax\_else \_commonname \_fi
61    \_ifnum\_NameCount=0\_namecount\_relax \_maybeetal \_fi
62 }
63 \_let\_editorname=\_authorname
64
65 \_def\_prepareauedoptions#1{%
66    \_def\_mabyetal{}\_csname lb@abbreviatefalse\_endcsname
67    \_biboptionvalue{#1max}\_authormax
68    \_biboptionvalue{#1min}\_authormin
69    \_biboptionvalue{#1pre}\_authorpre
70    \_biboptionvalue{#1print}\_authorprint
71    \_isbiboption{#1etal}\_iftrue \_def\_maybeetal{\_Mtext{bib.etal}}\_fi
72    \_biboptionvalue{#1trim}\_autrim
73    \_let\_namecountraw=\_namecount
74    \_ifx\_authormax\_empty \_else
75        \_ifnum 0\_authormax<0\_namecount
76            \_edef\_namecount{\_ifx\_authormin\_empty\_authormax\_else\_authormin\_fi}%
77            \_def\_maybeetal{\_Mtext{bib.etal}}%
78    \_fi\_fi
79    \_ifx\_autrim\_empty \_def\_autrim{10000}\_fi
80    \_ifnum\_autrim=0 \_def\_autrim{10000}\_fi
81    \_ifnum 0\_namecount<\_autrim\_relax \_else \_AbbreviateFirstname \_fi
82 }
83 \_def\_maybeetal{}
84
85 \_ifx\upper\_undefined
86    \_ifx\caps \_undefined \_def\upper{\_uppercase\_ea}\_else
87                       \_def\upper#1{{\caps\_rm #1}}\_fi
88 \_fi
89 \_let\_upper=\upper
```

Preparing bib-mark (used when \nonumcitations is set).

```
95 \_def\_setbibmark{%
96    \_ifx\_dobibmark\_undefined \_def\_dobibmark{}\_fi
97    \_RetrieveFieldIn{bibmark}\_tmp
98    \_ifx\_tmp\_empty \_RetrieveFieldIn{year}\_tmp \_edef\_tmp{\_dobibmark, \_tmp}\_fi
99    \_bibmark=\_ea{\_tmp}%
100 }
```

Setting phrases.

```
106 \_def\_bibconjunctionand{\_Mtext{bib.and}}
107 \_def\_preurl{\_Mtext{bib.available}}
108 \_let\_predoi=\_preurl
109 \_def\_postedition{\_mtext{bib.edition}}
110 \_def\_Inclause{In:~}
111 \_def\_prevolume{\_mtext{bib.volume}}
112 \_def\_prenumber{\_mtext{bib.number}}
113 \_def\_prepages{\_mtext{bib.prepages}}
114 \_def\_posteditor{\_ifnum0\_namecountraw>1 \_Mtext{bib.editors}\_else\_Mtext{bib.editor}\_fi}
```

_Mtext{⟨*identifier*⟩} expands to a phrase by outer document language (no entry language).

```
121 \_chardef\_documentlanguage=\_language
122 \_def\_Mtext#1{\_csname _mt:#1:\_csname _lan:\_the\_documentlanguage\_endcsname\_endcsname}
123
124 \_CreateField {lang}
125 \_def\_setlang#1{\_ifx#1\_empty \_else
126    \_setbox0=\_vbox{\_langinput{#1}}%
127    \_ifcsname _mt:bib.and:#1\_endcsname
128        \_language=\_csname _#1Patt\_endcsname \_relax
129    \_else \_opwarning{No phrases for "#1" used by [\EntryKey] in .bib}%
130    \_fi\_fi
131 }
```

Non-standard field names.

```
137 \_CreateField {ednote}
138 \_CreateField {citedate}
139 \_CreateField {numbering}
140 \_CreateField {isbn}
141 \_CreateField {issn}
142 \_CreateField {doi}
143 \_CreateField {url}
144 \_CreateField {bibmark}
```

Sorting.

```
150 \_SortingOrder{name,year}{lfvj}
151 \_SpecialSort {key}
```

Supporting macros.

```
157 \_def\_bibwarninga{\_bibwarning}
158 \_def\_bibwarningb{\_bibwarning}
159
160 \_def\_docitedate #1/#2/#3/#4\_relax{[\_Mtext{bib.citedate}%
161     \_if^#2^#1\_else
162         \_if^#3^#1/#2\_else
163             \_cs{\_cs{_lan:\_the\_documentlanguage}dateformat}#1/#2/#3\relax
164     \_fi\_fi ]%
165 }
166 \_def\_doyear#1{
167     \_biboptionvalue{yearprint}\_yearprint
168     \_ifx\_yearprint\_empty#1\_else\_def\YEAR{#1}\_yearprint\_fi
169 }
170 \_def\_preparenumbering{%
171     \_def\VOL{\_RetrieveField{volume}}%
172     \_def\NO{\_RetrieveField{number}}%
173     \_def\PP{\_RetrieveField{pages}}%
174 }
175 \_def\_prepareednote{%
176     \_def\EDN{\_RetrieveField{edition}}%
177     \_def\ADDR{\_RetrieveField{address}}%
178     \_def\PUBL{\_RetrieveField{publisher}}%
179     \_def\YEAR{\_RetrieveField{year}}%
180     \_def\AU{\_bprintb[!author]{\_doauthor0{####1}}{}}%
181     \_def\ED{\_bprintb[!editor]{\_doeditor0{####1}}{}}%
182     \_preparenumbering
183 }
184 \_def\_doedition#1{%
185     \_biboptionvalue{editionprint}\_editionprint
186     \_ifx\_editionprint\_empty#1\_postedition\_else\_def\ED{#1}\_editionprint\_fi
187 }
188 \_def\_doauthor#1#2{\_prepareauedoptions{au}\_let\_iseditorlist=\_undefined
189     \_if1#1\_def\AU{#2}\_else\_let\_authorprint=\_empty\_fi
190     \_ifx\_authorprint\_empty #2\_else \_authorprint\_fi
191 }
192 \_def\_doeditor#1#2{\_prepareauedoptions{ed}\_let\_firstauthorformat=\_otherauthorformat
193     \_if1#1\_def\ED{#2}\_else\_let\_authorprint=\_empty\_fi
194     \_ifx\_authorprint\_empty #2\_posteditor\_else \_authorprint\_fi
195 }
```

Entry types.

```
201 \_sdef{_print:BEGIN}{%
202     \_readbiboptions
203     \_biboptionvalue{titlepost}\_titlepost
204     \_isbiboption{unpublished}\_iftrue \_let\_bibwarninga=\_relax \_let\_bibwarningb=\_relax \_fi
205     \_isbiboption{nowarn}\_iftrue \_let\_bibwarning=\_relax \_fi
206     \_isbiboption{urlalso}\_iftrue \_def\_preurl{\_Mtext{bib.availablealso}}\_fi
207     \_RetrieveFieldIn{lang}\_langentry \_setlang\_langentry
208 }
209 \_sdef{_print:END}{%
210     \_bprinta [note]        {*.}{}%
```

```
211     \_setbibmark
212 }
213 \_def\_bookgeneric#1{%
214     \_bprinta [howpublished]  {[*].\ }{}%
215     \_bprintb [edition]    {\_doedition{##1}\:\ }{}%
216     \_bprinta [ednote]     {*.\ }{}%
217     \_bprinta [address]    {*\_bprintv[publisher]{:}{\_bprintv[year]{,}{.}}\ }{\_bibwarninga}%
218     \_bprinta [publisher]  {*\_bprintv[year]{,}{.}\ }{\_bibwarninga}%
219     \_bprintb [year]       {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}{}}{.}\ }%
220                                                       {\_bibwarning}%
221     \_bprinta [numbering]  {\_preparenumbering*\_bprintv[citedate]{}{\:}\ }{}%
222     \_bprinta [citedate]   {\_docitedate*///\_relax.\ }{}%
223     #1%
224     \_bprinta [series]     {*.\ }{}%
225     \_bprinta [isbn]       {ISBN~*.\ }{\_bibwarningb}%
226     \_bprinta [issn]       {ISSN~*.\ }{}%
227     \_bprintb [doi]        {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
228     \_bprintb [url]        {\_preurl\_url{##1}. }{}%
229 }
230 \_sdef{_print:book}{%
231     \_bprintb [!author]    {\_doauthor1{##1}\:\ }{\_bibwarning}%
232     \_bprintb [title]      {{\_em##1}\_bprintc\_titlepost{\:\ *}\_bprintv[howpublished]{}{\:}\ }%
233                                                       {\_bibwarning}%
234     \_bookgeneric{}%
235 }
236 \_sdef{_print:article}{%
237     \_biboptionvalue{journalpost}\_journalpost
238     \_bprintb [!author]    {\_doauthor1{##1}\:\ }{\_bibwarning}%
239     \_bprinta [title]      {*.\ \_bprintc\_titlepost{*.\ }}{\_bibwarning}%
240     \_bprintb [journal]    {{\_em##1}\_bprintc\_journalpost{\:\ *}\_bprintv[howpublished]{}{\:}\ }%
241                                                       {\_bibwarninga}%
242     \_bprinta [howpublished]  {[*].\ }{}%
243     \_bprinta [address]    {*\_bprintb[publisher]{:}{,}\ }{}%
244     \_bprinta [publisher]  {*, }{}%
245     \_bprinta [month]      {*, }{}%
246     \_bprintb [year]       {\_doyear{##1}\_bprintv[volume,number,pages]{,}{\:}\ }{}%
247     \_bprinta [numbering]  {\_preparenumbering*\_bprintv[citedate]{}{\:}\ }
248                           {\_bprinta [volume]  {\_prevolume*\_bprintv[number,pages]{,}{\:}\ }{}%
249                            \_bprinta [number]  {\_prenumber*\_bprintv[pages]{,}{\:}\ }{}%
250                            \_bprintb [pages]   {\_prepages\_hbox{##1}\_bprintv[citedate]{}{\:}\ }%
251                                                       {\_bibwarninga}}%
252     \_bprinta [citedate]   {\_docitedate*///\_relax.\ }{}%
253     \_bprinta [issn]       {ISSN~*.\ }{}%
254     \_bprintb [doi]        {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
255     \_bprintb [url]        {\_preurl\_url{##1}. }{}%
256 }
257 \_sdef{_print:inbook}{%
258     \_let\_bibwarningb=\_relax
259     \_bprintb [!author]    {\_doauthor1{##1}\:\ }{\_bibwarning}%
260     \_bprinta [title]      {*.\ }{\_bibwarning}%
261                           \_Inclause
262     \_bprintb [!editor]    {\_doeditor1{##1}\:\ }{}%
263     \_bprintb [booktitle] {{\_em##1}\_bprintc\_titlepost{\:\ *}\_bprintv[howpublished]{}{\:}\ }%
264                                                       {\_bibwarning}%
265     \_bookgeneric{\_bprintb [pages]   {\_prepages\_hbox{##1}. }{}}%
266 }
267 \_slet{_print:inproceedings}{_print:inbook}
268 \_slet{_print:conference}{_print:inbook}
269
270 \_sdef{_print:thesis}{%
271     \_bprintb [!author]    {\_doauthor1{##1}\:\ }{\_bibwarning}%
272     \_bprintb [title]      {{\_em##1}\_bprintc\_titlepost{\:\ *}\_bprintv[howpublished]{}{\:}\ }%
273                                                       {\_bibwarning}%
274     \_bprinta [howpublished]  {[*].\ }{}%
275     \_bprinta [address]    {*\_bprintv[school]{:}{\_bprintv[year]{,}{.}}\ }{\_bibwarning}%
276     \_bprinta [school]     {*\_bprintv[year]{,}{.}\ }{\_bibwarning}%
277     \_bprinta [month]      {*, }{}%
278     \_bprintb [year]       {\_doyear{##1}\_bprintv[citedate]{}{.}\ }{\_bibwarninga}%
279     \_bprinta [citedate]   {\_docitedate*///\_relax.\ }{}%
```

```
280    \_bprinta [type]        {*\_bprintv[ednote]{,}{.}\ }%
281                            {\_ifx\_thesistype\_undefined\_bibwarning
282                             \_else\_thesistype\_bprintv[ednote]{,}{.}\ \_fi}%
283    \_bprinta [ednote]      {*.\ }{}%
284    \_bprintb [doi]         {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
285    \_bprintb [url]         {\_preurl\_url{##1}. }{}%
286 }
287 \_sdef{_print:phdthesis}{\_def\_thesistype{\_Mtext{bib.phdthesis}}\_cs{_print:thesis}}
288 \_sdef{_print:mastershesis}{\_def\_thesistype{\_Mtext{bib.masthesis}}\_cs{_print:thesis}}
289 \_sdef{_print:bachelorsthesis}{\_def\_thesistype{\_Mtext{bib.bachthesis}}\_cs{_print:thesis}}
290
291 \_sdef{_print:generic}{%
292    \_bprintb [!author]     {\_doauthor1{##1}\:\ }{\_bibwarning}%
293    \_bprintb [title]       {{\_em##1}\_bprintc\_titlepost{\:\ *}\_bprintv[howpublished]{}{\:\ }%
294                                                                    {\_bibwarning}%
295    \_bprinta [howpublished]  {[*].\ }{}%
296    \_bprinta [ednote]      {\_prepareednote*\_bprintv[citedate]{}{.}\ }{\_bibwarning}%
297    \_bprinta [year]        {}{\_bibwarning}%
298    \_bprinta [citedate]    {\_docitedate*///\_relax.\ }{}%
299    \_bprintb [doi]         {\_predoi DOI \_ulink[http://dx.doi.org/##1]{##1}.\ }{}%
300    \_bprintb [url]         {\_preurl\_url{##1}. }{}%
301 }
302 \_slet{_print:booklet}{_print:generic}
303 \_slet{_print:incolleciton}{_print:generic}
304 \_slet{_print:manual}{_print:generic}
305 \_slet{_print:proceedings}{_print:generic}
306 \_slet{_print:techreport}{_print:generic}
307 \_slet{_print:unpublished}{_print:generic}
308
309 \_sdef{_print:misc}{\_let\_bibwarning=\_relax \_cs{_print:generic}}
```

2.33   Sorting and making Index

```
3 \_codedecl \makeindex {Makeindex and sorting <2022-06-28>} % preloaded in format
```

\makeindex implements sorting algorithm at TEX macro-language level. You need not any external program. The sorting can be used for various other applications, see an example in OpTEX trick 0068.

There are two passes in the sorting algorithm. The primary pass does not distinguish between a group of letters (typically non-accented and accented). If the result of comparing two string is equal in primary pass then the secondary pass is started. It distinguishes between variously accented letters. Czech rules, for example, says: not accented before dieresis before acute before circumflex before ring. At less priority: lowercase letters must be before uppercase letters.

The _sortingdatalatin implements these rules for the languages with latin alphabets. The groups between commas are not distinguished in the first pass. The second pass distinguishes all characters mentioned in the _sortingdatalatin (commas are ignored). The order of letters in the _sortingdatalatin macro is significant for the sorting algorithm.

```
27 \_def \_sortingdatalatin {%
28    /,{ },-,&,@,%
29    aAàÀâÂäÄáÁ,%
30    ąĄ,%
31    bB,%
32    cC,%
33    ćĆčČ,%
34    dDdĎ,%
35    eEèÈéÉëËêÊěĚ,%
36    ęĘ,%
37    fF,%
38    gG,%
39    hH,%
40    ^^T^^U^^V,% ch Ch CH
41    iIíÍïÏîÎ,%
42    jJ,%
43    kK,%
44    lLíÍĺĽ,%
```

```
45    łŁ,%
46    mM,%
47    nNňÑ,%
48    ńŃñÑ,%
49    oOöÜóÓôÔ,%
50    pP,%
51    qQ,%
52    rRŕŔ,%
53    řŘ,%
54    sSß,%
55    śŚšŠ,%
56    tTťŤ,%
57    uUùÙûÛüÜúÚůŮűŰ,%
58    vV,%
59    wW,%
60    xX,%
61    yYýÝÿŸ,%
62    zZ,%
63    žŽ,%
64    źŹ,%
65    żŻ,%
66    ^^Z,% Hungarian: cz:c^^Z, etc., see \_compoundcharshu in lang-data.opm
67    0,1,2,3,4,5,6,7,8,9,'%
68  }
```

Characters to be ignored during sorting are declared in `\_ignoredcharsgeneric`. These characters are ignored in the first pass without additional condition. All characters are taken into account in the second pass: ASCII characters with code < 65 are sorted first if they are not mentioned in the `\_sortingdata...` macro. Others not mentioned characters have undefined behavior during sorting.

```
79  \_def \_ignoredcharsgeneric  {.,;?!:'"|()[]<>=+-}
```

Sorting is always processed by rules of a given language. The macros `\_sortingdata`⟨*lang-tag*⟩, `\_ignoredchars`⟨*lang-tag*⟩ and `\_compoundchars`⟨*lang-tag*⟩ declare these rules. The ⟨*lang-tag*⟩ is ISO code of the language: en, cs, de, pl, es for example. The English language is implemented here. Other languages are implemented in the `lang-data.opm` file (see section 2.37.4).

```
90  \_let \_sortingdataen = \_sortingdatalatin   % English alphabet is subset of Latin
91  \_let \_ignoredcharsen = \_ignoredcharsgeneric
92  \_def \_compoundcharsen {}  % English doesn't have compound characters like DZ
```

The `\_compoundchars`⟨*lang-tag*⟩ can declare changes performed before sorting. For example Czech language declares:

```
\_let \_sortingdatacs = \_sortingdatalatin  % Czech alphabet is subset of Latin
\_def \_compoundcharscs {ch:^^T Ch:^^U CH:^^V}
```

It transforms two-letters `ch` to single character `^^T` because ch is treated as single compound character by Czech rules and CH is sorted between H and I. See `\_sortingdatalatin` where `^^T` is used. This declaration makes more transformations of Ch and CH too. The declarations of the form `x:y` in the `\_compoundchars`⟨*lang-tag*⟩ are separated by space.

You can declare a transformation from single letter to more letters too. For example German rules sets ß equal to ss during sorting:

```
\_let \_sortingdatade = \_sortingdatalatin  % German alphabet is subset of Latin
\_def \_compoundcharsde {ß:ss}
```

If there are two words equal after first pass of sorting: Masse (mass) and Maße (measures) for example, then second pass must decide about the order. DIN 5007, section 6.1 says: ss must be before ß in this case. So, we want to switch off the `\_compoundchars` declaration for the second pass and use the order of s and ß given in `\_sortingdata`. This is possible if the `\_xcompoundchars`⟨*lang-tag*⟩ is defined. It has precedence in the second pass of sorting. We declare for German:

```
\_def \_xcompoundcharsde {}
```

Geman rules mention alternative sorting for phone-books or similar lists of names. The letters ä ö ü should be interpreted as ae, oe and ue. So we get Mueller < Müller < Muff. If this rule is not taken into account, we get Mueller < Muff < Müller. The rule can be implemented by:

```
    \_def \_compoundcharsde {ß:ss Ä:AE Ö:OE Ü:UE ä:ae ö:oe ü:ue}
```

Because u < ü in `\_sortingdata` and because `\_xcompoundcharsde` is empty, we have Mueller < Müller after second pass of the sorting.

You can declare these macros for more languages if you wish to use `\makeindex` with sorting rules with respect to your language. Note: if you need to map compound characters to a character, don't use `^^I`, `^^J` or `^^M` because these characters have very specific category codes.

If you created `\_sortingdata` etc. for your language, please, send them to me. I am ready to add them to the file `lang-data.opm` in a new OpTeX release. See also section 2.37.4.

French sorting rule says: if the words are the same except for accents then accented letters are sorted after unaccented leters but read the words from their end in the second pass. For example corect sorting is: cote < côte < coté < côté. This rule can be activated if the contol sequence `\_secondpass`⟨*lang-tag*⟩ is set to `\_reversewords`. For example, `lang-data.opm` declares `\_let\_secondpassfr=\_reversewords`.

Preparing to primary pass is performed by the `\_setprimarysorting` macro implemented here. The ⟨*lang-tag*⟩ is saved to the `\_sortinglang` macro when sorting is initialized in `\_dosorting` (it is typicaly derived from current `\language` value). The `\_setprimarysorting` is called from `\_dosorting` macro and all processing of sorting is in a group. It sets actual `\_sortingdata`, `\_compoundchars` and `\_ignoredchars` if given language declares them. If not then warning will be printed using `\_nold` macro and English data are used. The `\lccode` of all characters from `\_sortingdata` and `\_ignoredchars` are set. The sorted words will be converted using `\_compoundchars` followed by `\lowercase` before first pass is run.

makeindex.opm

```
164 \_def\_setprimarysorting {%
165    \_ea\_let \_ea\_sortingdata \_csname _sortingdata\_sortinglang\_endcsname
166    \_ea\_let \_ea\_compoundchars \_csname _compoundchars\_sortinglang\_endcsname
167    \_ea\_let \_ea\_ignoredchars \_csname _ignoredchars\_sortinglang\_endcsname
168    \_def\_nold{}%
169    \_ifx \_sortingdata\_relax \_addto\_nold{ sortingdata}%
170        \_let \_sortingdata = \_sortingdataen \_fi
171    \_ifx \_compoundchars\_relax \_addto\_nold{ compoundchars}%
172        \_let \_compoundchars = \_compoundcharsen \_fi
173    \_ifx \_ignoredchars\_relax \_addto\_nold{ ignoredchars}%
174        \_let \_ignoredchars = \_ignoredcharsen \_fi
175    \_ifx\_nold\_empty \_else \_opwarning{Missing\_nold\_space for language (\_sortinglang)}\_fi
176    \_ifx \_compoundchars\_empty \_else
177        \_edef \_compoundchars {\_detokenize\_ea{\_compoundchars} }\_fi % all must be catcode 12
178    \_def \_act ##1{\_ifx##1\_relax \_else
179        \_ifx##1,\_advance\_tmpnum by1
180        \_else \_lccode`##1=\_tmpnum \_fi
181        \_ea\_act \_fi}%
182    \_tmpnum=65 \_ea\_act \_sortingdata \_relax
183    \_def \_act ##1{\_ifx##1\_relax \_else
184        \_lccode`##1=`\^^I
185        \_ea\_act \_fi}%
186    \_ea\_act \_ignoredchars \_relax
187 }
```

Preparing to secondary pass is implemented by the `\_setsecondarysorting` macro.

makeindex.opm

```
193 \_def\_setsecondarysorting {%
194    \_def \_act ##1{\_ifx##1\_relax \_else
195        \_ifx##1,\_else \_advance\_tmpnum by1 \_lccode`##1=\_tmpnum \_fi
196        \_ea\_act \_fi}%
197    \_tmpnum=64 \_ea\_act \_sortingdata \_relax
198 }
```

Strings to be sorted are prepared in `\,`⟨*string*⟩ control sequences (to save `\TeX` memory). The `\_preparesorting \,`⟨*string*⟩ converts ⟨*string*⟩ to `\_tmpb` with respect to the data initialized in `\_setprimarysorting` or `\_setsecondarysorting`.

The compoud characters are converted by the `\_docompound` macro.

makeindex.opm

```
209 \_def \_preparesorting #1{%
210    \_edef \_tmpb {\_ea\_ignoreit\_csstring #1}% \,<string> -> <string>
211    \_ea \_docompound \_compoundchars \_relax:{}      % replace compound characters
212    \_lowercase \_ea{\_ea\_def \_ea\_tmpb \_ea{\_tmpb}}% convert in respect to \_sortingdata
```

174

```
213    \_ea\_replstring \_ea\_tmpb \_ea{\_csstring\^^I}{}% remove ignored characters
214 }
215 \_def \_docompound #1:#2 {%
216    \_ifx\_relax#1\_else \_replstring\_tmpb {#1}{#2}\_ea\_docompound \_fi
217 }
```

Macro **_isAleB** \,⟨*string1*⟩ \,⟨*string2*⟩ returns the result of comparison of given two strings to **_ifAleB** control sequence. Usage: **_isAleB** \,⟨*string1*⟩ \,⟨*string2*⟩ **_ifAleB ... _else ... _fi** The converted strings (in respect of the data prepared for first pass) must be saved as values of \,⟨*string1*⟩ and \,⟨*string2*⟩ macros. The reason is speed: we don't want to convert them repeatedly in each comparison. The macro **_testAleB** ⟨*converted-string1*⟩&_relax⟨*converted-string2*⟩&_relax \,⟨*string1*⟩\,⟨*string2*⟩ does the real work. It reads the first character from both converted strings, compares them and if it is equal then calls itself recursively else gives the result.

```
234 \_newifi \_ifAleB
235
236 \_def\_isAleB #1#2{%
237    \_edef\_tmpb {#1&\_relax#2&\_relax}%
238    \_ea \_testAleB \_tmpb #1#2%
239 }
240 \_def\_testAleB #1#2\_relax #3#4\_relax #5#6{%
241    \_if #1#3\_if #1&\_testAleBsecondary #5#6%   goto to the second pass::
242        \_else \_testAleB #2\_relax #4\_relax #5#6%
243        \_fi
244    \_else \_ifnum `#1<`#3 \_AleBtrue \_else \_AleBfalse \_fi
245    \_fi
246 }
```

The **_testAleBsecondary** \,⟨*string1*⟩ \,⟨*string2*⟩ is run if the words are equal in the primary pass. It runs **_setsecondarysorting** if it was not initialized already. Then prepares compared words to _tmpa and _tmpb and corrects them by **_prepsecondpass** if needed. Finaly, the test is recursively done by the macro **_testAleBsecondaryX** ⟨*converted-string1*⟩0_relax⟨*converted-string2*⟩1_relax

```
257
258 \_def\_testAleBsecondary#1#2{%
259    \_setsecondarysorting \_let\_setsecondarysorting=\_relax
260    \_preparesorting#1\_let\_tmpa=\_tmpb \_preparesorting#2%
261    \_prepsecondpass
262    \_edef\_tmpb{\_tmpa0\_relax\_tmpb1\_relax}%
263    \_ea\_testAleBsecondaryX \_tmpb
264 }
265 \_def\_testAleBsecondaryX #1#2\_relax #3#4\_relax {%
266    \_if #1#3\_testAleBsecondaryX #2\_relax #4\_relax
267    \_else \_ifnum `#1<`#3 \_AleBtrue \_else \_AleBfalse \_fi
268    \_fi
269 }
```

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 a comma when it starts. It ends with sorted items in _iilist without commas. So **_dosorting** macro must prepare commas between items.

```
279 \_def\_mergesort #1#2,#3{% by Miroslav Olsak
280    \_ifx,#1%                    % prazdna-skupina,neco,  (#2=neco #3=pokracovani)
281        \_addto\_iilist{#2,}%        % dvojice skupin vyresena
282        \_sortreturn{\_fif\_mergesort#3}%   % \mergesort pokracovani
283    \_fi
284    \_ifx,#3%                    % neco,prazna-skupina,  (#1#2=neco #3=,)
285        \_addto\_iilist{#1#2,}%      % dvojice skupin vyresena
286        \_sortreturn{\_fif\_mergesort}%     % \mergesort dalsi
287    \_fi
288    \_ifx\_end#3%                 % neco,konec (#1#2=neco)
289        \_ifx\_empty\_iilist              % neco=kompletni setrideny seznam
290            \_def\_iilist{#1#2}%
291            \_sortreturn{\_fif\_fif\_gobbletoend}%   % koncim
292        \_else                    % neco=posledni skupina nebo \end
293            \_sortreturn{\_fif\_fif      % spojim \indexbuffer+necoa cele znova
```

```
294                        \_edef\_iilist{\_ea}\_ea\_mergesort\_iilist#1#2,#3}%
295     \_fi\_fi                    % zatriduji: p1+neco1,p2+neco2, (#1#2=p1+neco1 #3=p2)
296     \_isAleB #1#3\_ifAleB       % p1<p2
297        \_addto\_iilist{#1}%       % p1 do bufferu
298        \_sortreturn{\_fif\_mergesort#2,#3}%          % \mergesort neco1,p2+neco2,
299     \_else                     % p1>p2
300        \_addto\_iilist{#3}%       % p2 do bufferu
301        \_sortreturn{\_fif\_mergesort#1#2,}%          % \mergesort p1+neco1,neco2,
302     \_fi
303     \_relax % zarazka, na ktere se zastavi \sortreturn
304 }
305 \_def\_sortreturn#1#2\_fi\_relax{#1} \_def\_fif{\_fi}
306 \_def\_gobbletoend #1\_end{}
```

The `\_dosorting` `\list` macro redefines `\list` as sorted `\list`. The `\list` have to include control sequences in the form `\⟨c⟩⟨string⟩`. These control sequences will be sorted with respect to ⟨*strings*⟩ without change of meanings of these control sequences. Their meanings are irrelevant when sorting. The first character ⟨*c*⟩ in `\⟨c⟩⟨string⟩` should be whatever. It does not influence the sorting. OpTEX uses comma at this place for sorting indexes: `\,⟨word1⟩ \,⟨word2⟩ \,⟨word3⟩ …`.

The current language (chosen for hyphenation patterns) is used for sorting data. If the macro `\_sortinglang` is defined as ⟨*lang-tag*⟩ (for example `\def\_sortinglang{de}` for German) then this has precedence and current language is not used. Moreover, if you specify `\_asciisortingtrue` then ASCII sorting will be processed and all language sorting data will be ignored.

```
325 \_newifi \_ifasciisorting  \_asciisortingfalse
326 \_def\_dosorting #1{%
327    \_begingroup
328       \_ifasciisorting \_def\_sortinglang{ASCII}\_fi
329       \_ifx\_sortinglang\_undefined \_edef\_sortinglang{\_cs{_lan:\_the\_language}}\_fi
330       \_message{OpTeX: Sorting \_string#1 (\_sortinglang) ...^^J}%
331       \_ismacro\_sortinglang{ASCII}\_iftrue
332          \_def \_preparesorting##1{\_edef\_tmpb{\_ea\_ignoreit\_csstring##1}}%
333          \_let \_setsecondarysorting=\_relax
334       \_else
335          \_setprimarysorting
336       \_fi
337       \_def \_act##1{\_preparesorting ##1\_edef##1{\_tmpb}}%
338       \_ea\_xargs \_ea\_act #1;% \_preparesorting for first pass of sorting applied
339       \_ifcsname _xcompoundchars\_sortinglang\_endcsname
340          \_ea\_let \_ea\_compoundchars \_csname _xcompoundchars\_sortinglang\_endcsname
341       \_fi  % \_compoundchars can differ in the second pass of sorting
342       \_csname _secondpass\_sortinglang \_endcsname % activates \_reversewords if needed
343       \_def \_act##1{\_addto #1{##1,}}%
344       \_edef #1{\_ea}\_ea\_xargs \_ea\_act #1;% commas between items added, mergesort initialized
345       \_edef \_iilist{\_ea}\_ea\_mergesort #1\_end,\_end
346    \_ea\_endgroup
347    \_ea\_def\_ea#1\_ea{\_iilist}%
348 }
```

French rules needs reverese reading the words in the second pass. The `\_reversewords` is activated in this case and it adds new job to the macro `\_prepsecondpass`: it reverses the letters in the compared words (saved in `\_tmpa` and `\_tmpb`) by the expandable `\_sortrevers` macro. The `\_prepsecondpass` macro is used in the `\_testAleBsecondary` and it is empty by default.

```
359 \_def\_prepsecondpass{}
360 \_def\_reversewords{%
361    \_addto\_prepsecondpass{\_edef\_tmpa{\_ea\_sortrevers\_tmpa\_relax}%
362                          \_edef\_tmpb{\_ea\_sortrevers\_tmpb\_relax}}%
363 }
364 \_def\_sortrevers #1#2\_relax{\_ifx^#2^#1\_else \_sortrevers#2\_relax #1\_fi}
```

The `\makeindex` prints the index. First, it sorts the `\_iilist` second, it prints the sorted `\_iilist`, each item is printed using `\_printindexitem`.
We set `\leftskip=\iindent` and we suppose that each index entry starts by `\noindent\hskip-\iindent` (see the macro `\_printii`). Then the next lines of the same index entry (if the page list is broken to more pages) is indented by `\leftskip=\iindent`.

```
377 \_def\_makeindex{\_par
378   \_ifx\_iilist\_empty \_opwarning{index data-buffer is empty. TeX me again}%
379   \_incr\_unresolvedrefs
380   \_else
381     \_dosorting \_iilist % sorting \_iilist
382     \_bgroup
383       \_rightskip=0pt plus1fil \_exhyphenpenalty=10000 \_leftskip=\_iindent
384       \_ea\_xargs \_ea\_printindexitem \_iilist ;\_par
385     \_egroup
386   \_fi
387 }
388 \_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.

```
398 \_def\_printindexitem #1{%
399   \_ifcsname _\_csstring #1\_endcsname
400     \_ea\_ea\_ea \_printii \_csname _\_csstring #1\_endcsname &%
401   \_else
402     \_ea\_ea\_ea\_printii \_ea\_ignoreit \_csstring #1&%
403   \_fi
404   \_ea\_printiipages #1&
405 }
```

`\_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}
```

```
422 \_def\_printii #1#2&{%
423   \_ismacro\_lastii{#1}\_iffalse \_newiiletter{#1}{#2}\_def\_lastii{#1}\_fi
424   \_gdef\_currii{#1#2}\_the\_everyii\_noindent
425   \_hskip-\_iindent \_ignorespaces\_printiiA#1#2//}
426 \_def\_printiiA #1/{\_if^#1^\_let\_previi=\_currii \_else
427   \_ea\_scanprevii\_previi/&\_edef\_tmpb{\_detokenize{#1}}%
428   \_ifx\_tmpa\_tmpb \_iiemdash \_else#1 \_gdef\_previi{}\_fi
429   \_ea\_printiiA\_fi
430 }
431 \_def\_iiemdash{\_kern.1em---\_space}
432 \_def\_lastii{}
433 \_def\_newiiletter#1#2{}
434
435 \_def\_scanprevii#1/#2&{\_def\_previi{#2}\_edef\_tmpa{\_detokenize{#1}}}
436 \_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 solve these tasks.

```
450 \_def\_printiipages#1&{\_let\_pgtype=\_undefined \_tmpnum=0 \_printpages #1,:,\_par}
451 \_def\_printpages#1:#2,{%  state automaton for compriming pages
452   \_ifx,#1,\_uselastpgnum
453   \_else \_def\_tmpa{#2}%
454     \_ifx\_pgtype\_tmpa \_else
455       \_let\_pgtype=\_tmpa
456       \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
457       \_tmpnum=#1 \_returnfi \_fi
458     \_ifnum\_tmpnum=#1 \_returnfi \_fi
```

```
459        \_advance\_tmpnum by1
460        \_ifnum\_tmpnum=#1 \_ifx\_lastpgnum\_undefined \_usepgdash\_fi
461                      \_edef\_lastpgnum{\_the\_tmpnum:{\_pgtype}}%
462                      \_returnfi \_fi
463        \_uselastpgnum \_usepgcomma \_pgprint#1:{#2}%
464        \_tmpnum=#1
465        \_relax
466     \_ea\_printpages \_fi
467 }
468 \_def\_returnfi #1\_relax{\_fi}
469 \_def\_uselastpgnum{\_ifx\_lastpgnum\_undefined
470     \_else \_ea\_pgprint\_lastpgnum \_let\_lastpgnum=\_undefined \_fi
471 }
472 \_def\_usepgcomma{\_ifnum\_tmpnum>0, \_fi} % comma+space between page numbers
473 \_def\_usepgdash{\_hbox{--}}              % dash in the <from>--<to> form
```

You can re-define **_pgprint** ⟨*gpageno*⟩:{⟨*iitype*⟩} if you need to implement more ⟨*iitypes*⟩.

```
480 \_def\_pgprint #1:#2{%
481     \_ifx ,#2,\_pgprintA{#1}\_returnfi \_fi
482     \_ifx b#2{\_bf \_pgprintA{#1}}\_returnfi \_fi
483     \_ifx i#2{\_it \_pgprintA{#1}}\_returnfi \_fi
484     \_ifx u#2\_pgu{\_pgprintA{#1}}\_returnfi \_fi
485     \_pgprintA{#1}\_relax
486 }
487 \_def\_pgprintA #1{\_ilink[pg:#1]{\_cs{_pgi:#1}}} % \ilink[pg:<gpageno>]{<pageno>}
488 \_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 other variants of indexing macros expand internally to **\iindex**.

```
496 \_def\_iindex#1{\_isempty{#1}\_iffalse
497     \_openref{\_def~{ }\_ewref\_Xindex{{#1}{\_iitypesaved}}}\_fi}
498 \_public \iindex ;
```

The **_Xindex**{⟨*word*⟩}{⟨*iitype*⟩} stores **\,**⟨*word*⟩ to the **_iilist** if there is the 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. Moreover, 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*⟩.

```
510 \_def \_iilist {}
511 \_def \_Xindex #1#2{\_ea\_XindexA \_csname ,#1\_ea\_endcsname \_currpage {#2}}
512 \_def \_XindexA #1#2#3#4{% #1=\,<word> #2=<gpageno> #3=<pageno> #4=<iitype>
513     \_ifx#1\_relax \_global\_addto \_iilist {#1}%
514                 \_gdef #1{#2:#4}%
515     \_else \_global\_addto #1{,#2:#4}%
516     \_fi
517     \_sxdef{_pgi:#2}{#3}%
518 }
```

The implementation of macros **\ii**, **\iid**, **\iis** follows. Note that **\ii** works in the horizontal mode in order to the **\write** whatsit is not broken from the following word. If you need to keep vertical mode, use **\iindex**{⟨*word*⟩} directly.

The **\iitype** {⟨*type*⟩} saves the ⟨*type*⟩ to the **_iitypesaved** macro. It is used in the **\iindex** macro.

```
530 \_def\_ii #1 {\_leavevmode\_def\_tmp{#1}\_iiA #1,,\_def\_iitypesaved{}}
531
532 \_def\_iiA #1,{\_if$#1$\_else\_def\_tmpa{#1}%
533     \_ifx\_tmpa\_iiatsign \_ea\_iiB\_tmp,,\_else\_iindex{#1}\_fi
534     \_ea\_iiA\_fi}
535 \_def\_iiatsign{@}
536
537 \_def\_iiB #1,{\_if$#1$\_else \_iiC#1/\_relax \_ea\_iiB\_fi}
538 \_def\_iiC #1/#2\_relax{\_if$#2$\_else\_iindex{#2#1}\_fi}
539
540 \_def\_iid #1 {\_leavevmode\_iindex{#1}\_def\_iitypesaved{}#1\_futurelet\_tmp\_iiD}
541 \_def\_iiD{\_ifx\_tmp,\_else\_ifx\_tmp.\_else\_space\_fi\_fi}
```

```
542  \_def\_iis #1 #2{{\_def~{ }\_global\_sdef{_,#1}{#2}}\_ignorespaces}
543
544  \_def\_iitypesaved{}
545
546  \_def\_iitype #1{\_def\_iitypesaved{#1}\_ignorespaces}
547
548  \_public \ii \iid \iis \iitype ;
```

2.34 Footnotes and marginal notes

```
3  \_codedecl \fnote {Footnotes, marginal notes OpTeX <2020-05-26>} % preloaded in format
```

_gfnotenum is a counter which counts footnotes globally in the whole document.
_lfnotenum is a counter which counts footnotes at each chapter from one. It is used for local page footnote counters too.
_ifpgfnote says that footnote numbers are counted on each page from one. We need to run **\openref** in this case.
\fnotenum is a macro that expands to footnote number counted in declared part.
\fnotenumchapters declares footnotes numbered in each chapter from one (default), **\fnotenumglobal** declares footnotes numbered in whole document from one and **\fnotenumpages** declares footnotes numbered at each page from one.

```
18  \_newcount\_gfnotenum \_gfnotenum=0
19  \_newcount\_lfnotenum
20
21  \_newifi \_ifpgfnote
22  \_def \_fnotenumglobal   {\_def\_fnotenum{\_the\_gfnotenum}\_pgfnotefalse}
23  \_def \_fnotenumchapters {\_def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse}
24  \_def \_fnotenumpages    {\_def\_fnotenum{\_trycs{_fn:\_the\_gfnotenum}{?}}\_pgfnotetrue}
25  \_fnotenumchapters  % default are footnotes counted from one in each chapter
26  \_def \fnotenum{\_fnotenum}
27  \_public \fnotenumglobal \fnotenumchapters \fnotenumpages ;
28  \_let \runningfnotes = \fnotenumglobal % for backward compatibility
```

The **_printfnotemark** prints the footnote mark. You can re-define this macro if you want another design of footnotes. For example

```
\fnotenumpages
\def \_printfnotemark {\ifcase 0\fnotenum\or
    *\or**\or***\or$^\mathbox{†}$\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 the footnote itself, then you can define **_printfnotemarkA** and **_printfnotemarkB**.

The **\fnotelinks**⟨*colorA*⟩⟨*colorB*⟩ implements the hyperlinked footnotes (from text to footnote and backward).

```
48  \_def \_printfnotemark  {$^{\_fnotenum}$}   % default footnote mark
49  \_def \_printfnotemarkA {\_printfnotemark}  % footnote marks used in text
50  \_def \_printfnotemarkB {\_printfnotemark}  % footnote marks used in front of footnotes
51
52  \_def \_fnotelinks#1#2{% <inText color> <inFootnote color>
53     \_def\_printfnotemarkA{\_link[fnt:\_the\_gfnotenum]{#1}{\_printfnotemark}%
54                         \_dest[fnf:\_the\_gfnotenum]}%
55     \_def\_printfnotemarkB{\_link[fnf:\_the\_gfnotenum]{#2}{\_printfnotemark}%
56                         \_dest[fnt:\_the\_gfnotenum]}%
57  }
58  \_public \fnotelinks ;
```

Each footnote saves the **_Xfnote** (without parameter) to the `.ref` file (if **\openref**). We can create the mapping from ⟨*gfnotenum*⟩ to⟨*pgfnotenum*⟩ in the macro **_fn:**⟨*fnotenum*⟩. Each **_Xpage** macro sets the **_lfnotenum** to zero.

```
67  \_def \_Xfnote {\_incr\_lfnotenum \_incr\_gfnotenum
68      \_sxdef{_fn:\_the\_gfnotenum}{\_the\_lfnotenum}}
```

The \fnote {⟨*text*⟩} macro is simple, \fnotemark and \fnotetext does the real work.

```
75  \_def\_fnote{\_fnotemark1\_fnotetext}
76  \_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 can propagate a macro parameter into the Insert group before the text is printed (see section 2.18). This propagated macro is _fnset which sets smaller fonts.

Note that \vfootnote and _opfootnote don't read the text as a parameter but during the normal horizontal mode. This is the reason why catcode changes (for example in-line verbatim) can be used here.

```
90  \_def\_fnotetext{\_incr\_gfnotenum \_incr\_lfnotenum % global increment
91      \_ifpgfnote \_openref \_fi
92      \_wref \_Xfnote{}%
93      \_ifpgfnote \_ifcsname _fn:\_the\_gfnotenum \_endcsname \_else
94          \_opwarning{unknown \_noexpand\fnote mark. TeX me again}%
95          \_incr\_unresolvedrefs
96      \_fi\_fi
97      \_opfootnote\_fnset\_printfnotemarkB
98  }
99  \_def\_fnset{\_everypar={}\_scalemain \_typoscale[800/800]}
100
101 \_public \fnote \fnotemark \fnotetext ;
```

By default \mnote{⟨*text*⟩} are in right margin at odd pages and they are in left margin at even pages. The \mnote macro saves its position to .ref file as _Xmnote without parameter. We define _mn:⟨*mnotenum*⟩ as \right or \left when the .ref file is read. The \ifnum 0≤0#2 trick returns true if ⟨*pageno*⟩ has a numeric type and false if it is a non-numeric type (Roman numeral, for example). We prefer to use ⟨*pageno*⟩, but only if it has the numeric type. We use ⟨*gpageno*⟩ in other cases.

```
113 \_newcount\_mnotenum    \_mnotenum=0        % global counter of mnotes
114 \_def \_Xmnote {\_incr\_mnotenum \_ea \_XmnoteA \_currpage}
115 \_def \_XmnoteA #1#2{% #1=<gpageno> #2=<pageno>
116     \_sxdef{mn:\_the\_mnotenum}{\_ifodd\_numtype{#2}{#1} \_right \_else \_left \_fi}}
117 \_def \_numtype #1#2{\_ifnum 0<0#1 #1\_else #2\_fi}
```

User can declare \fixmnotes\left or \fixmnotes\right. It defines _mnotesfixed as _left or _right which declares the placement of all marginal notes and such declaration has a precedence.

```
125 \_def \_fixmnotes #1{\_edef\_mnotesfixed{\_cs{\_csstring #1}}}
126 \_public \fixmnotes ;
```

The _mnoteD{⟨*text*⟩} macro sets the position of the marginal note. The outer box of marginal note has zero width and zero depth and it is appended after current line using \vadjust primitive or it is inverted to vertical mode as a box shifted down by \parskip and with \vskip-\baselineskip followed.

```
135 \_def\_mnote #1#{\_ifx^#1^\_else \_mnoteC#1\_end \_fi \_mnoteD}
136 \_def\_mnoteC up#1\_end{\_mnoteskip=#1\_relax} % \mnote up<dimen> {<text>} syntax
137 \_long\_def\_mnoteD#1{%
138     \_ifvmode \_vskip\_parskip{\_mnoteA{#1}}\_nobreak\_vskip-\_baselineskip\_vskip-\_parskip \_else
139     \_lower\_dp\_strutbox\_hbox{}\_vadjust{\_kern-\_dp\_strutbox \_mnoteA{#1}\_kern\_dp\_strutbox}%
140     \_fi
141 }
142 \_public \mnote ;
```

The \mnoteskip is a dimen value that denotes the vertical shift of marginal note from its normal position. A positive value means shift up, negative down. The \mnoteskip register is set to zero after the marginal note is printed. The new syntax \mnote up⟨*dimen*⟩{⟨*text*⟩} is possible too, but public \mnoteskip is kept for backward compatibility.

```
152 \_newdimen\_mnoteskip
153 \_public \mnoteskip ;
```

The `\_mnoteA` macro does the real work. The `\_lrmnote{`⟨*left*⟩`}{`⟨*right*⟩`}` uses only first or only second parameter depending on the left or right marginal note.

```
161 \_long\_def\_mnoteA #1{\_incr\_mnotenum
162    \_ifx\_mnotesfixed\_undefined
163       \_ifcsname _mn:\_the\_mnotenum \_endcsname
164          \_edef\_mnotesfixed{\_cs{_mn:\_the\_mnotenum}}%
165       \_else
166          \_opwarning{unknown \_noexpand\mnote side. TeX me again}\_openref
167          \_incr\_unresolvedrefs
168          \_def\_mnotesfixed{\_right}%
169    \_fi\_fi
170    \_hbox to0pt{\_wref\_Xmnote{}\_everypar={}%
171       \_lrmnote{\_kern-\_mnotesize \_kern-\_mnoteindent}{\_kern\_hsize \_kern\_mnoteindent}%
172       \_vbox to0pt{\_vss \_setbox0=\_vtop{\_hsize=\_mnotesize
173             \_lrmnote{\_leftskip=0pt plus 1fill \_rightskip=0pt}
174                      {\_rightskip=0pt plus 1fil \_leftskip=0pt}%
175          {\_the\_everymnote\_noindent#1\_endgraf}}%
176          \_dp0=0pt \_box0 \_kern\_mnoteskip \_global\_mnoteskip=0pt}\_hss}%
177 }
178 \_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.

```
185 \_regmacro {\_def\fnote#1{}} {\_def\fnote#1{}} {\_def\fnote#1{}}
186 \_regmacro {\_def\fnotemark#1{}} {\_def\fnotemark#1{}} {\_def\fnotemark#1{}}
187 \_regmacro {\_def\mnote#1{}} {\_def\mnote#1{}} {\_def\mnote#1{}}
```

2.35   Styles

OpTeX provides three styles: `\report`, `\letter` and `\slides`. Their behavior is documented in user part of the manual in the section 1.7.2 and `\slides` style (for presentations) is documented in `op-slides.pdf` which is an example of the presentation.

2.35.1   `\report` and `\letter` styles

```
3 \_codedecl \report {Basic styles of OpTeX <2021-03-10>} % preloaded in format
```

We define auxiliary macro first (used by the `\address` macro)
The `{\boxlines` ⟨*line-1*⟩⟨*eol*⟩⟨*line-2*⟩⟨*eol*⟩. . . ⟨*line-n*⟩⟨*eol*⟩`}` returns to the outer vertical mode a box with ⟨*line-1*⟩, next box with ⟨*line-2*⟩ etc. Each box has its natural width. This is reason why we cannot use paragraph mode where each resulting box has the width `\hsize`. The ⟨*eol*⟩ is set active and `\everypar` starts `\hbox{` and acive ⟨*eol*⟩ closes this `\hbox` by `}`.

```
16 \_def\_boxlines{%
17    \_def\_boxlinesE{\_ifhmode\_egroup\_empty\_fi}%
18    \_def\_nl{\_boxlinesE}%
19    \_bgroup \_lccode`\~=`\^^M\_lowercase{\_egroup\_let~}\_boxlinesE
20    \_everypar{\_setbox0=\_lastbox\_endgraf
21       \_hbox\_bgroup \_catcode`\^^M=13 \_let\_par=\_nl \_aftergroup\_boxlinesC}%
22 }
23 \_def\_boxlinesC{\_futurelet\_next\_boxlinesD}
24 \_def\_boxlinesD{\_ifx\_next\_empty\_else\_ea\_egroup\_fi}
25
26 \_public \boxlines ;
```

The `\report` style initialization macro is defined here.

```
32 \_def\_report{
33    \_typosize[11/13.2]
34    \_vsize=\_dimexpr \_topskip + 52\_baselineskip \_relax % added 2020-03-28
35    \_let\_titfont=\_chapfont
36    \_titskip=3ex
37    \_eoldef\_author##1{\_removelastskip\_bigskip
38       {\_leftskip=0pt plus1fill \_rightskip=\_leftskip \_it \_noindent ##1\_par}\_nobreak\_bigskip
39    }
40    \_public \author ;
41    \_parindent=1.2em \_iindent=\_parindent \_ttindent=\_parindent
42    \_footline={\_global\_footline={\_hss\_rmfixed\_folio\_hss}}
43 }
```

The \letter style initialization macro is defined here.

The \letter defines \address and \subject macros.

See the files demo/op-letter-*.tex for usage examples.

```
53 \_def\_letter{
54    \_def\_address{\_vtop\_bgroup\_boxlines \_parskip=0pt \_let\_par=\_egroup}
55    \_def\_subject{{\_bf \_mtext{subj}: }}
56    \_public \address \subject ;
57    \_typosize[11/14]
58    \_vsize=\_dimexpr \_topskip + 49\_baselineskip \_relax % added 2020-03-28
59    \_parindent=0pt
60    \_parskip=\_medskipamount
61    \_nopagenumbers
62 }
63 \_public \letter \report ;
```

The \slides macro reads macro file slides.opm, see the section 2.35.2.

```
69 \_def\_slides{\_par
70   \_opinput{slides.opm}
71   \_adef*{\_relax\_ifmmode*\_else\_ea\_startitem\_fi}
72 }
73 \_public \slides ;
```

2.35.2 \slides style for presentations

```
3 \_codedecl \slideshow {Slides style for OpTeX <2022-05-12>} % loaded on demand by \slides
```

Default margins and design is declared here. The _ttfont is scaled by mag1.15 in order to balance the ex height of Helvetica (Heros) and LM fonts Typewriter. The \begtt...\endtt verbatim is printed by smaller text.

```
12 \_margins/1 a5l (14,14,10,3)mm  % landscape A5 format
13 \_def\_wideformat{\_margins/1 (263,148) (16,16,10,3)mm } % 16:9 format
14
15 \_ifx\_fontnamegen\_undefined \_fontfam[Heros]
16    \_let\_ttfont=\_undefined \_famvardef\_ttfont{\_setfontsize{mag1.15}\_tt}
17 \_fi
18 \_typosize[16/19]
19 \_def\_urlfont{}
20 \_everytt={\_typosize[13/16] \_advance\_hsize by10mm}
21 \_fontdef\_fixbf{\_bf}
22
23 \_nopagenumbers
24 \_parindent=0pt
25 \_ttindent=5mm
26 \_parskip=5pt plus 4pt minus2pt
27 \_rightskip=0pt plus 1fil
28 \_ttindent=10pt
29 \_def\_ttskip{\_smallskip}
30 \_let\_scolor=\Blue  % secondary color used in default design
31
32 \_onlyrgb   % RGB color space is better for presentations
```

The bottom margin is set to 3 mm. If we use 1 mm, then the baseline of \footline is 2 mm from the bottom page. This is the depth of the \Grey rectangle used for page numbers. It is r-lapped to \hoffset width because left margin = \hoffset = right margin. It is 14 mm for narrow pages or 16 mm for wide pages.

```
42 \_footlinedist=1mm
43 \_footline={\_hss \_rlap{%
44    \_rlap{\Grey\_kern.2\_hoffset\_vrule height6mm depth2mm width.8\_hoffset}%
45                   \_hbox to\_hoffset{\White\_hss\_folio\_kern3mm}}}
```

The \subtit is defined analogically like \tit.

```
51 \_eoldef\_subtit#1{\_vskip20pt {\_leftskip=0pt plus1fill \_rightskip=\_leftskip
52    \_subtitfont #1\_nbpar}}
```

The `\pshow`⟨*num*⟩ prints the text in invisible (transparent) font when `\layernum`<⟨*num*⟩. For transparency we need to define special graphics states.

```
60 \_def\_Transparent {\_transparency255 }
61 \_public \Transparent ;
62
63 \_def\_use#1#2{\_ifnum\_layernum#1\_relax#2\_fi}
64 \_def\_pshow#1{\_use{=#1}\Red \_use{<#1}\_Transparent \_ignorespaces}
```

The main level list of items is activated here. The `\_item:X` and `\_item:x` are used and are re-defined here. If we are in a nested level of items and `\pg+` is used then `\egroups` macro expands to the right number of `\egroup`s to close the page correctly. The level of nested item lists is saved to the `\_ilevel` register and used when we start again the next text after `\pg+`.

```
76 \_newcount\_gilevel
77 \_def\*{*}
78 \_adef*{\_relax\_ifmmode*\_else\_ea\_startitem\_fi} % defined also in styles.opm
79 \_sdef{_item:X}{\_scolor\_raise.2ex\_fullrectangle{.8ex}\_kern.5em}
80 \_sdef{_item:x}{\_scolor\_raise.3ex\_fullrectangle{.6ex}\_kern.4em}
81 \_style X
82 \_def\_egroups{\_par\_global\_gilevel=\_ilevel \_egroup}
83 \_everylist={\_novspaces \_ifcase\_ilevel \_or \_style x \_else \_style - \_fi
84    \_addto\_egroups{\_egroup}}
```

The default values of `\pg`, i. e. `\pg;`, `\pg+` and `\pg.` are very simple. They are used when `\showslides` is not specified.

```
91 \_def\_pg#1{\_cs{_spg:#1}}
92 \_sdef{_spg:;}{\_vfil\_break \_lfnotenumreset}
93 \_sdef{_spg:.}{\_endslides}
94 \_sdef{_spg:+}{\_par}
```

The `\_endslides` is defined as `\_end` primitive (preceded by `\_byehook`), but slide-designer can redefine it. For example, OpTeX trick 0029 shows how to define clickable navigation to the pages and how to check the data integrity at the end of the document using `\_endslides`.

The `\bye` macro is redefined here as an alternative to `\pg.`.

```
106 \_def\_endslides{\_vfill \_supereject \_byehook \_end}
107 \_def\bye{\_pg.}
```

We need no numbers and no table of contents when using slides. The `\_printsec` macro is redefined in order the title is centered and typeset in `\_scolor`.

```
115 \_def\_titfont{\_typosize[42/60]\_bf \_scolor}
116 \_def\_subtitfont{\_typosize[20/30]\_bf}
117 \_def\_secfont{\_typosize[25/30]\_bf \_scolor}
118
119 \_nonum \_notoc \_let\_resetnonumnotoc=\_relax
120 \_def\_printsec#1{\_par
121    \_abovetitle{\_penalty-400}\_bigskip
122    {\_secfont \_noindent \_leftskip=0pt plus1fill \_rightskip=\_leftskip
123       \_printrefnum[@\_quad]#1\_nbpar}\_insertmark{#1}%
124    \_nobreak \_belowtitle{\_medskip}%
125 }
```

When `\slideshow` is active then each page is opened by `\setbox\_slidepage=\vbox\bgroup` (roughly speaking) and closed by `\egroup`. The material is `\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 and prints page by the `\_printlayers`

```
143 \_newbox\_slidepage  \_newbox\_slidepageB
144 \_countdef\_slidelayer=1

145
146 \_def\_slideshow{\_slidelayer=1 \_slideshowactive
147    \_let\slideopen=\_relax  % first wins
148    \_setbox\_slidepage=\_vbox\_bgroup\_bgroup}

149
150 \_def\_slideshowactive{%
151    \_sdef{_spg:;}{\_closepage \_global\_slidelayer=1 \_resetpage \_openslide}
152    \_sdef{_spg:.}{\_closepage \_endslides}
153    \_sdef{_spg:+}{\_closepage \_incr\_slidelayer \_decr\_pageno \_openslide}
154    \_let\_layers=\_layersactive
155    \_slidelinks % to prevent hyperlink-dests duplication
156 }
157 \_def\_openslide{\_setbox\_slidepage=\_vbox\_bgroup\_bgroup \_setilevel
158    \_ifvoid\_slidepage \_else \_unvbox\_slidepage \_nointerlineskip\_lastbox \_fi}
159 \_def\_setilevel{\_loop \_decr\_gilevel \_ifnum\_gilevel<0 \_else \_begitems \_repeat}

160
161 \_def\_closepage{\_egroups \_egroup
162    \_ifnum \_maxlayers=0 \_unvcopy\_slidepage \_vfil\_break
163    \_else \_begingroup \_setwarnslides \_layernum=0
164       \_loop
165          \_ifnum\_layernum<\_maxlayers \_advance\_layernum by1
166             \_printlayers \_vfil\_break
167             \_ifnum\_layernum<\_maxlayers \_incr\_slidelayer \_decr\_pageno \_fi
168       \_repeat
169       \_global\_maxlayers=0
170       \_incr\_layernum \_global\_setbox\_slidepage=\_vbox{\_printlayers}%
171       \_endgroup
172    \_fi}
173 \_def\_resetpage{%
174    \_global\_setbox\_slidepage=\_box\_voidbox \_global\_setbox\_slidepageB=\_box\_voidbox
175    \_lfnotenumreset
176 }
177 \_def\_setwarnslides{%
178    \_def\pg##1{\_opwarning{\_string\pg##1 \_layersenv}\_def\pg####1{}}%
179    \_def\layers##1 {\_opwarning{\_string\layers\_space \_layersenv}\_def\layers####1{}}%
180 }
181 \_def\_layersenv{cannot be inside \_string\layers...\_string\endlayers, ignored}

182
183 \_def\_printlayers{\_unvcopy\_slidepage \_prevdepth=\_dp\_slidepage
184    {\_layertext \_endgraf}%
185    \_vskip\_parskip
186    \_unvcopy\_slidepageB
187 }
188 \_let\_destboxori=\_destbox

189
190 \_newcount\_layernum \_newcount\_maxlayers
191 \_maxlayers=0

192
193 \_long\_def\_layersactive #1 #2\endlayers{%
194    \_par\_penalty0\_egroup\_egroup
195    \_gdef\_layertext{\_settinglayer#2}%
196    \_global\_maxlayers=#1
197    \_setbox\_slidepageB=\_vbox\_bgroup\_bgroup
198       \_setbox0=\_vbox{{\_layernum=1 \_globaldefs=-1 \_layertext\_endgraf}}\_prevdepth=\_dp0
199 }
200 \_public \subtit \slideshow \pg \wideformat \use \pshow \layernum ;
```

`\slideopen` should be used instead `\slideshow` to deactivate it but keep the borders of groups.

```
207 \_def\_slideopen{\_let\slideshow=\_relax % first wins
208    \_sdef{_spg:;}{\_egroups\_vfil\_break \_lfnotenumreset\_bgroup \_setilevel}
209    \_sdef{_spg:.}{\_egroups\_endslides}
210    \_sdef{_spg:+}{\_egroups\_bgroup \_setilevel}
211    \_let\_layersopen=\_egroup \_let\_layersclose=\_bgroup
212    \_bgroup
213 }
214 \_public \slideopen ;
```

When \slideshow is active then the destinations of internal hyperlinks cannot be duplicated to more "virtual" pages because hyperlink destinations have to be unique in the whole document.

The \slideshow creates boxes of typesetting material and copies them to more pages. So, we have to suppress creating destinations in these boxes. This is done in the _slidelinks macro. We can move creating these destinations to the output routine. _sdestbox is saved value of the original _destbox which is redefined to do only \addto_destboxes{_sdestbox[⟨label⟩]}. All destinations saved to _destboxes are created at the start of the next output routine in the _pagedest macro. The output routine removes _destboxes, so each destination is created only once.

Limitations of this solution: destinations are only at the start of the page, no at the real place where \wlabel was used. The first "virtual" page where \wlabel is used includes its destination. If you want to go to the final page of the partially uncovering ideas then use \label[⟨label⟩]\wlabel{text} in the last part of the page (before \pg;) o use \pgref instead \ref.

```
239  \_def\_slidelinks{%
240     \_def \_destbox[##1]{\_edef\_tmp{\_noexpand\_sdestbox[##1]}%
241        \_global\_ea\_addto\_ea\_destboxes\_ea{\_tmp}}%
242     \_def \_pagedest {%
243        \_hbox{\_def\_destheight{25pt}\_sdestbox[pg:\_the\_gpageno]\_destboxes}%
244        \_nointerlineskip \_gdef\_destboxes{}%
245     }%
246     \_ifx \_dest\_destactive \_else \_let\_pagedest=\_relax \_fi
247  }
248  \_let\_sdestbox = \_destbox
249  \_def\_destboxes{}    % initial value of \_destboxes
250  \_let\_bibgl=\_global % \advance\bibnum must be global if they are at more pages
```

The _settinglayer is used in the _layertext macro to prevent printing "Duplicate label" warning when it is expanded. It is done by special value of _slideshook (used by the \label macro). Moreover, the warning about illegal use of \bib, \usebib in \layers environment is activated.

```
260  \_def\_settinglayer{%
261     \_def\_slideshook ##1##2{}%
262     \_def\_bibB[##1]{\_nousebib}\_def\_usebib/##1 (##2) ##3 {\_nousebib}%
263  }
264  \_def\_nousebib{\_opwarning{Don't use \noexpand\bib nor \noexpand\usebib in \string\layers}}
```

Default \layers ⟨num⟩ macro (when \slideshow is not activated) is simple. It prints the ⟨layered-text⟩ with \layernum=⟨num⟩+1 because we need the result after last layer is processed.

```
272  \_long\_def\_layers #1 #2\endlayers{\_par
273     \_layersopen {\_layernum=\_numexpr#1+1\_relax #2\_endgraf}\_layersclose}
274  \_let\_layersopen=\_relax
275  \_let\_layersclose=\_relax
276
277  \_def\layers{\_layers}
```

We must to redefine \fnotenumpages because the data from .ref file are less usable for implementing such a feature: the footnote should be in more layers repeatedly. But we can suppose that each page starts by \pg; macro, so we can reset the footnote counter by this macro.

```
287  \_def \_fnotenumpages {\_def\_fnotenum{\_the\_lfnotenum}\_pgfnotefalse
288     \_def\_lfnotenumreset{\_global\_lfnotenum=0 }}
289  \_let \_lfnotenumreset=\_relax
290  \_public \fnotenumpages ;
```

2.36 Logos

```
3  \_codedecl \TeX {Logos TeX, LuaTeX, etc. <2020-02-28>} % preloaded in format
```

Despite plain TeX each macro for logos ends by \ignoreslash. This macro ignores the next slash if it is present. You can use \TeX/ like this for protecting the space following the logo. This is visually more comfortable. The macros \TeX, \OpTeX, \LuaTeX, \XeTeX are defined.

```
13  \_protected\_def \_TeX {T\_kern-.1667em\_lower.5ex\_hbox{E}\_kern-.125emX\_ignoreslash}
14  \_protected\_def \_OpTeX {Op\_kern-.1em\_TeX}
15  \_protected\_def \_LuaTeX {Lua\_TeX}
16  \_protected\_def \_XeTeX {X\_kern-.125em\_phantom E%
17      \_pdfsave\_rlap{\_pdfscale{-1}{1}\_lower.5ex\_hbox{E}}\_pdfrestore \_kern-.1667em \_TeX}
18
19  \_def\_ignoreslash {\_isnextchar/\_ignoreit{}}
20
21  \_public \TeX \OpTeX \LuaTeX \XeTeX \ignoreslash ;
```

The `\_slantcorr` macro expands to the slant-correction of the current font. It is used to shifting A if the `\LaTeX` logo is in italic.

```
28  \_protected\_def \_LaTeX{\_tmpdim=.42ex L\_kern-.36em \_kern \_slantcorr % slant correction
29      \_raise \_tmpdim \_hbox{\_thefontscale[710]A}%
30      \_kern-.15em \_kern-\_slantcorr \_TeX}
31  \_def\_slantcorr{\_ea\_ignorept \_the\_fontdimen1\_font\_tmpdim}
32
33  \_public \LaTeX ;
```

`\OPmac`, `\CS` and `\csplain` logos.

```
39  \_def\_OPmac{\_leavevmode
40      \_lower.2ex\_hbox{\_thefontscale[1400]O}\_kern-.86em P{\_em mac}\_ignoreslash}
41  \_def\_CS{$\_cal C$\_kern-.1667em\_lower.5ex\_hbox{$\_cal S$}\_ignoreslash}
42  \_def\_csplain{\_CS plain\_ignoreslash}
43
44  \_public \OPmac \CS \csplain ;
```

The expandable versions of logos used in Outlines need the expandable `\ingnslash` (instead of the `\ignoreslash`).

```
51  \_def\_ignslash#1{\_ifx/#1\_else #1\_fi}
52  \_regmacro {}{}{% conversion for PDF outlines
53      \_def\TeX{TeX\_ignslash}\_def\OpTeX{OpTeX\_ignslash}%
54      \_def\LuaTeX{LuaTeX\_ignslash}\_def\XeTeX{XeTeX\_ignslash}%
55      \_def\LaTeX{LaTeX\_ignslash}\_def\OPmac{OPmac\_ignslash}%
56      \_def\CS{CS}\_def\csplain{csplain\_ignslash}%
57  }
58  \_public \ignslash ;
```

2.37 Multilingual support

2.37.1 Lowercase, 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 `\catcode`=12 for all codes above 127.

The file `UnicodeData.txt` is read if this file exists in your TeX distribution. The format is specified at http://www.unicode.org/L2/L1999/UnicodeData.html. We read only `Ll` (lowercase letters), `Lu` (upperase letters) and `Lo` (other letters) and set appopriate codes. The scanner of `UnicodeData.txt` is implemented here in the group (lines 6 to 15). After the group is closed then the file `uni-lcuc.opm` is leaved by `\endinput`.

If the file `UnicodeData.txt` does not exsists then internal data are used. They follow to the end of the file `uni-lcuc.opm`.

```
3   \_wlog{Setting lccodes and uccodes for Unicode characters <2021-04-07>} % preloaded in format.
4
5   \_isfile{UnicodeData.txt}\_iftrue
6   \_begingroup
7       \_sdef{lc:Ll}#1#2#3#4{\_global\_lccode"#2="#2  \_global\_uccode"#2="0#3 }
8       \_sdef{lc:Lu}#1#2#3#4{\_global\_lccode"#2="0#4 \_global\_uccode"#2="#2 }
9       \_sdef{lc:Lo}#1#2#3#4{\_global\_lccode"#2="#2  \_global\_uccode"#2="#2 }
10      \_def\_pa#1;#2;#3;#4;#5;#6;#7;#8;#9;{\_ifx;#1;\_else\_ea\_pb\_fi{#1}{#3}}
11      \_def\_pb#1#2#3;#4;#5;#6;#7;#8 {\_csname lc:#2\_endcsname\_pc{#1}{#6}{#7}\_pa}
12      \_def\_pc#1#2#3{}       % ignored if the character hasn't Ll, Lu, nor Lo type
```

```
13     \_everyeof={;;;;;;;;;;} % end of file
14     \_ea\_pa\_input UnicodeData.txt
15  \_endgroup \_endinput \_fi  % \endinput here, if UnicodeData.txt was loaded
16
17  % If UnicodeData.txt not found, we have internal copy here from csplain, 2014:
18
19  \_def\_tmp #1 #2 {\_ifx^#1^\_else
20     \_lccode"#1="#1
21     \_ifx.#2%
22        \_uccode"#1="#1
23     \_else
24        \_uccode"#2="#2
25        \_lccode"#2="#1
26        \_uccode"#1="#2
27     \_fi
28     \_ea \_tmp \_fi
29  }
30  \_tmp
31  00AA .
32  00B5 039C
33  00BA .
34  00E0 00C0
35  00E1 00C1
36  00E2 00C2
37  00E3 00C3
38  00E4 00C4
39  00E5 00C5
```

...etc., 15900 similar lines (see uni-lcuc.opm)

2.37.2 Multilingual phrases and quotation marks

languages.opm

```
3  \_codedecl \_mtext {Languages <2022-02-19>} % preloaded in format
```

Four words are generated by OpTEX macros: "Chapter", "Table", "Figure" and "Subject". These phrases are generated depending on the current value of the \language register, if you use _mtext{⟨phrase-id⟩}, specially _mtext{chap}, _mtext{t}, _mtext{f} or _mtext{subj}. If your macros generate more words then you can define such words by \sdef{_mt:⟨phrase-id⟩:⟨lang-tag⟩} where ⟨phrase-id⟩ is a label for the declared word and ⟨lang-tag⟩ is a language shortcut declared by _preplang.

languages.opm

```
16  \_def\_mtext#1{\_trycs{_mt:#1:\_trycs{_lan:\_the\_language}{en}}
17     {\_csname _mt:#1:en\_endcsname}}
```

We can declare such language-dependent words by

```
\_sdef{_mt:chap:en}{Chapter} \_sdef{_mt:chap:cs}{Kapitola}
\_sdef{_mt:t:en}{Table}     \_sdef{_mt:t:cs}{Tabulka}
```

etc. but we use more "compact" macro _langw ⟨lang-tag⟩ ⟨chapter⟩ ⟨table⟩ ⟨figure⟩ ⟨subject⟩ for declaring them.

languages.opm

```
30  \_def \_langw #1 #2 #3 #4 #5 {%
31     \_sdef{_mt:chap:#1}{#2}\_sdef{_mt:t:#1}{#3}\_sdef{_mt:f:#1}{#4}%
32     \_sdef{_mt:subj:#1}{#5}%
33  }
```

More phrases are auto-generated in bibliography references. They are declared by
_langb ⟨lang-tag⟩ {⟨and⟩} {⟨et-al⟩} {⟨ed⟩} {⟨cit⟩} {⟨vol⟩} {⟨no⟩} {⟨pp⟩} {⟨p⟩} {⟨ed⟩} {⟨eds⟩}
{⟨avail-from⟩} {⟨avali-to⟩} {⟨ba-thesis⟩} {⟨ma-thesis⟩} {⟨phd-thesis⟩}. It is used similar way as the
_langw above. Both these macros are used in lang-data.opm file, see the end of section 2.37.3.

languages.opm

```
43  \_def\_langb#1 #2#3#4#5#6#7#8#9{\_def\_mbib##1##2{\_sdef{_mt:bib.##2:#1}{##1}}%
44     \_mbib{#2}{and}\_mbib{#3}{etal}\_mbib{#4}{edition}\_mbib{#5}{citedate}\_mbib{#6}{volume}%
45     \_mbib{#7}{number}\_mbib{#8}{prepages}\_mbib{#9}{postpages}\_langbA}
46  \_def\_langbA#1#2#3#4#5#6#7{\_mbib{#1}{editor}\_mbib{#2}{editors}\_mbib{#3}{available}%
47     \_mbib{#4}{availablealso}\_mbib{#5}{bachthesis}\_mbib{#6}{masthesis}\_mbib{#7}{phdthesis}}
```

\today macro needs auto-generated words for each name of the month.
_monthw ⟨lang-tag⟩ ⟨January⟩ ⟨February⟩ ... ⟨December⟩ is used for decaring them.
The language-dependent format for printing date should be declared like

_sdef{_mt:today:en}{_mtext{m_the_month} _the_day, _the_year}

This example declares date format for English where ⟨*lang-tag*⟩ is en.

```
60  \_def \_monthw #1 #2 #3 #4 #5 #6 #7 {%
61      \_sdef{_mt:m1:#1}{#2}\_sdef{_mt:m2:#1}{#3}\_sdef{_mt:m3:#1}{#4}%
62      \_sdef{_mt:m4:#1}{#5}\_sdef{_mt:m5:#1}{#6}\_sdef{_mt:m6:#1}{#7}%
63      \_monthwB #1
64  }
65  \_def \_monthwB #1 #2 #3 #4 #5 #6 #7 {%
66      \_sdef{_mt:m7:#1}{#2}\_sdef{_mt:m8:#1}{#3}\_sdef{_mt:m9:#1}{#4}%
67      \_sdef{_mt:m10:#1}{#5}\_sdef{_mt:m11:#1}{#6}\_sdef{_mt:m12:#1}{#7}%
68  }
69  \_def\_today{\_mtext{today}}
70  \_public \today ;
```

Quotes should be tagged by \"⟨*text*⟩" and \'⟨*text*⟩' if \⟨*iso-code*⟩quotes is declared at beginning of the document (for example \enquotes). If not, then the control sequences \" and \' are undefined. Remember, that they are used in another meaning when the \oldaccents command is used. The macros \" and \' are not defined as \protected because we need their expansion when \outlines are created. User can declare quotes by \quoteschars⟨*clqq*⟩⟨*crqq*⟩⟨*clq*⟩⟨*crq*⟩, where ⟨*clqq*⟩...⟨*crqq*⟩ are normal quotes and ⟨*clq*⟩...⟨*crq*⟩ are alternative quotes. or use \altquotes to swap between the meaning of these two types of quotes. \enquotes, \csquotes, \frquotes, \dequotes, \skquotes are defined here. Languages in general provide the \quotes declaration macro. It declares the quotation marks depending on the actual selected language. For example, \eslang \quotes declares Spanish language including its quotation marks used for \"⟨*text*⟩" and \'⟨*text*⟩'. The language-dependent quotation marks should be declared by _quotationmarks ⟨*lang-tag*⟩ {⟨*clqq*⟩⟨*crqq*⟩⟨*clq*⟩⟨*crq*⟩} in the lang-data.opm file.

```
92  \_def \_enquotes {\_quoteschars ""''}
93  \_def \_csquotes {\_quoteschars „"''}
94  \_def \_frquotes {\_quoteschars ""«»}
95  \_let \_dequotes = \_csquotes
96  \_let \_skquotes = \_csquotes
97
98  \_def \_quotes {\_trycs{_qt:\_trycs{_lan:\_the\_language}{en}}{\_enquotes}}
99  \_def \_quotationmarks #1 #2{\_sdef{_qt:#1}{\_quoteschars #2}}
100
101  \_public \quotes \enquotes \csquotes \frquotes \dequotes \skquotes ;
```

The \quoteschars⟨*lqq*⟩⟨*rqq*⟩⟨*lq*⟩⟨*rq*⟩ defines \" and \" as _qqA in normal mode and as expadable macros in outline mode. We want to well process the common cases: \"`&`" or \"`{`". This is the reason why the quotes parameter is read in verbatim mode and retokenized again by \scantextokens. We want to allow to quote the quotes mark itself by \"{`"`}". This is the reason why the sub-verbatim mode is used when the first character is { in the parameter.

The \" is defined as _qqA_qqB⟨*lqq*⟩⟨*rqq*⟩ and \' as _qqA_qqC⟨*lq*⟩⟨*rq*⟩. The _qqA_qqB⟨*clqq*⟩⟨*crqq*⟩ runs _qqB⟨*lqq*⟩⟨*rqq*⟩⟨*text*⟩".

The _regquotes\"""⟨*L*⟩⟨*R*⟩ does \def\"#1"{⟨*L*⟩#1⟨*R*⟩} for outlines but the " separator is active (because " and ' are active in \pdfunidef).

```
117  \_def \_quoteschars #1#2#3#4{\_def\_altquotes{\_quoteschars#3#4#1#2}\_public\altquotes;%
118      \_protected\_def \"{\_qqA\_qqB#1#2}\_protected\_def \'{\_qqA\_qqC#3#4}%
119      \_regmacro{}{}{\_regquotes\""#1#2\_regquotes\''#3#4}}
120
121  \_def\_qqA#1#2#3{\_bgroup\_setverb \_catcode`\ =10
122      \_isnextchar\_bgroup{\_catcode`\{=1 \_catcode`\}=2 #1#2#3}{#1#2#3}}
123  \_long\_def\_qqB#1#2#3"{\_egroup#1\_scantextokens{#3}#2}
124  \_long\_def\_qqC#1#2#3'{\_egroup#1\_scantextokens{#3}#2}
125  \_def\_regquotes#1#2#3#4{\_bgroup \_lccode`~=`#2\_lowercase{\_egroup \_def#1##1~}{#3##1#4}}
```

Sometimes should be usable to leave the markup "such" or 'such' i.e. without the first backslash. Then you can make the characters " and ' active by the \activequotes macro and leave quotes without the first backslash. First, declare \⟨*iso-code*⟩quotes, then \altquotes (if needed) and finally \activequotes.

```
135  \_def\_activequotes{\_let\_actqq=\"\_adef"{\_actqq}\_let\_actq=\'\_adef'{\_actq}%
136      \_regmacro{}{}{\_adef"{\"}\_adef'{\'}}}
137
138  \_public \quoteschars \activequotes ;
```

2.37.3 Languages declaration

```
3  \_codedecl \langlist {Languages declaration <2022-02-19>} % preloaded in format
```

_preplang ⟨*lang-id*⟩ ⟨*LongName*⟩ ⟨*lang-tag*⟩ ⟨*hyph-tag*⟩ ⟨*lr-hyph*⟩ declares a new language. The parameters (separated by space) are

- ⟨*lang-id*⟩: language identifier. It should be derived from ISO 639-1 code but additional letters can be eventually added because ⟨*lang-id*⟩ must be used uniquely in the whole declaration list. The _preplang macro creates the language switch _⟨*lang-id*⟩lang and defines also \⟨*lang-id*⟩lang as a macro which expands to _⟨*lang-id*⟩lang. For example, _preplang cs Czech ... creates _cslang as the language switch and defines \def\cslang{_cslang}.
- ⟨*LongName*⟩: full name of the language.
- ⟨*lang-tag*⟩: language tag, which is used for setting language-dependent phrases and sorting data. If a language have two or more hyphenation patterns but a single phrases set, then we declare this language more than once with the same ⟨*lang-tag*⟩ but different ⟨*lang-hyph*⟩.
- ⟨*hyph-tag*⟩: a part of the file name where the hyphenation patterns are prepared in Unicode. The full file name is hyph-⟨*hyph-tag*⟩.tex. If ⟨*hyph-tag*⟩ is {} then no hyphenation patterns are loaded.
- ⟨*lr-hyph*⟩: two digits, they denote \lefthyphenmin and \righthyphenmin values.

preplang allocates a new internal number by \newlanguage⟨*lang-id*⟩Patt which will be bound to the hyphenation patterns. But the patterns nor other language data are not read at this moment. The _⟨*lang-id*⟩lang is defined as _langinit. When the _⟨*lang-id*⟩lang switch is used firstly in a document then the language is initialized, i.e. hyphenation patterns and language-dependent data are read. The _⟨*lang-id*⟩lang is re-defined itself after such initialization. _preplang does also \def_ulan:⟨*longname*⟩ {⟨*lang-id*⟩}, this is needed for the \uselanguage macro.

```
37  \_def\_preplang #1 #2 #3 #4 #5#6{% lang-id LongName lang-tag hyph-tag lr-hyph
38     \_ifcsname _#1lang\_endcsname \_else
39        \_ea\_newlanguage\_csname _#1Patt\_endcsname
40        \_xdef\_langlist{\_langlist\_space#1(#2)}%
41     \_fi
42     \_lowercase{\_sxdef{_ulan:#2}}{#1}%
43     \_slet{_#1lang}{_relax}%
44     \_sxdef {#1lang}{\_cs{_#1lang}}%
45     \_sxdef {_#1lang}{\_noexpand\_langinit \_cs{_#1lang}#1(#2)#3[#4]#5#6}%
46  }
```

The _preplang macro adds ⟨*lang-id*⟩(⟨*LongName*⟩) to the _langlist macro which is accessible by \langlist. It can be used for reporting declared languages.

```
53  \_def\langlist{\_langlist}
54  \_def\_langlist{en(USEnglish)}
```

All languages with hyphenation patterns provided by TEXlive are declared here. The language switches \cslang, \sklang, \delang, \pllang and many others are declared. You can declare more languages by _preplang in your document, if you want.

The usage of _preplang with ⟨*lang-id*⟩ already declared is allowed. The language is re-declared in this case. This can be used in your document before first usage of the \⟨*lang-id*⟩lang switch.

```
67  %           lang-id  LongName       lang-tag  hyph-tag     lr-hyph
68  \_preplang enus      USenglishmax   en        en-us        23
69  % Europe:
70  \_preplang engb      UKenglish      en        en-gb        23
71  \_preplang be        Belarusian     be        be           22
72  \_preplang bg        Bulgarian      bg        bg           22
73  \_preplang ca        Catalan        ca        ca           22
74  \_preplang hr        Croatian       hr        hr           22
75  \_preplang cs        Czech          cs        cs           23
76  \_preplang da        Danish         da        da           22
77  \_preplang nl        Dutch          nl        nl           22
78  \_preplang et        Estonian       et        et           23
79  \_preplang fi        Finnish        fi        fi           22
80  \_preplang fis       schoolFinnish  fi        fi-x-school  11
81  \_preplang fr        French         fr        fr           22
```

```
 82  \_preplang de        nGerman         de       de-1996         22
 83  \_preplang deo       oldGerman       de       de-1901         22
 84  \_preplang gsw       swissGerman     de       de-ch-1901      22
 85  \_preplang elm       monoGreek       el       el-monoton      11
 86  \_preplang elp       Greek           el       el-polyton      11
 87  \_preplang grc       ancientGreek    grc      grc             11
 88  \_preplang hu        Hungarian       hu       hu              22
 89  \_preplang is        Icelandic       is       is              22
 90  \_preplang ga        Irish           ga       ga              23
 91  \_preplang it        Italian         it       it              22
 92  \_preplang la        Latin           la       la              22
 93  \_preplang lac       classicLatin    la       la-x-classic    22
 94  \_preplang lal       liturgicalLatin la       la-x-liturgic   22
 95  \_preplang lv        Latvian         lv       lv              22
 96  \_preplang lt        Lithuanian      lt       lt              22
 97  \_preplang mk        Macedonian      mk       mk              22
 98  \_preplang pl        Polish          pl       pl              22
 99  \_preplang pt        Portuguese      pt       pt              23
100  \_preplang ro        Romanian        ro       ro              22
101  \_preplang rm        Romansh         rm       rm              22
102  \_preplang ru        Russian         ru       ru              22
103  \_preplang srl       Serbian         sr-latn  sh-latn         22
104  \_preplang src       SerbianCyrl     sr-cyrl  sh-cyrl         22
105  \_preplang sk        Slovak          sk       sk              23
106  \_preplang sl        Slovenian       sl       sl              22
107  \_preplang es        Spanish         es       es              22
108  \_preplang sv        Swedish         sv       sv              22
109  \_preplang uk        Ukrainian       uk       uk              22
110  \_preplang cy        Welsh           cy       cy              23
111  % Others:
112  \_preplang af        Afrikaans       af       af              12
113  \_preplang hy        Armenian        hy       hy              12
114  \_preplang as        Assamese        as       as              11
115  \_preplang eu        Basque          eu       eu              22
116  \_preplang bn        Bengali         bn       bn              11
117  \_preplang nb        Bokmal          nb       nb              22
118  \_preplang cop       Coptic          cop      cop             11
119  \_preplang cu        churchslavonic  cu       cu              12
120  \_preplang eo        Esperanto       eo       eo              22
121  \_preplang ethi      Ethiopic        ethi     mul-ethi        11
122  \_preplang fur       Friulan         fur      fur             22
123  \_preplang gl        Galician        gl       gl              22
124  \_preplang ka        Georgian        ka       ka              12
125  \_preplang gu        Gujarati        gu       gu              11
126  \_preplang hi        Hindi           hi       hi              11
127  \_preplang id        Indonesian      id       id              22
128  \_preplang ia        Interlingua     ia       ia              22
129  \_preplang kn        Kannada         kn       kn              11
130  \_preplang kmr       Kurmanji        kmr      kmr             22
131  \_preplang ml        Malayalam       ml       ml              11
132  \_preplang mr        Marathi         mr       mr              11
133  \_preplang mn        Mongolian       mn       mn-cyrl         22
134  \_preplang nn        Nynorsk         nn       nn              22
135  \_preplang oc        Occitan         oc       oc              22
136  \_preplang or        Oriya           or       or              11
137  \_preplang pi        Pali            pi       pi              12
138  \_preplang pa        Panjabi         pa       pa              11
139  \_preplang pms       Piedmontese     pms      pms             22
140  \_preplang zh        Pinyin          zh       zh-latn-pinyin  11
141  \_preplang sa        Sanskrit        sa       sa              13
142  \_preplang ta        Tamil           ta       ta              11
143  \_preplang te        Telugu          te       te              11
144  \_preplang th        Thai            th       th              23
145  \_preplang tr        Turkish         tr       tr              22
146  \_preplang tk        Turkmen         tk       tk              22
147  \_preplang hsb       Uppersorbian    hsb      hsb             22
```

_preplangmore ⟨lang-id⟩⟨space⟩{⟨text⟩} declares more activities of the language switch. The ⟨text⟩ is
processed whenever _⟨lang-id⟩lang is invoked. If _preplangmore is not declared for given language

then `\_langdefault` is processed.

You can implement selecting a required script for given language, for example:

```
\_preplangmore ru {\_frenchspacing \_setff{script=cyrl}\selectcyrlfont}
\_addto\_langdefaut {\_setff{}\selectlatnfont}
```

The macros `\selectcyrlfont` and `\selectlatnfont` are not defined in OpTEX. If you follow this example, you have to define them after your decision what fonts will be used in your specific situation.

```
165 \_def\_preplangmore #1 #2{\_ea \_gdef \_csname _langspecific:#1\_endcsname{#2}}
166 \_def\_langdefault{\_frenchspacing}
167
168 \_preplangmore en   {\_nonfrenchspacing}
169 \_preplangmore enus {\_nonfrenchspacing}
170 \_def\_langdefault  {\_frenchspacing}
```

The default `\language=0` is US-English with original hyphenation patterns preloaded in the format (see the end of section 2.10). We define `\_enlang` and `\enlang` switches. Note that if no language switch is used in the document then `\language=0` and US-English patterns are used, but `\nonfrenchspacing` isn't set.

```
181 \_chardef\_enPatt=0
182 \_sdef{_lan:0}{en}
183 \_sdef{_ulan:usenglish}{en}
184 \_def\_enlang{\_uselang{en}\_enPatt23} % \lefthyph=2 \righthyph=3
185 \_def\enlang{\_enlang}
```

The list of declared languages are reported during format generation.

```
191 \_message{Declared languages: \_langlist.
192    Use \_string\<lang-id>lang to initialize language,
193    \_string\cslang\_space for example.}
```

Each language switch `\_⟨lang-id⟩lang` defined by `\_preplang` has its initial state
`\_langinit \⟨switch⟩ ⟨lang-id⟩(⟨LongName⟩)⟨lang-tag⟩[⟨hyph-tag⟩]⟨lr-hyph⟩`. The `\_langinit` macro does:

- The internal language ⟨number⟩ is extracted from `\_the\_⟨lang-id⟩Patt`.
- `\def \_lan:⟨number⟩ {⟨lang-tag⟩}` for mapping from `\language` number to the ⟨lang-tag⟩.
- loads `hyph-⟨hyph-tag⟩.tex` file with hyphenation patterns when `\language=⟨number⟩`.
- loads the part of `lang-data.opm` file with language-dependent phrases using `\_langinput`.
- `\def \_⟨lang-id⟩lang {\_uselang{⟨lang-id⟩}\_⟨lang-id⟩Patt ⟨lr-hyph⟩}`, i.e. the switch redefines itself for doing a "normal job" when the language switch is used repeatedly.
- Runs itself (i.e. `\_⟨lang-id⟩lang`) again for doing the "normal job" firstly.

```
212 \_def\_langinit #1#2(#3)#4[#5]#6#7{% \_switch lang-id(LongName)lang-tag[hyph-file]lr-hyph
213    \_sxdef{_lan:\_ea\_the\_csname _#2Patt\_endcsname}{#4}%
214    \_begingroup \_setbox0=\_vbox{% we don't want spaces in horizontal mode
215       \_setctable\_optexcatcodes
216       % loading patterns:
217       \_language=\cs{_#2Patt}\_relax
218       \_ifx^#5^\_else
219          \_wlog{Loading hyphenation for #3: \_string\language=\_the\_language\_space(#5)}%
220          \_let\patterns=\_patterns \_let\hyphenation=\_hyphenation \_def\message##1{}%
221          \_isfile {hyph-#5}\_iftrue \_input{hyph-#5}%
222          \_else \_opwarning{No hyph. patterns #5 for #3, missing package?}\_fi
223       \_fi
224       % loading language data:
225       \_langinput{#4}%
226    }\_endgroup
227    \xdef#1{\noexpand\_uselang{#2}\_csname _#2Patt\_endcsname #6#7}%
228    #1% do language switch
229 }
```

`\_uselang{⟨lang-id⟩}\_⟨lang-id⟩Patt ⟨pre-hyph⟩⟨post-hyph⟩` is used as "normal job" of the switch. It sets `\language`, `\lefthyphenmin`, `\righthyphenmin`. Finally, it runs data from `\_preplangmore` or runs `\_langdefault`.

191

```
238 \_def\_uselang#1#2#3#4{\_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\_relax
239    \_trycs{_langspecific:#1}{\_langdefault}%
240 }
```

The \uselanguage {⟨*LongName*⟩} macro is defined here (for compatibility with e-plain users). Its parameter is case insensitive.

```
247 \_def\_uselanguage#1{\_def\_tmp{#1}\_lowercase{\_cs{\_trycs{_ulan:#1}{0x}lang}}}
248 \_sdef{_0xlang}{\_opwarning{\_string\uselanguage{\_tmp}: Unknown language name, ignored}}
249 \_public \uselanguage ;
```

2.37.4 Data for various languages

The "language data" include declarations of rules for sorting (see section 2.33), language-dependent phrases and quotation marks (see section 2.37.2). The language data are collected in the single lang-data.opm file. Appropriate parts of this file is read by _langinput{⟨*lang-tag*⟩}. First few lines of the file looks like:

```
 3 \_codedecl \_langdata {Language dependent data <2022-06-28>} % only en, cs preloaded in format
 4
 5 \_langdata en {English} % ----------------------------------------------
 6 \_langw en  Chapter      Table       Figure       Subject
 7 \_langb en  {, and } { et al.} { ed.} {cit.~} {Vol.~} {No.~} {pp.~} {~p.} {,~ed.} {,~eds.}
 8             {Available from } {Available also from }
 9             {Bachelor's Thesis} {Master's Thesis} {Ph.D. Thesis}
10 \_monthw en January February March April May June
11            July August September October November December
12 \_sdef{_mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
13 \_quotationmarks en {""''}
14
15 %\_let \_sortingdataen =  \_sortingdatalatin % set already, see section 2.33, makeindex.opm
16 %\_let \_ignoredcharsen = \_ignoredcharsgeneric
17 %\_def \_compoundcharsen {}
18
19 \_langdata cs {Czech}   % ----------------------------------------------
20 %            Chapter      Table       Figure       Subject
21 \_langw cs  Kapitola     Tabulka     Obrázek      Věc
22 %            {, and } { et al.} { ed.} {cit.~} {Vol.~} {No.~} {pp.~} {~p.} {,~ed.} {,~eds.}
23 %            {Available from } {Available also from }
24 %            {Bachelor's Thesis} {Master's Thesis} {Ph.D. Thesis}
25 \_langb cs  { a } { a~kol.} { vyd.} {vid.~} {ročník~} {č.~} {s.~} {~s.} {,~editor} {,~editoři}
26            {Dostupné na } {Dostupné též na }
27            {Bakalářská práce} {Diplomová práce} {Disertační práce}
28 %            January February March April May June
29 %            July August September October November December
30 \_monthw cs ledna února března dubna května června
31            července srpna září října listopadu prosince
32 \_sdef{_mt:today:cs}{\_the\_day.~\_mtext{m\_the\_month} \_the\_year} % date format
33 \_quotationmarks cs {„",'}
34
35 \_let \_sortingdatacs  = \_sortingdatalatin
36 \_let \_ignoredcharscs = \_ignoredcharsgeneric
37 \_def \_compoundcharscs {ch:^^T Ch:^^U CH:^^V} % see \_compoundchars in section 2.33
38
39
40 \_langdata de {German}  % ----------------------------------------------
41 \_langw de  Kapitel      Tabelle     Abbildung    Betreff
42 \_quotationmarks de {„",'}
43 %todo
44 \_let \_sortingdatade  = \_sortingdatalatin
45 \_let \_ignoredcharsde = \_ignoredcharsgeneric
46 \_def \_compoundcharsde {ß:ss}
47 \_def \_xcompoundcharsde {} % ß is interpreted in second pass of sorting
48 %\_def \_compoundcharsde {ß:ss Ä:AE Ö:OE Ü:UE ä:ae ö:oe ü:ue} % alternative sorting
49
```

...etc. (see lang-data.opm)

There are analogical declaration for more languages here. Unfortunately, this file is far for completeness. I welcome you send me a part of declaration for your language.

If your language is missing in this file then a warning is reported during language initialization. You can create your private declaration in your macros (analogical as in the `lang-data.opm` file but without the `\_langdata` prefix). Then you will want to remove the warning about missing data. This can be done by `\nolanginput`{⟨*lang-tag*⟩} given before initialization of your language.

The whole file `lang-data.opm` is not preloaded in the format because I suppose a plenty languages here and I don't want to waste the TeX memory by these declarations. Each part of this file prefixed by `\_langdata` ⟨*lang-tag*⟩ {⟨*LongName*⟩} is read separately when `\_langinput`{⟨*lang-tag*⟩} is used. And it is used in the `\_langinit` macro (i.e. when the language is initialized), so the appropriate part of this file is read automatically on demand.

If the part of the `lang-data.opm` concerned by ⟨*lang-tag*⟩ is read already then `\_li:`⟨*lang-tag*⟩ is set to `R` and we don't read this part of the file again.

```
285  \_def\_langinput #1{%
286     \_unless \_ifcsname _li:#1\_endcsname
287        \_bgroup
288           \_edef\_tmp{\_noexpand\_langdata #1 }\_everyeof\_ea{\_tmp{}}%
289           \_long \_ea\_def \_ea\_tmp \_ea##\_ea1\_tmp{\_readlangdata{#1}}%
290           \_globaldefs=1
291           \_ea\_tmp \_input{lang-data.opm}%
292           \_ea\_glet \_csname _li:#1\_endcsname R%
293        \_egroup
294     \_fi
295  }
296  \_def\_readlangdata #1#2{%
297     \_ifx^#2^\_opwarning{Missing data for language "#1" in lang-data.opm}%
298     \_else \_wlog{Reading data for the language #2 (#1)}%
299     \_fi
300  }
301  \_def\_langdata #1 #2{\_endinput}
302  \_def\_nolanginput #1{\_ea\_glet \_csname _li:#1\_endcsname N}
303  \_public \nolanginput ;
```

Data of two preferred languages are preloaded in the format:

```
309  \_langinput{en} \_langinput{cs}
```

2.38 Other macros

Miscellaneous macros are here.

```
3  \_codedecl \uv {Miscenaleous <2022-05-04>} % preloaded in format
```

`\useOpTeX` and `\useoptex` are declared as `\relax`.

```
9  \_let \useOpTeX = \_relax   \_let \useoptex = \_relax
```

The `\lastpage` and `\totalpages` get the information from the `\_currpage`. The `\_Xpage` from `.ref` file sets the `\_currpage`.

```
16  \_def\_totalpages {\_openref\_ea\_ignoresecond\_currpage}
17  \_def\_lastpage   {\_openref\_ea\_usesecond\_currpage}
18  \_def\_currpage {{0}{?}}
19  \_public \lastpage \totalpages ;
```

We need `\uv`, `\clqq`, `\crqq`, `\flqq`, `\frqq`, `\uslang`, `\ehyph` `\chyph`, `\shyph`, for backward compatibility with 𝒞𝒮plain. Codes are set according to Unicode because we are using Czech only in Unicode when LuaTeX is used.

```
28
29  % for compatibility with csplain:
30
31  \_chardef\clqq=8222  \_chardef\crqq=8220
32  \_chardef\flqq=171    \_chardef\frqq=187
33  \_chardef\promile=8240
34
35  \_def\uv#1{\clqq#1\crqq}
36
37  \_let\uslang=\enlang  \_let\ehyph=\enlang
38  \_let\chyph=\cslang    \_let\shyph=\sklang
39  \_let\csUnicode=\csPatt \_let\czUnicode=\csPatt \_let\skUnicode=\skPatt
```

The `\letfont` was used in $\mathcal{CS}$plain instead of `\fontlet`.

```
45  \_let \letfont = \_fontlet
```

Non-breaking space in Unicode.

```
51  \let ^^a0=~
```

Old macro packages need these funny control sequences. We don't use them in new macros.

```
58  \_catcode`\@=11
59  \_let\z@=\_zo  \_let\z@skip=\_zoskip
60  \_newdimen\p@ \p@=1pt
61  \_toksdef\toks@=0
62  \_let\voidb@x=\_voidbox
63  \_chardef\@ne=1 \_chardef\tw@=2 \_chardef\thr@@=3 \_chardef\sixt@@n=16
64  \_mathchardef\@m=1000 \_mathchardef\@M=10000 \_mathchardef\@MM=20000
65  \_countdef\m@ne=22 \m@ne=-1
66  \_chardef\@cclv=255 \_mathchardef\@cclvi=256
67  \_skipdef\skip@=0
68  \_dimendef\dimen@=0  \_dimendef\dimen@i=1
69  \_dimendef\dimen@ii=2
70  \_countdef\count@=255
71  \_def\m@th{\_mathsurround\z@}
72  \_def\o@lign{\_lineskiplimit\z@ \_oalign}
73  \_def\n@space{\_nulldelimiterspace\z@ \m@th}
74  \_newdimen\p@renwd \p@renwd=8.75pt
75  \_def\alloc@#1#2#3#4#5{\_allocator#5{\_csstring#2}#3}
76  \_catcode`\@=12
```

We don't want to read `opmac.tex` unless `\input opmac` is specified.

```
82  \_def\OPmacversion{OpTeX}
```

We allow empty lines in math formulae. It is more comfortable.

```
88  \_suppressmathparerror = 1
```

Lorem ipsum can be printed by `\lipsum`[⟨*range*⟩] or `\lorem`[⟨*range*⟩], for example `\lipsum[3]` or `\lipsum[112-121]`, max=150.

First usage of `\lipsum` reads the LaTeX file `lipsum.ltd.tex` by `\_lipsumload` and prints the selected paragraph(s). Next usages of `\lipsum` prints the selected paragraph(s) from memory. This second and more usages of `\lipsum` are fully expandable. If you want to have all printings of `\lipsum` expandable, use dummy `\lipsum[0]` first.

`\lipsum` adds `\_par` after each printed paragraph. If you don't need such `\_par` here, use `\lipsumtext`[⟨*number*⟩] or `\lipsum[`⟨*number*⟩`.]` (i.e. dot after the parameter). The first case prints the paragraph ⟨*number*⟩ without the final `\_par` and the second case prints only first sentence from the paragraph ⟨*number*⟩ using `\_lipsumdot`.

```
108  \_def\_lipsumtext[#1]{\_lipsumload\_cs{_lip:#1}}
109  \_def\_lipsum[#1]{\_lipsumA #1.]{#1}}
110  \_def\_lipsumA #1.#2]#3{\_ifx^#2^\_lipsumB #1\_empty-\_empty\_end \_else \_lipsumdot[#1].\_fi}
111  \_def\_lipsumB #1-#2\_empty#3\_end{%
112      \_fornum #1..\_ifx^#2^#1\_else#2\_fi \_do {\_lipsumtext[##1]\_par}}
113  \_def\_lipsumload{{%
114      \_setbox0=\_vbox{\_tmpnum=0 % vertical mode during \input lipsum.ltd.tex
115          \_def\ProvidesFile##1[##2]{}%
116          \_def\SetLipsumLanguage##1{}%
117          \_def\NewLipsumPar{\_incr\_tmpnum \_sxdef{_lip:\_the\_tmpnum}}%
118          \_opinput {lipsum.ltd.tex}%
119          \_global\_let\_lipsumload=\_empty
120      }}}
121  \_def\_lipsumdot[#1]{\_lipsumload \_ea\_ea\_ea \_lipsumdotA \_csname _lip:#1\_endcsname.\_end}
122  \_def\_lipsumdotA #1.#2\_end {#1}
123
124  \_public \lipsum \lipsumtext ;
125  \_let \lorem=\lipsum
```

LuaTEX version 1.14 and newer provides `\partokenname` which allows to specify something different than `\par` at empty lines. We set `\_par` (see bellow) in OpTEX version 1.04+ and newer. Some macros were rewritten due to this change. And we copy old versions of these changed macros here in order to allow to use older LuaTEX versions where `\partokenname` is not provided.

Note that your macros where a parameter is separated by the empty line must be changed too. Use `\def\macro #1\_par{...}` instead `\def\macro #1\par{...}`.

```
139  \_ifx\_partokenname\_undefined % LuaTeX 1.13 or older:
140
141      \_def\_begmulti #1 {\_par\_bgroup\_wipeepar\_multiskip\_penalty0 \_def\_Ncols{#1}
142          \_setbox6=\_vbox\_bgroup\_bgroup \_let\_setxhsize=\_relax \_penalty-99
143          \_advance\_hsize by\_colsep
144          \_divide\_hsize by\_Ncols  \_advance\_hsize by-\_colsep
145          \_mullines=0
146          \_def\par{\_ifhmode\_endgraf\_global\_advance\_mullines by\_prevgraf\_fi}%
147      }
148      \_def\_incaption {\_bgroup
149          \_ifcsname _\_tmpa num\_endcsname \_ea\_incr \_csname _\_tmpa num\_endcsname
150          \_else \_opwarning{Unknown caption /\_tmpa}\_fi
151          \_edef\_thecapnum {\_csname _the\_tmpa num\_endcsname}%
152          \_edef\_thecaptitle{\_mtext{\_tmpa}}%
153          \_ea\_the \_csname _everycaption\_tmpa\_endcsname
154          \_def\_par{\_nbpar\_egroup}\_let\par=\_par
155          \_cs{_printcaption\_tmpa}%
156      }
157      \_def\_boxlines{%
158          \_def\_boxlinesE{\_ifhmode\_egroup\_empty\_fi}%
159          \_def\_nl{\_boxlinesE}%
160          \_bgroup \_lccode`\~=`\^^M\_lowercase{\_egroup\_let~}\_boxlinesE
161          \_everypar{\_setbox0=\_lastbox\_endgraf
162              \_hbox\_bgroup \_catcode`\^^M=13 \_let\par=\_nl \_aftergroup\_boxlinesC}%
163      }
164      \_def\_letter{
165          \_def\_address{\_vtop\_bgroup\_boxlines \_parskip=0pt \_let\par=\_egroup}
166          \_def\_subject{{\_bf \_mtext{subj}: }}
167          \_public \address \subject ;
168          \_typosize[11/14]
169          \_vsize=\_dimexpr \_topskip + 49\_baselineskip \_relax % added 2020-03-28
170          \_parindent=0pt
171          \_parskip=\_medskipamount
172          \_nopagenumbers
173      }
174      \_def\_printverbline#1{\_putttpenalty \_indent \_printverblinenum \_kern\_ttshift #1\par}
175      \_public \begmulti \boxlines \letter ;
176
177  \_else % LuaTeX 1.14 or newer:
```

We set `\partokenneame` to `\_par` in order to keep the name `\par` in user name space. I.e. a user can say `\def\par{paragraph}` for example without crash of processing the document. Se section 2.2 for more details about the name space concept.

Moreover, we set `\partokencontext` to one in order to the `\_par` token is inserted not only at empty lines, but also at the end of `\vbox`, `\vtop` and `\vcenter` if horizontal mode is opened here. This differs from default TEX behavior where horizontal mode is closed in these cases without inserting par token.

We set `\_partokenset` to defined value 1 in order to the macro programmer can easily check these settings in OpTEX format by `\ifx\_partokenset\undefined ... \else ...\fi`.

```
194      \_partokenname\_par
195      \_partokencontext=1
196      \_let\_partokenset=1
197  \_fi
```

2.39   Lua code embedded to the format

The file `optex.lua` is loaded into the format in `optex.ini` as byte-code and initialized by `\everyjob`, see section 2.1.

The file implements part of the functionality from `luatexbase` namespace, nowadays defined by LaTeX kernel. `luatexbase` deals with modules, allocators, and callback management. Callback management is a nice extension and is actually used in OpTeX. Other functions are defined more or less just to suit luaotfload's use.

The allocations are declared in subsection 2.39.2, calbacks are implemented in subsection 2.39.3 and handling with colors can be found in the subsection 2.39.5.

```
4
5  local fmt = string.format
6
```

2.39.1 General

Define namespace where some OpTeX functions will be added.

```
10
11 optex = optex or {}
12
```

Error function used by following functions for critical errors.

```
14 local function err(message)
15     error("\nerror: "..message.."\n")
16 end
```

For a `\chardef`'d, `\countdef`'d, etc., csname return corresponding register number. The responsibility of providing a `\XXdef`'d name is on the caller.

```
20 function registernumber(name)
21     return token.create(name).index
22 end
```

MD5 hash of given file.

```
25 function mdfive(file)
26     local fh = io.open(file, "rb")
27     if fh then
28         local data = fh:read("*a")
29         fh:close()
30         tex.print(md5.sumhexa(data))
31     end
32 end
```

2.39.2 Allocators

```
35 alloc = alloc or {}
```

An attribute allocator in Lua that cooperates with normal OpTeX allocator.

```
38 local attributes = {}
39 function alloc.new_attribute(name)
40     local cnt = tex.count["_attributealloc"] + 1
41     if cnt > 65534 then
42         tex.error("No room for a new attribute")
43     else
44         tex.setcount("global", "_attributealloc", cnt)
45         texio.write_nl("log", '"'..name..'"=\\attribute'..tostring(cnt))
46         attributes[name] = cnt
47         return cnt
48     end
49 end
```

Allocator for Lua functions ("pseudoprimitives"). It passes variadic arguments ("...") like `"global"` to `token.set_lua`.

```
53 local function_table = lua.get_functions_table()
54 function define_lua_command(csname, fn, ...)
55     local luafnalloc = #function_table + 1
56     token.set_lua(csname, luafnalloc, ...) -- WARNING: needs LuaTeX 1.08 (2019) or newer
57     function_table[luafnalloc] = fn
58 end
```

196

`provides_module` is needed by older version of luaotfload

```
61  provides_module = function() end
```

2.39.3 Callbacks

```
64  callback = callback or {}
```

Save `callback.register` function for internal use.

```
67  local callback_register = callback.register
68  function callback.register(name, fn)
69      err("direct registering of callbacks is forbidden, use 'callback.add_to_callback'")
70  end
```

Table with lists of functions for different callbacks.

```
73  local callback_functions = {}
```

Table that maps callback name to a list of descriptions of its added functions. The order corresponds with `callback_functions`.

```
76  local callback_description = {}
```

Table used to differentiate user callbacks from standard callbacks. Contains user callbacks as keys.

```
80  local user_callbacks = {}
```

Table containing default functions for callbacks, which are called if either a user created callback is defined, but doesn't have added functions or for standard callbacks that are "extended" (see `mlist_to_hlist` and its pre/post filters below).

```
85  local default_functions = {}
```

Table that maps standard (and later user) callback names to their types.

```
88   local callback_types = {
89       -- file discovery
90       find_read_file     = "exclusive",
91       find_write_file    = "exclusive",
92       find_font_file     = "data",
93       find_output_file   = "data",
94       find_format_file   = "data",
95       find_vf_file       = "data",
96       find_map_file      = "data",
97       find_enc_file      = "data",
98       find_pk_file       = "data",
99       find_data_file     = "data",
100      find_opentype_file = "data",
101      find_truetype_file = "data",
102      find_type1_file    = "data",
103      find_image_file    = "data",
104
105      open_read_file     = "exclusive",
106      read_font_file     = "exclusive",
107      read_vf_file       = "exclusive",
108      read_map_file      = "exclusive",
109      read_enc_file      = "exclusive",
110      read_pk_file       = "exclusive",
111      read_data_file     = "exclusive",
112      read_truetype_file = "exclusive",
113      read_type1_file    = "exclusive",
114      read_opentype_file = "exclusive",
115
116      -- data processing
117      process_input_buffer  = "data",
118      process_output_buffer = "data",
119      process_jobname       = "data",
120      input_level_string    = "data",
121
122      -- node list processing
```

```
123        contribute_filter       = "simple",
124        buildpage_filter        = "simple",
125        build_page_insert       = "exclusive",
126        pre_linebreak_filter    = "list",
127        linebreak_filter        = "exclusive",
128        append_to_vlist_filter  = "exclusive",
129        post_linebreak_filter   = "reverselist",
130        hpack_filter            = "list",
131        vpack_filter            = "list",
132        hpack_quality           = "list",
133        vpack_quality           = "list",
134        process_rule            = "exclusive",
135        pre_output_filter       = "list",
136        hyphenate               = "simple",
137        ligaturing              = "simple",
138        kerning                 = "simple",
139        insert_local_par        = "simple",
140        mlist_to_hlist          = "exclusive",
141
142        -- information reporting
143        pre_dump                = "simple",
144        start_run               = "simple",
145        stop_run                = "simple",
146        start_page_number       = "simple",
147        stop_page_number        = "simple",
148        show_error_hook         = "simple",
149        show_error_message      = "simple",
150        show_lua_error_hook     = "simple",
151        start_file              = "simple",
152        stop_file               = "simple",
153        call_edit               = "simple",
154        finish_synctex          = "simple",
155        wrapup_run              = "simple",
156
157        -- pdf related
158        finish_pdffile          = "data",
159        finish_pdfpage          = "data",
160        page_order_index        = "data",
161        process_pdf_image_content = "data",
162
163        -- font related
164        define_font     = "exclusive",
165        glyph_not_found = "exclusive",
166        glyph_info      = "exclusive",
167
168        -- undocumented
169        glyph_stream_provider = "exclusive",
170        provide_charproc_data = "exclusive",
171 }
```

Return a list containing descriptions of added callback functions for specific callback.

```
175 function callback.callback_descriptions(name)
176     return callback_description[name] or {}
177 end
178
179 local valid_callback_types = {
180     exclusive = true,
181     simple = true,
182     data = true,
183     list = true,
184     reverselist = true,
185 }
```

Create a user callback that can only be called manually using `call_callback`. A default function is only needed by "exclusive" callbacks.

```
189 function callback.create_callback(name, cbtype, default)
190     if callback_types[name] then
191         err("cannot create callback '"..name.."' - it already exists")
```

```
192        elseif not valid_callback_types[cbtype] then
193            err("cannot create callback '"..name.. "' with invalid callback type '"..cbtype.."'")
194        elseif ctype == "exclusive" and not default then
195            err("unable to create exclusive callback '"..name.."', default function is required")
196        end
197
198        callback_types[name] = cbtype
199        default_functions[name] = default or nil
200        user_callbacks[name] = true
201    end
```

Add a function to the list of functions executed when callback is called. For standard luatex callback a proxy function that calls our machinery is registered as the real callback function. This doesn't happen for user callbacks, that are called manually by user using `call_callback` or for standard callbacks that have default functions – like `mlist_to_hlist` (see below).

```
209  local call_callback
210  function callback.add_to_callback(name, fn, description)
211      if user_callbacks[name] or callback_functions[name] or default_functions[name] then
212          -- either:
213          --   a) user callback - no need to register anything
214          --   b) standard callback that has already been registered
215          --   c) standard callback with default function registered separately
216          --      (mlist_to_hlist)
217      elseif callback_types[name] then
218          -- This is a standard luatex callback with first function being added,
219          -- register a proxy function as a real callback. Assert, so we know
220          -- when things break, like when callbacks get redefined by future
221          -- luatex.
222          callback_register(name, function(...)
223              return call_callback(name, ...)
224          end)
225      else
226          err("cannot add to callback '"..name.."' - no such callback exists")
227      end
228
229      -- add function to callback list for this callback
230      callback_functions[name] = callback_functions[name] or {}
231      table.insert(callback_functions[name], fn)
232
233      -- add description to description list
234      callback_description[name] = callback_description[name] or {}
235      table.insert(callback_description[name], description)
236  end
```

Remove a function from the list of functions executed when callback is called. If last function in the list is removed delete the list entirely.

```
240  function callback.remove_from_callback(name, description)
241      local descriptions = callback_description[name]
242      local index
243      for i, desc in ipairs(descriptions) do
244          if desc == description then
245              index = i
246              break
247          end
248      end
249
250      table.remove(descriptions, index)
251      local fn = table.remove(callback_functions[name], index)
252
253      if #descriptions == 0 then
254          -- Delete the list entirely to allow easy checking of "truthiness".
255          callback_functions[name] = nil
256
257          if not user_callbacks[name] and not default_functions[name] then
258              -- this is a standard callback with no added functions and no
259              -- default function (i.e. not mlist_to_hlist), restore standard
260              -- behaviour by unregistering.
```

```
261                callback_register(name, nil)
262            end
263        end
264
265        return fn, description
266 end
```

helper iterator generator for iterating over reverselist callback functions

```
269 local function reverse_ipairs(t)
270     local i, n = #t + 1, 1
271     return function()
272         i = i - 1
273         if i >= n then
274             return i, t[i]
275         end
276     end
277 end
```

Call all functions added to callback. This function handles standard callbacks as well as user created callbacks. It can happen that this function is called when no functions were added to callback – like for user created callbacks or `mlist_to_hlist` (see below), these are handled either by a default function (like for `mlist_to_hlist` and those user created callbacks that set a default function) or by doing nothing for empty function list.

```
286 function callback.call_callback(name, ...)
287     local cbtype = callback_types[name]
288     -- either take added functions or the default function if there is one
289     local functions = callback_functions[name] or {default_functions[name]}
290
291     if cbtype == nil then
292         err("cannot call callback '"..name.."' - no such callback exists")
293     elseif cbtype == "exclusive" then
294         -- only one function, atleast default function is guaranteed by
295         -- create_callback
296         return functions[1](...)
297     elseif cbtype == "simple" then
298         -- call all functions one after another, no passing of data
299         for _, fn in ipairs(functions) do
300             fn(...)
301         end
302         return
303     elseif cbtype == "data" then
304         -- pass data (first argument) from one function to other, while keeping
305         -- other arguments
306         local data = (...)
307         for _, fn in ipairs(functions) do
308             data = fn(data, select(2, ...))
309         end
310         return data
311     end
312
313     -- list and reverselist are like data, but "true" keeps data (head node)
314     -- unchanged and "false" ends the chain immediately
315     local iter
316     if cbtype == "list" then
317         iter = ipairs
318     elseif cbtype == "reverselist" then
319         iter = reverse_ipairs
320     end
321
322     local head = (...)
323     local new_head
324     local changed = false
325     for _, fn in iter(functions) do
326         new_head = fn(head, select(2, ...))
327         if new_head == false then
328             return false
329         elseif new_head ~= true then
```

```
330            head = new_head
331            changed = true
332        end
333     end
334     return not changed or head
335 end
336 call_callback = callback.call_callback
```

Create "virtual" callbacks `pre/post_mlist_to_hlist_filter` by setting `mlist_to_hlist` callback. The default behaviour of `mlist_to_hlist` is kept by using a default function, but it can still be overriden by using `add_to_callback`.

```
342 default_functions["mlist_to_hlist"] = node.mlist_to_hlist
343 callback.create_callback("pre_mlist_to_hlist_filter", "list")
344 callback.create_callback("post_mlist_to_hlist_filter", "reverselist")
345 callback_register("mlist_to_hlist", function(head, ...)
346     -- pre_mlist_to_hlist_filter
347     local new_head = call_callback("pre_mlist_to_hlist_filter", head, ...)
348     if new_head == false then
349         node.flush_list(head)
350         return nil
351     elseif new_head ~= true then
352         head = new_head
353     end
354     -- mlist_to_hlist means either added functions or standard luatex behavior
355     -- of node.mlist_to_hlist (handled by default function)
356     head = call_callback("mlist_to_hlist", head, ...)
357     -- post_mlist_to_hlist_filter
358     new_head = call_callback("post_mlist_to_hlist_filter", head, ...)
359     if new_head == false then
360         node.flush_list(head)
361         return nil
362     elseif new_head ~= true then
363         head = new_head
364     end
365     return head
366 end)
```

For preprocessing boxes just before shipout we define custom callback. This is used for coloring based on attributes. There is however a challenge - how to call this callback? We could redefine `\shipout` and `\pdfxform` (which both run `ship_out` procedure internally), but they would lose their primtive meaning – i.e. `\immediate` wouldn't work with `\pdfxform`. The compromise is to require anyone to run `\_preshipout`⟨*destination box number*⟩⟨*box specification*⟩ just before `\shipout` or `\pdfxform` if they want to call `pre_shipout_filter` (and achieve colors and possibly more).

```
377 callback.create_callback("pre_shipout_filter", "list")
378
379 local tex_setbox = tex.setbox
380 local token_scanint = token.scan_int
381 local token_scanlist = token.scan_list
382 define_lua_command("_preshipout", function()
383     local boxnum = token_scanint()
384     local head = token_scanlist()
385     head = call_callback("pre_shipout_filter", head)
386     tex_setbox(boxnum, head)
387 end)
```

Compatibility with LaTeX through luatexbase namespace. Needed for luaotfload.

```
391 luatexbase = {
392     registernumber = registernumber,
393     attributes = attributes,
394     provides_module = provides_module,
395     new_attribute = alloc.new_attribute,
396     callback_descriptions = callback.callback_descriptions,
397     create_callback = callback.create_callback,
398     add_to_callback = callback.add_to_callback,
399     remove_from_callback = callback.remove_from_callback,
400     call_callback = callback.call_callback,
```

```
401        callbacktypes = {}
402 }
```

\tracingmacros callback registered. Use \tracingmacros=3 or \tracingmacros=4 if you want to see the result.

```
406 callback.add_to_callback("input_level_string", function(n)
407      if tex.tracingmacros > 3 then
408          return "[" .. n .. "] "
409      elseif tex.tracingmacros > 2 then
410          return "~" .. string.rep(".",n)
411      else
412          return ""
413      end
414 end, "_tracingmacros")
```

2.39.4 Management of PDF page resources

Traditionally, pdfTeX allowed managing PDF page resources (graphics states, patterns, shadings, etc.) using a single toks register, \pdfpageresources. This is insufficient due to the expected PDF object structer and also because many "packages" want to add page resources and thus fight for the access to that register. We add a finer alternative, which allows adding different kinds of resources to a global page resources dictionary. Note that some resource types (fonts and XObjects) are already managed by LuaTeX and shouldn't be added!

XObject forms can also use resources, but there are several ways to make LuaTeX reference resources from forms. It is hence left up to the user to insert page resources managed by us, if they need them. For that, use pdf.get_page_resources(), or the below TeX alternative for that.

```
431 local pdfdict_mt = {
432      __tostring = function(dict)
433          local out = {"<<"}
434          for k, v in pairs(dict) do
435              out[#out+1] = fmt("/%s %s", tostring(k), tostring(v))
436          end
437          out[#out+1] = ">>"
438          return table.concat(out, "\n")
439      end,
440 }
441 local function pdf_dict(t)
442      return setmetatable(t or {}, pdfdict_mt)
443 end
```

```
445 local resource_dict_objects = {}
446 local page_resources = {}
447 function pdf.add_page_resource(type, name, value)
448      local resources = page_resources[type]
449      if not resources then
450          local obj = pdf.reserveobj()
451          pdf.setpageresources(fmt("%s /%s %d 0 R", pdf.get_page_resources(), type, obj))
452          resource_dict_objects[type] = obj
453          resources = pdf_dict()
454          page_resources[type] = resources
455      end
456      page_resources[type][name] = value
457 end
458 function pdf.get_page_resources()
459      return pdf.getpageresources() or ""
460 end
```

New "pseudo" primitives are introduced. _addpageresource{⟨type⟩}{⟨PDF name⟩}{⟨PDF dict⟩} adds more reources of given resource ⟨type⟩ to our data structure. _pageresources expands to the saved ⟨type⟩s and object numbers.

```
466 define_lua_command("_addpageresource", function()
467      pdf.add_page_resource(token.scan_string(), token.scan_string(), token.scan_string())
468 end)
469 define_lua_command("_pageresources", function()
470      tex.print(pdf.get_page_resources())
471 end)
```

We write the objects with resources to the PDF file in the `finish_pdffile` callback.

```
475 callback.add_to_callback("finish_pdffile", function()
476     for type, dict in pairs(page_resources) do
477         local obj = resource_dict_objects[type]
478         pdf.immediateobj(obj, tostring(dict))
479     end
480 end)
```

2.39.5 Handling of colors and transparency using attributes

Because LuaTeX doesn't do anything with attributes, we have to add meaning to them. We do this by intercepting TeX just before it ships out a page and inject PDF literals according to attributes.

```
488 local node_id = node.id
489 local node_subtype = node.subtype
490 local glyph_id = node_id("glyph")
491 local rule_id = node_id("rule")
492 local glue_id = node_id("glue")
493 local hlist_id = node_id("hlist")
494 local vlist_id = node_id("vlist")
495 local disc_id = node_id("disc")
496 local whatsit_id = node_id("whatsit")
497 local pdfliteral_id = node_subtype("pdf_literal")
498 local pdfsave_id = node_subtype("pdf_save")
499 local pdfrestore_id = node_subtype("pdf_restore")
500 local token_getmacro = token.get_macro
501
502 local direct = node.direct
503 local todirect = direct.todirect
504 local tonode = direct.tonode
505 local getfield = direct.getfield
506 local setfield = direct.setfield
507 local getwhd = direct.getwhd
508 local getid = direct.getid
509 local getlist = direct.getlist
510 local setlist = direct.setlist
511 local getleader = direct.getleader
512 local getattribute = direct.get_attribute
513 local insertbefore = direct.insert_before
514 local copy = direct.copy
515 local traverse = direct.traverse
516 local one_bp = tex.sp("1bp")
```

The attribute for coloring is allocated in `colors.opm`

```
519 local color_attribute = registernumber("_colorattr")
520 local transp_attribute = registernumber("_transpattr")
```

Now we define function which creates whatsit nodes with PDF literals. We do this by creating a base literal, which we then copy and customize.

```
525 local pdf_base_literal = direct.new("whatsit", "pdf_literal")
526 setfield(pdf_base_literal, "mode", 2) -- direct mode
527 local function pdfliteral(str)
528     local literal = copy(pdf_base_literal)
529     setfield(literal, "data", str)
530     return literal
531 end
532 optex.directpdfliteral = pdfliteral
```

The function `colorize(head, current, current_stroke, current_tr)` goes through a node list and injects PDF literals according to attributes. Its arguments are the head of the list to be colored and the current color for fills and strokes and the current trasparency attribute. It is a recursive function – nested horizontal and vertical lists are handled in the same way. Only the attributes of "content" nodes (glyphs, rules, etc.) matter. Users drawing with PDF literals have to set color themselves.

Whatsit node with color setting PDF literal is injected only when a different color or transparency is needed. Our injection does not care about boxing levels, but this isn't a problem, since PDF literal whatsits just instruct the `\shipout` related procedures to emit the literal.

We also set the stroke and non-stroke colors separately. This is because stroke color is not always needed – LuaTeX itself only uses it for rules whose one dimension is less than or equal to 1 bp and for fonts whose `mode` is set to 1 (outline) or 2 (outline and fill). Catching these cases is a little bit involved. For example rules are problematic, because at this point their dimensions can still be running (-2^{30}) – they may or may not be below the one big point limit. Also the text direction is involved. Because of the negative value for running dimensions the simplistic check, while not fully correct, should produce the right results. We currently don't check for the font mode at all.

Leaders (represented by glue nodes with leader field) are not handled fully. They are problematic, because their content is repeated more times and it would have to be ensured that the coloring would be right even for e.g. leaders that start and end on a different color. We came to conclusion that this is not worth, hence leaders are handled just opaquely and only the attribute of the glue node itself is checked. For setting different colors inside leaders, raw PDF literals have to be used.

We use the `node.direct` way of working with nodes. This is less safe, and certainly not idiomatic Lua, but faster and codewise more close to the way TeX works with nodes.

```
570  local function is_color_needed(head, n, id, subtype) -- returns fill, stroke color needed
571      if id == glyph_id then
572          return true, false
573      elseif id == glue_id then
574          n = getleader(n)
575          if n then
576              return true, true
577          end
578      elseif id == rule_id then
579          local width, height, depth = getwhd(n)
580          if width <= one_bp or height + depth <= one_bp then
581              -- running (-2^30) may need both
582              return true, true
583          end
584          return true, false
585      elseif id == whatsit_id and (subtype == pdfliteral_id
586              or subtype == pdfsave_id
587              or subtype == pdfrestore_id) then
588          return true, true
589      end
590      return false, false
591  end
592
593  local function colorize(head, current, current_stroke, current_tr)
594      for n, id, subtype in traverse(head) do
595          if id == hlist_id or id == vlist_id then
596              -- nested list, just recurse
597              local list = getlist(n)
598              list, current, current_stroke, current_tr =
599                  colorize(list, current, current_stroke, current_tr)
600              setlist(n, list)
601          elseif id == disc_id then
602              -- at this point only no-break (replace) list is of any interest
603              local replace = getfield(n, "replace")
604              if replace then
605                  replace, current, current_stroke, current_tr =
606                      colorize(replace, current, current_stroke, current_tr)
607                  setfield(n, "replace", replace)
608              end
609          else
610              local fill_needed, stroke_needed = is_color_needed(head, n, id, subtype)
611              local new = getattribute(n, color_attribute) or 0
612              local newtr = getattribute(n, transp_attribute) or 0
613              local newliteral = nil
614              if current ~= new and fill_needed then
615                  newliteral = token_getmacro("_color:"..new)
616                  current = new
617              end
618              if current_stroke ~= new and stroke_needed then
619                  local stroke_color = token_getmacro("_color-s:"..current)
620                  if stroke_color then
621                      if newliteral then
```

```
622                    newliteral = fmt("%s %s", newliteral, stroke_color)
623                else
624                    newliteral = stroke_color
625                end
626                current_stroke = new
627            end
628        end
629        if newtr ~= current_tr and fill_needed then -- (fill_ or stroke_needed) = fill_neded
630            if newliteral ~= nil then
631                newliteral = fmt("%s /tr%d gs", newliteral, newtr)
632            else
633                newliteral = fmt("/tr%d gs", newtr)
634            end
635            current_tr = newtr
636        end
637        if newliteral then
638            head = insertbefore(head, n, pdfliteral(newliteral))
639        end
640        end
641    end
642    return head, current, current_stroke, current_tr
643 end
```

Colorization should be run just before shipout. We use our custom callback for this. See the definition of `pre_shipout_filter` for details on limitations.

```
648 callback.add_to_callback("pre_shipout_filter", function(list)
649     -- By setting initial color to -1 we force initial setting of color on
650     -- every page. This is useful for transparently supporting other default
651     -- colors than black (although it has a price for each normal document).
652     local list = colorize(todirect(list), -1, -1, 0)
653     return tonode(list)
654 end, "_colors")
```

We also hook into `luaotfload`'s handling of color. Instead of the default behavior (inserting colorstack whatsits) we set our own attribute. The hook has to be registered *after* `luaotfload` is loaded.

```
659 function optex_hook_into_luaotfload()
660     if not luaotfload.set_colorhandler then
661         return -- old luaotfload, colored fonts will be broken
662     end
663     local setattribute = direct.set_attribute
664     local token_setmacro = token.set_macro
665     local color_count = registernumber("_colorcnt")
666     local tex_getcount, tex_setcount = tex.getcount, tex.setcount
667     luaotfload.set_colorhandler(function(head, n, rgbcolor) -- rgbcolor = "1 0 0 rg"
668         local attr = tonumber(token_getmacro("_color::"..rgbcolor))
669         if not attr then
670             attr = tex_getcount(color_count)
671             tex_setcount(color_count, attr + 1)
672             local strattr = tostring(attr)
673             token_setmacro("_color::"..rgbcolor, strattr)
674             token_setmacro("_color:"..strattr, rgbcolor)
675             -- no stroke color set
676         end
677         setattribute(n, color_attribute, attr)
678         return head, n
679     end)
680 end
681
682     -- History:
683     -- 2022-03-07 transparency in the colorize() function, current_tr added
684     -- 2022-03-05 resources management added
685     -- 2021-07-16 support for colors via attributes added
686     -- 2020-11-11 optex.lua released
```

2.40 Printing documentation

The `\printdoc` ⟨*filename*⟩⟨*space*⟩ and `\printdoctail` ⟨*filename*⟩⟨*space*⟩ commands are defined after the file `doc.opm` is load by `\load` [doc].

The `\printdoc` starts reading of given ⟨*filename*⟩ from the second line. The file is read in *the listing mode*. The `\prindoctail` starts reading given ⟨*filename*⟩ from the first occurrence of the `\_endcode`. The file is read in normal mode (like `\input` ⟨*filename*⟩).

The *listing mode* prints the lines as a listing of a code. This mode is finished when first ␣␣`\_doc` occurs or first `\_endcode` occurs. At least two spaces or one tab character must precede before such `\_doc`. On the other hand, the `\_endcode` must be at the left edge of the line without spaces. If this rule is not met then the listing mode continues.

If the first line or the last line of the listing mode is empty then such lines are not printed. The maximal number of printed lines in the listing mode is `\maxlines`. It is set to almost infinity (100000). You can set it to a more sensible value. Such a setting is valid only for the first following listing mode.

When the listing mode is finished by `\_doc` then the next lines are read in the normal way, but the material between `\begtt` ... `\endtt` pair is shifted by three letters left. The reason is that the three spaces of indentation is recommended in the `\_doc` ... `\_cod` pair and this shifting is compensation for this indentation.

The `\_cod` macro ignores the rest of the current line and starts the listing mode again.

When the listing mode is finished by the `\_endcode` then the `\endinput` is applied, the reading of the file opened by `\printdoc` is finished.

You cannot reach the end of the file (without `\_endcode`) in the listing mode.

The main documentation point is denoted by `\`\`\`⟨*sequence*⟩`` ` in red, for example `\`\`\foo`` `. The user documentation point is the first occurrence of `\^`\`⟨*sequence*⟩`` `, for example `\^`\`\foo`` `. There can be more such markups, all of them are hyperlinks to the main documentation point. And main documentation point is a hyperlink to the user documentation point if this point precedes. Finally, the `\~`\`⟨*sequence*⟩`` ` (for example `\~`\`\foo`` `) are hyperlinks to the user documentation point.

By default, the hyperink from main documentation point to the user documentation point is active only if it is backward link, i.e. the main documentation point is given later. The reason is that we don't know if such user documentation point will exist when creating main documentation point and we don't want broken links. If you are sure that user documentation point will follow then use prefix `\fw` before `\` `, for example `\fw\`\`\foo`` ` is main documentation point where the user documentation point is given later and forward hyperlink is created here.

Control sequences and their page positions of main documentation points and user documentation points are saved to the index.

The listing mode creates all control sequences which are listed in the index as an active link to the main documentation point of such control sequence and prints them in blue. Moreower, active links are control sequneces of the type `\_foo` or `\.foo` although the documentation mentions only `\foo`. Another text is printed in black.

The listing mode is able to generate external links to another OpTEX-like documentation, if the macros `\,`⟨*csname*⟩ and `\el:`⟨*csname*⟩ are defined. The second macro should create a hyperlink using `\_tmpa` where the link name of the ⟨*csname*⟩ is saved and `\_tmpb` where the name of the ⟨*csname*⟩ to be printed is saved (`\tmpb` can include preceding _ or . unlike `\_tmpa`). For example, suppose that we have created `optex-doc.eref` file by:

```
TEXINPUTS='.;$TEXMF/{doc,tex}//' optex optex-doc
grep Xindex optex-doc.ref > optex-doc.eref
```

The `.eref` file includes only `\_Xindex{`⟨*csname*⟩`}{}` lines from `optex-doc.ref` file. Then we can use following macros:

```
\def\_Xindex#1#2{\sdef{,#1}{}\slet{el:#1}{optexdoclink}}
\def\optexdoclink{%
   \edef\extlink{url:\optexdocurl\csstring\#cs:\_tmpa}%
   \_ea\_urlactive\_ea[\extlink]{\Cyan}{\csstring\\\_tmpb}}
\def\optexdocurl{http://petr.olsak.net/ftp/olsak/optex/optex-doc.pdf}
\isfile{optex-doc.eref}\iftrue \input{optex-doc.eref}\fi
```

All `\el:`⟨*csname*⟩, where ⟨*csname*⟩ is from `optex-doc.ref`, have the same meaning: `\optexdoclink` in this example. And `\optexdoclink` creates the external link in `\Cyan` color.

2.40.1 Implementation

```
3  \_codedecl \printdoc {Macros for documentation printing <2022-07-01>} % loaded on demand by \load[doc]
```

General decalarations.

```
9  \_fontfam[lmfonts]

10

11 \_let \mlinkcolor=\Red    % main doc. points
12 \_let \ulinkcolor=\Blue   % user doc. points
13 \_let \fnamecolor=\Brown  % file names in listing headers
14 \_def \bgverbcolor {\_setcmykcolor{0 0 .3 .03}} % background for listings
15 \_hyperlinks \Green \Green
16 \_enlang
17 \_enquotes
```

Maybe, somebody needs \seccc or \secccc?

```
23 \_eoldef\seccc#1{\_medskip \_noindent{\_bf#1}\_par\_nobreak\_firstnoindent}
24 \_def\secccc{\_medskip\_noindent $\_bullet$ }
```

\enddocument can be redefined.

```
30 \_let\enddocument=\_bye
```

A full page of listing causes underfull \vbox in output routine. We need to add a small tolerance.

```
37 \_pgbottomskip=0pt plus10pt minus2pt
```

The listing mode is implemented here. The \maxlines is maximal lines of code printed in the listing mode. The _catcodedot sets dot as letter in listngs (for package documentation where \.foo sequeces exist).

```
46 \_newcount \_maxlines    \_maxlines=100000
47 \_public \maxlines ;

48

49 \_eoldef\_cod#1{\_par \_wipeepar
50    \_vskip\_parskip \_medskip \_ttskip
51    \_begingroup
52    \_typosize[8/10]
53    \_let\_printverbline=\_printcodeline
54    \_ttline=\_inputlineno
55    \_setverb \_catcodedot
56    \_ifnum\_ttline<0 \_let\_printverblinenum=\_relax \_else \_initverblinenum \_fi
57    \_adef{ }{\ }\_adef\^^I{\t}\_parindent=\_ttindent \_parskip=0pt
58    \_def\t{\_hskip \_dimexpr\_tabspaces em/2\_relax}%
59    \_relax \_ttfont
60    \_endlinechar=`^^J
61    \_def\_tmpb{\_start}%
62    \_readverbline
63 }
64 \_def\_readverbline #1^^J{%
65    \_def\_tmpa{\_empty#1}%
66    \_let\_next=\_readverbline
67    \_ea\_isinlist\_ea\_tmpa\_ea{\_Doc}\_iftrue \_let\_next=\_processinput \_fi
68    \_ea\_isinlist\_ea\_tmpa\_ea{\_Doctab}\_iftrue \_let\_next=\_processinput \_fi
69    \_ea\_isinlist\_ea\_tmpa\_ea{\_Endcode}\_iftrue \_def\_next{\_processinput\_endinput}\_fi
70    \_ifx\_next\_readverbline \_addto\_tmpb{#1^^J}\_fi
71    \_next
72 }
73 {\_catcode`\ =13 \_gdef\_aspace{ }}\_def\_asp{\_ea\_noexpand\_aspace}
74 \_edef\_Doc{\_asp\_asp\_bslash _doc}
75 \_bgroup \_lccode`~=`\^^I \_lowercase{\_egroup\_edef\_Doctab{\_noexpand~\_bslash _doc}}
76 \_edef\_Endcode{\_noexpand\_empty\_bslash _endcode}
77 \_def\_catcodedot{\_catcode`\.=11 }
```

The scanner of the control sequences in the listing mode replaces all occurrences of \ by _makecs. This macro reads next tokens and accumulates them to _tmpa as long as they have category 11. It means that _tmpa includes the name of the following control sequence when _makecsF is run. The printing

form of the control sequence is set to `\_tmpb` and the test of existence `\,⟨csname⟩`is performed. If it is true then active hyperlink is created. If not, then the first `_` or `.` is removed from `\_tmpa` and the test is repeated.

```
90  \_def\_makecs{\_def\_tmpa{}\_futurelet\_next\_makecsA}
91  \_def\_makecsA{\_ifcat a\_noexpand\_next \_ea\_makecsB \_else \_ea\_makecsF \_fi}
92  \_def\_makecsB#1{\_addto\_tmpa{#1}\_futurelet\_next\_makecsA}
93  \_def\_makecsF{\_let\_tmpb=\_tmpa
94          \_ifx\_tmpa\_empty \_csstring\\%
95          \_else \_ifcsname ,\_tmpa\_endcsname \_trycs{el:\_tmpa}{\_intlink}%
96          \_else \_remfirstunderscoreordot\_tmpa
97                  \_ifx\_tmpa\_empty \_let\_tmpa=\_tmpb \_fi
98                  \_ifcsname ,\_tmpa\_endcsname \_trycs{el:\_tmpa}{\_intlink}%
99          \_else \_csstring\\\_tmpb \_fi\_fi\_fi
100 }
101 \_def\_processinput{%
102     \_let\_start=\_relax
103     \_ea\_replstring\_ea\_tmpb\_ea{\_aspace^^J}{^^J}
104     \_addto\_tmpb{\_end}%
105     \_isinlist\_tmpb{\_start^^J}\_iftrue \_advance\_ttline by1\_fi
106     \_replstring\_tmpb{\_start^^J}{\_start}%
107     \_replstring\_tmpb{\_start}{}%
108     \_replstring\_tmpb{^^J\_end}{\_end}%
109     \_replstring\_tmpb{^^J\_end}{}%
110     \_replstring\_tmpb{\_end}{}%
111     \_ea\_prepareverbdata\_ea\_tmpb\_ea{\_tmpb^^J}%
112     \_replthis{\_csstring\\}{\_noexpand\_makecs}%
113     \_ea\_printverb \_tmpb\_end
114     \_par
115     \_endgroup \_ttskip
116     \_isnextchar\_par{}{\_noindent}%
117 }
118 \_def\_remfirstunderscoreordot#1{\_ea\_remfirstuordotA#1\_relax#1}
119 \_def\_remfirstuordotA#1#2\_relax#3{\_if _#1\_def#3{#2}\_fi \_if\_string#1.\_def#3{#2}\_fi}
```

By default the internal link is created by `\_intlink` inside listing mode. But you can define `\el:⟨csname⟩` which has precedence and it can create an external link. The `\_tmpa` includes the name used in the link and `\_tmpb` is the name to be printed. See `\_makecsF` above and the example at the beginning of this section.

```
129 \_def\_intlink{\_link[cs:\_tmpa]{\ulinkcolor}{\_csstring\\\_tmpb}}
```

The lines in the listing mode have a yellow background.

```
135 \_def\_printcodeline#1{\_advance \_maxlines by-1
136     \_ifnum \_maxlines<0 \_ea \_endverbprinting \_fi
137     \_ifx\_printfilename\_relax \_penalty \_ttpenalty \_fi \_vskip-4pt
138     \_noindent\_rlap{\bgverbcolor \_vrule height8pt depth5pt width\_hsize}%
139     \_printfilename
140     \_indent \_printverblinenum #1\_par}
141
142 \_def\_printfilename{\_hbox to0pt{%
143     \_hskip\_hsize\_vbox to0pt{\_vss\_llap{\fnamecolor\docfile}\_kern7.5pt}\_hss}%
144     \_let\_printfilename=\_relax
145 }
146 \_everytt={\_let\_printverblinenum=\_relax}
147
148 \_long\_def\_endverbprinting#1\_end#2\_end{\_fi\_fi \_global\_maxlines=100000
149     \_noindent\_typosize[8/]\_dots etc. (see {\_tt\fnamecolor\docfile})}
```

`\docfile` is currently documented file.
`\printdoc` and `\printdoctail` macros are defined here.

```
156 \_def\docfile{}
157 \_def\_printdoc #1 {\_par \_def\docfile{#1}%
158     \_everytt={\_ttshift=-15pt \_let\_printverblinenum=\_relax}%
159     \_ea\_cod \_input #1
160     \_everytt={\_let\_printverblinenum=\_relax}%
161     \_def\docfile{}%
```

```
162 }
163 \_def\_printdoctail #1 {\_bgroup
164     \_everytt={}\_ttline=-1 \_ea\_printdoctailA \_input #1 \_egroup}
165 {\_long\_gdef\_printdoctailA#1\_endcode{}}
166
167 \_public \printdoc \printdoctail ;
```

You can do `\verbinput` `\vitt{`⟨*filename*⟩`}` (⟨*from*⟩-⟨*to*⟩) ⟨*filename*⟩ if you need analogical design like in listing mode.

```
174 \_def\_vitt#1{\_def\docfile{#1}\_ttline=-1
175     \_everytt={\_typosize[8/10]\_let\_printverbline=\_printcodeline \_medskip}}
176
177 \_public \vitt ;
```

The Index entries are without the trailing backslash in `.ref` file. When printing Index, we distinguish the Index entries with their main documentation point (they are created as links and backslash is added), Index entries with only user documentation points have backslash added but no link is created. Other index entries are printed as usuall without backslash.

```
188 \_addto \_ignoredcharsen {_}  % foo, _foo is the same in the fist pass of sorting
189 \_let\_optexprintii=\_printii % original \_printii used for other Index entries
190 \_def\_printii #1&{%
191     \_ifcsname cs:#1\_endcsname
192         \_noindent \_hskip-\_iindent {\_tt \_link[cs:#1]\ulinkcolor{\_bslash#1} }\_else
193         \_ifcsname cs:^#1\_endcsname \_noindent \_hskip-\_iindent {\_tt\_bslash#1 }\_else
194             \_afterfi{\_afterfi{\_optexprintii #1&}}\_fi\_fi
195 }
196 \_def\_pgprintA #1{#1} % no hyperlinks from page numbers
197
198 \_def\_printiipages#1&{\_let\_pgtype=\_undefined \_tmpnum=0
199     {\_rm\_printpages #1,:,\_par}}
200
201 \_sdef{_tocl:1}#1#2#3{\_nofirst\_bigskip
202     \_bf\_llaptoclink{#1}{#2}\_hfill \_pgn{#3}\_tocpar\_medskip}
```

If this macro is loaded by `\load` then we need to initialize catcodes using the `\_afteroad` macro.

```
209 \_def\_afterload{\_catcode`\<=13 \_catcode`\`=13 \_catcode`\.=11
210     \_wlog {doc.opm: catcodes of < and ` activated, catcode of . is letter.}%
211 }
212 \_catcode`.=11
```

The `<something>` will be print as ⟨*something*⟩.

```
218 \_let\lt=<
219 \_catcode`\<=13
220
221 \_def<#1>{$\langle\hbox{\it#1\/}\rangle$}
222 \_everyintt{\_catcode`\<=13 }
```

Main documentation points and hyperlinks to/from it. Main documentation point: `\`\foo``. User documentation point: `\^`\foo`, first occurence only. The next occurrences are only links to the main documentation point. Link to user documentation point: `\~`\foo`.

```
232 \_verbchar`
233
234 \_def\`#1`{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
235     \_ifcsname cs:\_tmp\_endcsname\_else \_dest[cs:\_tmp]\_fi
236     \_sxdef{cs:\_tmp}{}%
237     \_hbox{\_ifcsname cs:^\_tmp\_endcsname
238             \_link[cs:^\_tmp]{\mlinkcolor}{\_tt\_csstring\\\_tmp}\_else
239             {\_tt\mlinkcolor\_csstring\\\_tmp}\_fi}%
240 }
241 \_def\^`#1`{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
242     \_hbox{\_ifcsname cs:^\_tmp\_endcsname \_else \_dest[cs:^\_tmp]\_sxdef{cs:^\_tmp}{}\_fi
243         \_link[cs:\_tmp]{\ulinkcolor}{\_tt\_string#1}}%
244     \_futurelet\_next\_cslinkA
245 }
```

```
246 \_def\_cslinkA{\_ifx\_next`\_ea\_ignoreit \_else \_ea\_ea\_ea`\_ea\_string\_fi}

247

248 \_def\~`#1{\_leavevmode\_edef\_tmp{\_csstring#1}\_iindex{\_tmp}%
249     \_hbox{\_link[cs:^\_tmp]{\ulinkcolor}{\_tt\_string#1}}%
250     \_futurelet\_next\_cslinkA
251 }
```

The **\fw** macro for forward links to user documentation point (given later) is defined here.

```
258 \_def\_fw\`#1`{{\_slet{cs:^\_csstring#1}{}\`#1`}}
259 \_public \fw ;
```

Index

There are all control sequences used in OpTeX except TeX primitives. If you want to know something about TeX primitives then you can use another index from TeX in a Nutshell.

212

213

215

216