OpTEX

Format Based on Plain \TeX{} and OPmac\footnote{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 \url{http://petr.olsak.net/opmac-e.html} for more information about it.}

Petr Olšák, 2020

\url{http://petr.olsak.net/optex}

OpTEX is \LaTeX{} format with Plain \TeX{} and OPmac. Only Lua\TeX{} 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:

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

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

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

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

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

1 User documentation ........................................ 5
  1.1 Starting with OpTEX .................................... 5
  1.2 Page layout ............................................. 5
    1.2.1 Setting the margins .............................. 5
    1.2.2 Concept of default page ......................... 6
    1.2.3 Footnotes and marginal notes ................... 7
  1.3 Fonts .................................................. 7
    1.3.1 Font families .................................... 7
    1.3.2 Font sizes ....................................... 8
    1.3.3 Typesetting math .................................. 9
  1.4 Typical elements of document ........................... 10
    1.4.1 Chapters and sections ......................... 10
    1.4.2 Another numbered objects ...................... 10
    1.4.3 References .................................... 12
    1.4.4 Hyperlinks, outlines ............................ 12
    1.4.5 Lists ........................................... 13
    1.4.6 Tables .......................................... 14
    1.4.7 Verbatim ....................................... 16
  1.5 Autogenerated lists .................................... 18
    1.5.1 Table of contents ................................ 18
    1.5.2 Making the index ................................ 18
    1.5.3 Bib\TeXing ...................................... 20
  1.6 Graphics .............................................. 21
    1.6.1 Colors .......................................... 21
    1.6.2 Images ......................................... 21
    1.6.3 PDF transformations ............................. 22
    1.6.4 Ovals, circles .................................. 23
    1.6.5 Putting images and texts wherever ............. 23
  1.7 Others ................................................ 23
    1.7.1 Using more languages ............................ 23
    1.7.2 Pre-defined styles .............................. 24
    1.7.3 Loading other macro packages ................... 25
    1.7.4 Lorem ipsum dolor sit .......................... 25
    1.7.5 Logos .......................................... 25
    1.7.6 The last page ................................... 25
    1.7.7 Use Op\TeX ...................................... 26
  1.8 Summary .............................................. 26
  1.9 Compatibility with Plain \TeX ......................... 27

2 Technical documentation .................................... 28
  2.1 The main initialization file .......................... 28
  2.2 Concept of name spaces of control sequences .......... 30
    2.2.1 Prefixing internal control sequences ........... 30
    2.2.2 Name space of control sequences for users .... 30
    2.2.3 Macro files syntax ................................ 31
    2.2.4 Name spaces for package writers ................ 31
    2.2.5 Summary about rules for external macro files published for Op\TeX .................. 31
    2.2.6 The implementation of the name spaces .......... 32
  2.3 pdf\TeX initialization ................................ 33
2.23 Hyperlinks ......................................................... 108
2.24 Making table of contents ........................................... 110
2.25 PDF outlines ....................................................... 112
   2.25.1 Nesting PDF outlines ...................................... 112
   2.25.2 Strings in PDF outlines .................................... 113
2.26 Chapters, sections, subsections ................................... 114
2.27 Lists, items ....................................................... 119
2.28 Verbatim, listings ................................................ 120
   2.28.1 Inline and “display” verbatim ............................... 120
   2.28.2 Listings with syntax highlighting .......................... 125
2.29 Graphics .......................................................... 128
2.30 The \table macro, tables and rules ............................... 134
   2.30.1 The boundary declarator : ................................. 134
   2.30.2 Usage of the \tabskip primitive ........................... 134
   2.30.3 Tables to given width ....................................... 134
   2.30.4 \eqbox: boxes with equal width across the whole document .... 135
   2.30.5 Implementation of the \table macro and friends .......... 135
2.31 Balanced multi-columns .......................................... 139
2.32 Citations, bibliography .......................................... 141
   2.32.1 Macros for citations and bibliography preloaded in the format .... 141
   2.32.2 The \usebib command ....................................... 144
   2.32.3 Notes for bib style writers ................................ 145
   2.32.4 The usebib.opm macro file loaded when \usebib is used .... 146
   2.32.5 Usage of the bib-iso690 style ............................. 149
   2.32.6 Implementation of the bib-iso690 style .................... 155
2.33 Sorting and making Index ...................................... 159
2.34 Footnotes and marginal notes .................................... 165
2.35 Styles ............................................................ 167
   2.35.1 \report and \letter styles ................................. 167
   2.35.2 \slides style for presentations .......................... 168
2.36 Logos ............................................................ 171
2.37 Multilingual support ............................................ 172
   2.37.1 Lowercase, uppercase codes ................................. 172
   2.37.2 Hyphenations ................................................. 172
   2.37.3 Multilingual phrases and quotation marks .................... 175
2.38 Other macros ..................................................... 178
2.39 Lua code embedded to the format ................................ 179
2.40 Printing documentation .......................................... 184

Index 188
Chapter 1
User documentation

1.1 Starting with OpTEX

OpTEX is compiled as a format for LuaTEX. Maybe there is a command \texttt{optex} in your \TeX distribution. Then you can write into command line

\texttt{optex document}

You can try to process \texttt{optex op-demo} or \texttt{optex optex-doc}.

If there is no \texttt{optex} command, see more information about installation OpTEX at \url{http://petr.olsak.net/optex}.

A minimal document should be

\texttt{\fontfam[LMfonts]Hello World! \bye}

The first line \texttt{\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\textsuperscript{1}. So the sentence \texttt{Hello World} will be OK without the first line, but you cannot print such sentence in another languages (for example \texttt{Ahoj světe!}) where Unicode fonts are needed because of the characters like \v{e} are not mapped correctly in preloaded fonts.

A somewhat larger example with common settings should be:

\texttt{\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)
\bye}

Tady je zkušební textík v českém jazyce.

You can look at \texttt{op-demo.tex} file for more complex, but still simple example.

1.2 Page layout

1.2.1 Setting the margins

The \texttt{\margins} command declares margins of the document. This command have the following parameters:

\texttt{\margins/⟨pg⟩ ⟨fmt⟩ ⟨⟨left⟩,⟨right⟩,⟨top⟩,⟨bot⟩⟩⟨unit⟩}

example:

\texttt{\margins/1 a4 (2.5,2.5,2,2)cm}

Parameters are:

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

\textsuperscript{1} This is a technical limitation of Lua\TeX\ 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 formats dimensions stays unscaled. Example:

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

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

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

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

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

\headline = {}% \footline = \{\hss\rmfixed \_folio \_hss\} % \folio expands to page number
\headlinedist = 14pt % from baseline of \headline to top of page body
\footlinedist = 24pt % from last line in pagebody to baseline of footline
The page body should be divided to top insertions (floating tables and figures) followed by a real text and followed by footnotes. Typically, only real text is here.

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

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

1.2.3 Footnotes and marginal notes

The Plain \TeX’s macro \texttt{\textbackslash footnote} can be used as usual. But a new macro \texttt{\textbackslash fnote\{\texttt{text}\}} is defined. The footnote mark is added automatically and it is numbered on each chapter from one\textsuperscript{2}. The \texttt{\textlangle text\textrangle} is scaled to 80 %. User can redefine footnote mark or scaling, as shown in the section 2.34.

The \texttt{\textbackslash fnote} macro is fully applicable only in “normal outer” paragraph. It doesn’t work inside boxes (tables, for example). If you are solving such case then you can use the command \texttt{\textbackslash fnotemark\{\texttt{numeric-label}\}} inside the box: only the footnote mark is generated here. When the box is finished you can use \texttt{\textbackslash fnottext\{\texttt{text}\}}. This macro puts the \texttt{\textlangle text\textrangle} to the footnote. The \texttt{\textlangle numeric-label\textrangle} have to be 1 if only one such command is in the box. Second \texttt{\textbackslash fnotemark} inside the same box have to have the parameter 2 etc. The same number of \texttt{\textbackslash fnottexts} have to be written after the box as the number of \texttt{\textbackslash fnemarks} inserted inside the box. Example:

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

The marginal note can be printed by the \texttt{\textbackslash mnote\{\texttt{text}\}} macro. The \texttt{\textlangle text\textrangle} 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 \texttt{\textbackslash fixmnotes\textbackslash right} or \texttt{\textbackslash fixmnotes\textbackslash left}.

The \texttt{\textlangle text\textrangle} is formatted as a little paragraph with the maximal width \texttt{\textbackslash 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 \texttt{\mnote} in the text. The exceptions are possible by using the \texttt{\textbackslash up} keyword: \texttt{\mnote\textbackslash up\{\texttt{dimen}\}\{\texttt{text}\}}. You can set such \texttt{\textlangle dimen\textrangle} to each \texttt{\mnote} manually in final printing in order to margin notes do not overlap. The positive value of \texttt{\textlangle dimen\textrangle} shifts the note up and negative value shifts it down. For example \texttt{\mnote\textbackslash up\ 2\baselineskip\{\texttt{text}\}} shifts this marginal note two lines up.

1.3 Fonts

1.3.1 Font families

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

If you write \texttt{\fontfam\{\textlangle ?\textrangle\}} then all font families registered in Op\TeX are listed on the terminal and in the log file. If you write \texttt{\fontfam\{\textlangle catalog\textrangle\}} then a catalog of all fonts registered in\textsuperscript{2} You can declare \texttt{\textbackslash fnotenumglobal} if you want footnotes numbered in whole document from one or \texttt{\textbackslash fnotenumpages} if you want footnotes numbered at each page from one. Default setting is \texttt{\textbackslash fnotenumchapters}.
Op\TeX and available in your \TeX system is printed. The instructions how to register your own font family is appended in such catalog.

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

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

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 \textbf{important} text. % = This is \em \textit{important} text.
\it This is \em \textbf{important} text. % = This is/ \em \textbf{important} text.
\bf This is \em \textbf{important} text. % = This is \em \textit{important} text.
\bi This is \em \textbf{important} text. % = This is/ \em \textbf{important} text.

More about the Op\TeX 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 \texttt{\typosize[\langle fontsize\rangle/\langle baselineskip\rangle]} 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 \texttt{\ptunit=\langle something-else\rangle}, for instance \ptunit=1mm enlarges all font sizes declared by \texttt{\typosize}. Examples:

\begin{verbatim}
\typosize[10/12] % default of Plain TeX
\typosize[11/12.5] % font 11pt, baseline 12.5pt
\typosize[8/] % font 8pt, baseline unchanged
\end{verbatim}

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 \texttt{\typosize} for example) then you must first finalize the paragraph before closing the group: {\texttt{\typosize[12/14] \ldots\langle text of paragraph\rangle\ldots \par}}.

The command \texttt{\typoscale[\langle font-factor\rangle/\langle baselineskip-factor\rangle]} 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:

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

First usage of \texttt{\typosize} or \texttt{\typoscale} macro in your document sets so called main values, i.e. main font size and main baselineskip. They are internally saved in registers \texttt{\mainfysize} and \texttt{\mainbaselineskip}.
The \texttt{\textbackslash typoscale} command does scaling in respect to current values by default. If you want to do it in respect to main values, type \texttt{\textbackslash scalemain} immediately before \texttt{\textbackslash typoscale} command.

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

The \texttt{\textbackslash typosize} and \texttt{\textbackslash typoscale} macros initialize the font family by \texttt{\textbackslash rm}. You can re-size only the current font by the command \texttt{\textbackslash thefontsize[(font-size)]} or the font can be rescaled by \texttt{\textbackslash thefontscale[(factor)]}. These macros don’t change math fonts sizes nor baselineskip.

There is “low level” \texttt{\textbackslash setfontsize{(size-spec)}} command which behaves like a font modifier and sets given font size used by next variant selectors. It doesn’t change the font size immediately, but following variant selector does it. For example \texttt{\textbackslash setfontsize[15pt]\textbackslash currvar} sets current variant to 15pt.

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

More information about resizing of fonts is documented in the section 2.12.

1.3.3 Typesetting math

See the additional document \textit{Typesetting Math with \texttt{OpTeX}} for more details about this issue.

\texttt{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 \texttt{\textbackslash boldmath} variant. Most declared text font families (see \texttt{\textbackslash fontfam} in the section 1.3.1) are configured with recommended Unicode math font. This font is automatically loaded unless you specify \texttt{\textbackslash noloadmath} before first \texttt{\textbackslash 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 \texttt{\textbackslash loadmath{[⟨font-file]⟩}} or \texttt{\textbackslash loadmath{[⟨font-name]⟩}} before first \texttt{\textbackslash fontfam} command.

Hundreds math symbols and operators like in AMSTeX are accessible. For example \texttt{\textbackslash alpha α, \textbackslash geq ≥, \textbackslash sum ∑, \textbackslash sphericalangle ≪, \textbackslash bumpeq ≍). See AMSTeX manual or \textit{Typesetting Math with \texttt{OpTeX}} for complete list of math symbols.

The following math alphabets are available:

\begin{verbatim}
\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 A BC−XYZ
\frak % fracture abc−rzy, A BC−XYZ
\bbchar % double stroked letters ABC−XYZ
\bf % sans serif bold abc−xyz, ABC−XYZ
\bi % sans serif bold slanted abc−xyz, ABC−XYZ
\end{verbatim}

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

The math fonts can be scaled by \texttt{\textbackslash typosize} and \texttt{\textbackslash typoscale} macros. Two math fonts collections are prepared: \texttt{\textbackslash normalmath} for normal weight and \texttt{\textbackslash 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 \texttt{\textbackslash mathbox{⟨text⟩}} inside math mode. It behaves as \texttt{\textbackslash hbox{⟨text⟩}} (i.e. the \langle text⟩ is printed in horizontal non-math mode) but the size of the \langle text⟩ is adapted
to the context of math size (text or script or scriptscript). Moreover, there is the macro \mathstyles{\langle \text{math list}\rangle} which depends on the current math style. It is documented at the end of the section 2.15.

1.4  Typical elements of 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  \langle end of line\rangle
\chap Chapter title  \langle end of line\rangle
\sec Section title  \langle end of line\rangle
\secc Subsection title  \langle end of line\rangle

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

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

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

If a title is so long then it breaks to more lines in the output. It is better to hint the breakpoints because \TeX{} does not interpret the meaning of the title. User 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 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 in 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: \langle \caption \rangle \cskip \langle object \rangle or \langle object \rangle \cskip \langle \caption \rangle. The \cskip creates appropriate vertical space between them. Example:
The dependency of the computer-dependency on the age.

<table>
<thead>
<tr>
<th>age</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–1</td>
<td>unmeasured</td>
</tr>
<tr>
<td>1–6</td>
<td>observable</td>
</tr>
<tr>
<td>6–12</td>
<td>significant</td>
</tr>
<tr>
<td>12–20</td>
<td>extremal</td>
</tr>
<tr>
<td>20–40</td>
<td>normal</td>
</tr>
<tr>
<td>40–60</td>
<td>various</td>
</tr>
<tr>
<td>60–∞</td>
<td>moderate</td>
</tr>
</tbody>
</table>

This example produces:

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

You can see that the word “Table” followed by a number is added by the macro \caption/t.

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

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

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

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. 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...

\footnote{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 \langle label\rangle is appended to \chap, \sec, \secc, \caption/t, \caption/f or \eqmark. The alternative syntax is to use \label\langle label\rangle before mentioned commands (not necessarily directly before). The reference is realized by \ref\langle label\rangle or \pgref\langle label\rangle.

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]
\[ a^2 + b^2 = c^2 \]

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

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

You can use the \label\langle label\rangle before the \theorem, \definition etc. (macros defined with \numberedpar) if you want to reference these numbered objects. You can’t use \theorem\langle label\rangle because the optional parameter is reserved to another purpose here.

You can create a reference to whatever else by commands \label\langle label\rangle \wlabel{\langle text\rangle}. The connection between \langle label\rangle and \langle text\rangle is established. The \ref\langle label\rangle will print \langle text\rangle.

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

1.4.4 Hyperlinks, outlines

If the command \hyperlinks \langle color-in\rangle \langle color-out\rangle 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 \langle color-out\rangle. Others links are internal and they are colored by \langle color-in\rangle. 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 \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\footnote{More exactly, there are the same rules as for \code command, see section 1.4.7.}. You can insert the \| command in the \url argument as a potential breakpoint.

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

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

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

1.4.5 Lists

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

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

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}{⟨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 \indent value which is set to \parindent by default. The \ilevel register says what level of items is currently processed. Each \begitems starts \everylist tokens register. You can set, for example:

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

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

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 \crl \tskip2pt
  January & notebook & \$ 700 \cr
  February & skateboard & \$ 100 \cr
  July & yacht & k\$ 170 \crl}

generates the following result:

<table>
<thead>
<tr>
<th>Month</th>
<th>commodity</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>notebook</td>
<td>$ 700</td>
</tr>
<tr>
<td>February</td>
<td>skateboard</td>
<td>$ 100</td>
</tr>
<tr>
<td>July</td>
<td>yacht</td>
<td>k$ 170</td>
</tr>
</tbody>
</table>

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:
- \( p\{\langle\text{size}\rangle fL} \) fit left, i.e. left justified, ragged right,
- \( p\{\langle\text{size}\rangle fR} \) fit right, i.e. right justified, ragged left,
- \( p\{\langle\text{size}\rangle fC} \) fit center, i.e. ragged left plus right,
- \( p\{\langle\text{size}\rangle fS} \) fit special, short one-line paragraph centered, long paragraph normal,
- \( p\{\langle\text{size}\rangle fX} \) fit extra, left-right justified but last line centered.

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

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

The command \cr used in the \langle data\rangle part of the table is generally known from Plain \TeX. It marks the end of each row in the table. Moreover Op\TeX defines following similar commands:

- \( \text{\texttt{\textbf{\textbackslash cr}} \ldots} \) the end of the row with a horizontal line after it.
- \( \text{\texttt{\textbf{\textbackslash crl}} \ldots} \) the end of the row with a double horizontal line after it.
- \( \text{\texttt{\textbf{\textbackslash crli}} \ldots} \) like \texttt{\textbf{\textbackslash crl}} but the horizontal line doesn’t intersect the vertical double lines.
- \( \text{\texttt{\textbf{\textbackslash crlp}(\langle\text{list}\rangle \ldots} \) like \texttt{\textbf{\textbackslash crl}} but the lines are drawn only in the columns mentioned in comma separated \( \langle\text{list}\rangle \) of their numbers. For example \texttt{\textbf{\textbackslash crlp1-3,5}} is equal to \texttt{\textbf{\textbackslash crlp1,2,3,5}}.

The \texttt{\textbf{\textls{\textbackslash vskip}(\langle\text{dimen}\rangle} command works like the \texttt{\textbf{\textls{\textbackslash noalign{\textls{\textbackslash vskip}(\langle\text{dimen}\rangle}}} immediately after \texttt{\textbf{\textls{\textbackslash cr}}} commands but it doesn’t interrupt the vertical lines.

You can use following parameters for the \texttt{\textbf{\textls{\textbackslash table}}} macro. Default values are listed too.

\begin{verbatim}
\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
\vkern=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
\end{verbatim}

Example: if you do \texttt{\textbf{\textls{\textbackslash tabiteml}=${\textls{\textbackslash enspace}}\textbf{\textls{\textbackslash tabitemr}=${\textls{\textbackslash enspace}}}} then the \texttt{\textbf{\textls{\textbackslash table}}} acts like \texttt{\textbf{\textls{\textbackslash \LaTeX}}}’s array environment.

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

The \texttt{\textbf{\textls{\textbackslash frame}(\langle\text{text}\rangle} makes a frame around \langle\text{text}\rangle. You can put the whole \texttt{\textbf{\textls{\textbackslash table}}} into \texttt{\textbf{\textls{\textbackslash frame}}} if you need double-ruled border of the table. Example:

\begin{verbatim}
\frame{\texttt{\textbf{\textls{\textbackslash table\{c||l||r\}\{}}}} \texttt{\textbf{\textls{\textbackslash crl}}}
\texttt{\textbf{\textls{\textbackslash mspan3\{c\}\{}}}} \texttt{\textbf{\textls{\textbackslash bf Title}\texttt{\textbf{\textls{\textbackslash crl}}} \texttt{\textbf{\textls{\textbackslash noalign{\textbf{\textls{\textbackslash kern\textls{\textbackslash hhkern}}\texttt{\textbf{\textls{\textbackslash crli}}}}}}}}
\texttt{\textbf{\textls{\textbackslash first & second & third \texttt{\textbf{\textls{\textbackslash crli}}}}} seven & eight & nine \texttt{\textbf{\textls{\textbackslash crli}}}}
\end{verbatim}

15
creates the following result:

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
</tr>
<tr>
<td>second</td>
</tr>
<tr>
<td>third</td>
</tr>
<tr>
<td>seven</td>
</tr>
<tr>
<td>eight</td>
</tr>
<tr>
<td>nine</td>
</tr>
</tbody>
</table>

The \texttt{\textbackslash vspan\langle number\rangle\{\langle text\rangle\}} shifts the \langle text\rangle down in order it looks like to be in the center of the \langle number\rangle lines (current line is first). You can use this for creating tables like in the following example:

\begin{verbatim}
thistable{\tabstrut={\vrule height 20pt depth 10pt width 0pt}} \baselineskip=20pt \tablinespace=0pt \rulewidth=.8pt} \table{|8{c|}}{crlp(3-8)} \mspan2[c]{\{Singular\}} & \mspan3[c]{\{Plural\}} \crlp(3-8) \mspan2[c]{} & Neuter & Masculine & Feminine & Masculine & Feminine & Neuter \crl \mspan2[II] & Informal & \mspan3[c]{} & \mspan3[c]{} &\mspan6[c]{} \crl \mspan2[III] & Informal & \mspan2[O] & X & X & \mspan2[O] &\mspan2[O] \crlp(2,4-7) &\mspan4[c]{} &\mspan4[c]{} \crlp(2-8)
\end{verbatim}

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

The rule width of tables and implicit width of all \texttt{\textbackslash vrule}s and \texttt{\textbackslash hrule}s can be set by the command \texttt{\textbackslash rulewidth=\langle dimen\rangle}. The default value given by \TeX is 0.4pt.

The c, 1, r and p are default “declaration letters” but you can define more such letters by \texttt{\textbackslash def\_tabdeclare\langle letter\rangle\{\langle left\rangle##\langle right\rangle\}}. More about it is in technical documentation in section 2.30.5. See the definition of the \texttt{\tabdeclarec} macro, for example.

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

### 1.4.7 Verbatim

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The \ttline influences the line numbering by the same way as in \begtt...\endtt environment. If \ttline=-1 then real line numbers are printed (this is default). If \ttline=-1 then no line numbers are printed.
The \verbinput can be controlled by \everytt, \ttindent just like in \begtt...\endtt.

The \begtt...\endtt pair or \verbinput can be used for listings of codes. Automatic syntax
highlighting is possible, for example \begtt \hisyntax{C} activates colors for C programs. Or
\verbinput \hisyntax{HTML} (-) file.html can be used for HTML or XML codes. Op\TeX
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 124) 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 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}}}{⟨case-mark⟩}{⟨case-outline⟩}
\regmacro{\def\mylogo{\hbox{{\it My}\ LoGo}}}{⟨case-toc⟩}{⟨case-mark⟩}{⟨case-outline⟩}
\regmacro{\def\mylogo{MyLoGo}}{⟨case-toc⟩}{⟨case-mark⟩}{⟨case-outline⟩}

1.5.2 Making the index

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

The \ii command (insert to index) declares the word separated by the space as the index
item. This declaration is represented as invisible 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 \ii:

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

or:

Now we’ll deal with the \iid resistor .

Note that the dot or comma have to be separated by space when \iid is used. This space
(before dot or comma) is removed by the macro in the current text.
The multiple-words entries are commonly 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 space/linear. You can do this by the shorthand ,@ at the end of the \ii parameter. Example:

\ii linear/space, vector/space,@

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

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

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

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

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

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

\sec Index
\begmulti 3 \makeindex \endmulti

Only “pure words” can be propagated to the index by the \ii command. It means that there cannot be any macro, \TeX primitive, math selector etc. But there is another possibility to create such complex index entry. Use “pure equivalent” in the \ii parameter and map this equivalent to the real word which is printed in the index. 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\{\langle label \rangle\} (or \cite\{\langle label-1\rangle, \langle label-2\rangle, \ldots, \langle label-n\rangle\}) creates the citation in the form [42] (or [15, 19, 26]). If \shortcitations is declared at the beginning of the document then continuous sequences of numbers are re-printed like this: [3–5, 7, 9–11]. If \sortcitations is declared then numbers generated by one \cite command are sorted upward.

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

The \rcite\{\langle labels \rangle\} creates the same list as \cite\{\langle labels \rangle\} but without the outer brackets. Example: \[\rcite\{tbn\}, pg.~13\] creates [4, pg. 13].

You can define alternative formatting of \cite command. Example:
\[\text{\def\cite\[#1\]{\{\rcite\[#1\]\}}} \text{\cite\{<label>\}} \text{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\{\langle label \rangle\} commands.
- By \usebib/\{\langle type \rangle\} \{\langle style \rangle\} \{\langle bib-base \rangle\} command which reads *.bib files directly.

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

References created manually using \bib\{\langle label \rangle\} command.
\bib\{tst\} P. Olšák. \textit{Typografický systém \TeX{}}

If you are using \nonumcitations then you need to declare the \{\langle marks \rangle\} used by \cite command. To do it you must use long form of the \bib command in the format \bib\{\langle label \rangle\} = \{\langle mark \rangle\}. The spaces around equal sign are mandatory. Example:
\bib\{tbn\} = \{Olšák, 2001\}


Direct reading of *.bib files is possible by \usebib macro. This macro reads and uses macro package \texttt{librarian.tex} by Paul Isambert. The usage is:
\usebib/c \{\langle style \rangle\} \{\langle bib-base \rangle\} \% sorted by \cite-order (c=cite),
\usebib/s \{\langle style \rangle\} \{\langle bib-base \rangle\} \% sorted by style (s=style).
\% example:
\nocite\[*\] \usebib/s \{simple\} op-biblist \% prints all from op-biblist.bib

The \{\langle bib-base \rangle\} is one or more *.bib database source files (separated by spaces and without extension) and the \{\langle style \rangle\} is the part of the filename bib-\{\langle style \rangle\}.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 \{\langle bib-base \rangle\} 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 mentioned bib-record should be printed in the reference list. If \nocite\[*\] is used then all records from \{\langle bib-base \rangle\} are printed.
1.6 Graphics

1.6.1 Colors

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

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

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

The color selectors work locally in groups by default but with limitations. See the technical documentation, section 2.20 for more information.

The basic colors \Blue, \Red, \Cyan, \Yellow etc. are defined with CMYK components using \setcmykcolor. On the other hand, you can define a color with three RGB components and \morecolors defines such RGB colors. By default, the color model isn’t converted but only stored to PDF output for each used color. Thus, there may be a mix of color models in the PDF output which is not good idea. You can overcome this problem by declaration \onlyrgb or \onlycmyk. Then only 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:

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

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

The following example defines the macro for the colored text on the colored background.

Usage: \coloron\textit{background}/\textit{foreground}\{\textit{text}\}

The \coloron can be defined as follows:

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

1.6.2 Images

The \inspic {\textit{filename}.\textit{extension}} or \inspic \textit{filename}.\textit{extension}\space inserts the picture stored in the graphics file with the name \textit{filename}.\textit{extension} to the document.

---

\(^5\) Printed output is more equal to the monitor preview specially if you are using ICC profile for your printer.
can set the picture width by \picw\langle dimen\rangle 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\langle img/\rangle supposes that image files are in \img{} subdirectory. Note: the directory name must end by / in the \picdir declaration.

Inkscape\footnote{A powefull and free wysiwyg editor for creating vector graphics.} is able to save a picture to PDF and labels of the picture to another file\footnote{Choose “Omit text in PDF and create LaTeX file” option.}. This second file should be read by \TeX in order to print labels in the same font as document font. \opTeX{} supports this feature by \inkinspic{}\langle filename\rangle.pdf command. It reads and displays both: PDF image and labels generated by Inkscape.

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

1.6.3 PDF transformations

All typesetting elements are transformed by linear transformation given by the current transformation matrix. The \pdfsetmatrix \langle a \rangle \langle b \rangle \langle c \rangle \langle d \rangle command makes the internal multiplication with the current matrix so linear transformations can be composed. One linear transformation given by the \pdfsetmatrix above transforms the vector [0, 1] to [(a), (b)] and [1, 0] to [(c), (d)]. 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 have to be the same from \TeX’s point of view as from transformation point of view when \pdfrestore is processed. Due to this fact the \pdfsave \rlap{\langle transformed text\rangle} \pdfrestore or something similar is recommended.

\opTeX{} provides two special transformation macros \pdfscale and \pdfrotate:

\begin{verbatim}
\pdfscale{\langle horizontal-factor\rangle}{\langle vertical-factor\rangle}
\pdfrotate{\langle angle-in-degrees\rangle}
\end{verbatim}

These macros simply calls the properly \pdfsetmatrix command.

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

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

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

text1
\begin{tikzpicture}
\end{tikzpicture}
text2
\begin{tikzpicture}
\end{tikzpicture}
text3

You can see that \TeX{} knows nothing about dimensions of transformed material, it treats it as with a zero dimension object. The \transformbox{\langle transformation\rangle}{\langle text\rangle} macro solves
the problem. This macro puts the transformed material to a box with relevant dimension. The \textit{transformation} parameter includes one or more transformation commands \texttt{pdfsetmatrix}, \texttt{pdfscale}, \texttt{pdfrotate} with their parameters. The \texttt{text} is transformed text.

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

The \texttt{\rotbox{\langle deg\rangle}{\langle text\rangle}} is shortcut for \texttt{\transformbox{\pdfrotate{\langle deg\rangle}}{\langle text\rangle}}.

1.6.4 Ovals, circles

The \texttt{\inoval{\langle text\rangle}} creates a box like this: \texttt{text}. Multiline text can be put in an oval by the command \texttt{\inoval{\vbox{\langle text\rangle}}}.

The \texttt{\ovalparams=\{\langle settings\rangle\}} can be re-declared global settings by \texttt{\ovalparams=\{\langle settings\rangle\}}. The default settings are:

\begin{verbatim}
  \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-right margin, top-bottom margin
\end{verbatim}

The total distance from text to oval boundary is \texttt{\hhkern+\roundness} at the left and right sides and \texttt{\vvkern+\roundness} at the top and bottom sides of the text.

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

The \texttt{\incircle[\ratio=1.8]{\langle text\rangle}} creates a box like this \texttt{text}. The \texttt{\ratio} parameter means width/height. The usage is analogous like for oval. The default parameters are

\begin{verbatim}
  \circleparams=\{\ratio=1 \fcolor=\Yellow \lcolor=\Red \lwidth=0.5bp 
  \shadow=N \ignoremargins=N \hhkern=2pt \vvkern=2pt\}
\end{verbatim}

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

\begin{verbatim}
  \clipincircle 3cm 3.5cm 6cm 7cm \{\picw=6cm \inspic{myphoto.jpg}\}
\end{verbatim}

1.6.5 Putting images and texts wherever

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

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

1.7 Others

1.7.1 Using more languages

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

```
en(English) enUs(Englishmax) engh(UKenglish) it(italian) ia(interlingua) id(indonesian) cs(czech) sk(slovak)
de(nGerman) fr(french) pl(Polish) cy(welsh) da(Danish) es(Spanish) sl(Slovenian) fi(Finnish) hu(Hungarian)
tr(Turkish) et(Estonian) eu(Basque) ga(Irish) nb(Bokmal) nn(Nynorsk) pt(Dutch) pt(Portuguese) ro(Romanian)
hr(Croatian) zh(Pinyin) is(Icelandic) hsbs(UpperSorbian) af(Afrikaans) gl(Galician) kmr(Kurmanji) tk(Turkmen)
la(Latin) lac(classicLatin) lat(latinGall) elm(elm) elp(Greek) grc(ancientGreek) ca(Catalan) cop(Coptic)
nm(Mongolian) sa(Sanskrit) ru(Russian) uk(Ukrainian) lo(Lao) ka(georgian) km(Khmer) la(Latin) lac(plainLatin)
cslang(CSlang) sklang(SKlang)
```

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

For compatibility with \texttt{\cslang} plain, there are macros \texttt{\ehyph}, \texttt{\chyph}, \texttt{\shyph} which are equivalent to \texttt{\uselanguage}, \texttt{\uselanguage} and \texttt{\uselanguage}. You can switch between language patterns by \texttt{\iso-code} \texttt{lang} commands mentioned above. Default is \texttt{\cslang}.

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 \texttt{\iso-code lang}. OpTEX declares these words only for few languages: Czech, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English. If you need to use these words in another languages or you want to auto-generate more words in your macros, then you can declare it by \texttt{\sdef} or \texttt{\_langw} commands as shown in section 2.37.3.

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

If you declare \texttt{\iso-code quotes}, then the control sequences \texttt{\"} and \texttt{"} should be used like this: \texttt{\\langle quoted text\rangle"} or \texttt{\\langle quoted text\rangle"} (note that the terminating character is the same but it isn’t escaped). This prints language dependent normal or alternative quotes around \texttt{\langle quoted text\rangle}. The language is specified by \texttt{\iso-code}. OpTEX declares quotes only for Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English (\texttt{\csquotes}, \texttt{\dequotes}, \ldots, \texttt{\enquotes}). You can simply define your own quotes as shown in section 2.37.3. The \texttt{\"} is used for quotes visualy more similar to the " character which can be primary quotes or secondary quotes depending on the language rules. May be you want to alternate meaning of these two types of quotes. Use \texttt{\iso-code quotes\altquotes} in such case.

### 1.7.2 Pre-defined styles

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

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

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

\texttt{\address}\langle\text{first line of address}\rangle\langle\text{second line of address}\rangle\langle\text{etc.}\rangle\langle\text{empty line}\rangle

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

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

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

1.7.3 Loading other macro packages

You can load more macro packages by \texttt{\input⟨file-name⟩} or by \texttt{\load⟨file-names⟩}. The first case (\texttt{\input}) is \TeX primitive command, it can be used in the alternative old syntax \texttt{\input ⟨filename⟩⟨space⟩} too. The second case (\texttt{\load}) allows to specify 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 \texttt{⟨filename⟩.opm} or \texttt{⟨filename⟩.tex} or \texttt{⟨filename⟩}, first occurrence wins. Example:

\texttt{\load [qrcode, tikz]}

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

It is strongly recommended to use the \texttt{\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 \texttt{\input} primitive.

1.7.4 Lorem ipsum dolor sit

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

1.7.5 Logos

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

We are using \texttt{\TeX/} because it is cool. \texttt{\OpTeX/} is better than \texttt{\LaTeX}.

1.7.6 The last page

The number of the last page (it may be different from number of pages) is expanded by \texttt{\lastpage} macro. It expands to ? in first \TeX run and to the last page in next \TeX runs.
There is an example for footlines in the format “current page / last page”:
\footline={\hss \fixedrm \folio/\lastpage \hss}

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

1.7.7 Use Op\TeX

The command \useOpTeX (or \useoptex) does nothing in Op\TeX 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 Op\TeX 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
\tit item1,item2 % insertion the items to the index
\makeindex % the index is generated

\label [labname] % link target location
\ref [labname] % link to the chapter, section, subsection, equation
\pgref [labname] % link to the page of the chapter, section, ...

\caption/t % a numbered table caption
\caption/f % a numbered caption for the picture
\eqmark % a numbered equation

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

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

\load [filenames] % loading macro files
\fontfam [FamilyName] % selection of font family
\typosize [font-size/baselineskip] % 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
1.9 Compatibility with Plain TeX

All macros of Plain TeX are re-written in OpTeX. Common macros should work in the same sense as in original Plain TeX. Internal control sequences like \p@ or \f@tt are removed and mostly replaced by control sequences prefixed by _ (like _this). If you need to use basic set of old Plain TeX control sequences like \p@ (for example you are reading an old macro file), use \load[plain-at].

All primitives and common macros have two control sequences with the same meaning: in prefixed and unprefixed form. For example \hbox is equal to _hbox. Internal macros of OpTeX have and use only prefixed form. User should use unprefixed forms, but prefixed forms are accessible too, because the _ is set as a letter category code globally (in macro files and in users document too). User should re-define unprefixed forms of control sequences without worries that something internal will be broken (only the sequence \par cannot be re-defined without change of internal TeX behavior because it is hard-coded in TeX, unfortunately).

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

Plain 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 ’, ‘, , , =, , , , are undefined 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 letter with 1in margins but as A4 with 2.5cm margins. You can change it, for example by \margins/1 letter (1,1,1,1)in. This example sets the classical Plain TeX page layout.

The origin for typographical area is not at top left 1in 1in coordinates but at 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 to sections but original Plain TeX declares this control sequence for math secans.

---

8 The math accents macros like \acute, \bar, \dot, \hat still work.
Chapter 2

Technical documentation

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

\luatex -ini optex.ini

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

\optex optex-doc.tex

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

A knowledge about \TeX is expected from the reader. You can see a short document \TeX in a Nutshell or more detail \TeX by topic.

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

2.1 The main initialization file

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

```
%% This is part of OpTeX project, see http://petr.olsak.net/optex
%% OpTeX ini file
%% Petr Olsak <project started from: Jan. 2020>
\catcode \{=1 % left brace is begin-group character
\catcode \}=2 % right brace is end-group character
\catcode \$=3 % dollar sign is math shift
\catcode \&=4 % ampersand is alignment tab
\catcode \#=6 % hash mark is macro parameter character
\catcode \^^K=7 % circumflex and uparrow are for superscripts
\catcode \^^A=8 % downarrow is for subscripts
\catcode \^^I=10 % ascii tab is a blank space
\catcode \_=11 % underline can be used in control sequences
\catcode \^=13 % tilde is active
\catcode \^^a0=13 % non breaking space in Unicode
\catcode 127=12 % normal character
\def\optexversion{Beta 0.18 Dec.2020}
\def\fmtname{OpTeX}
\newlinechar=`\^^J
\ifx\directlua\undefined
```

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

The \optexversion and \fmtname are defined.

```
\def\optexversion{Beta 0.18 Dec.2020}
\def\fmtname{OpTeX}
```

We check if Lua\TeX engine is used at -ini state. And the `\n` character is set as `\newlinechar.`
The basic macros for macro file syntax is defined, i.e. \_endcode, \_doc and \_cod. The \_codedecl will be re-defined later.

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

The file \texttt{optex.lua} is embedded into the format as byte-code. It is documented in section \ref{sec:optex}.

\_everyjob register is initialized and the format is saved by the \_dump command.
2.2 Concept of name spaces of control sequences

2.2.1 Prefixing internal control sequences

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

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

\textit{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 \textit{OpTeX} macros as internal. User can redefine such prefixed control sequence only if he/she explicitly know what happens.

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

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

\texttt{\public <list of control sequences> ;}

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

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

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

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

2.2.2 Name space of control sequences for users

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

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

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

\texttt{\int\_a^b} ... is interpreted as \texttt{\int\_a\^b}
\texttt{\max\_M} ... is interpreted as \texttt{\max\_M}
\texttt{\alpha\_{ij}} ... is interpreted as \texttt{\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 \texttt{\mathsboff}. After this, you can still write $\$\int\_a^b\$ (Unicode) or $\$\int\_a\^b\$ without problems but \texttt{\int\_a^b} yields to undefined control sequence.
\texttt{\textbackslash int\_a}. You can activate this feature again by \texttt{\textbackslash mathbol}. The effect will take shape from next line read from input file.

- Control sequences in the form \texttt{\textbackslash \_\{word\}} is intended for package writers as internal macros for a package with \texttt{\{pkg\}} identifier, see section 2.2.4.

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

### 2.2.3 Macro files syntax

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

The code in macro files starts by \texttt{\textbackslash codedecl} and ends by \texttt{\textbackslash endcode}. The \texttt{\textbackslash endcode} is equivalent for \texttt{\textbackslash endinput}, so documentation can follow. The \texttt{\textbackslash codedecl} has syntax:

\texttt{\textbackslash codedecl \sequence \{Name <version>\}}

If the mentioned \texttt{\sequence} is defined, then \texttt{\textbackslash codedecl} does the same as \texttt{\textbackslash endinput}: this protect from reading the file twice. We suppose, that \texttt{\sequence} is defined in the macro file.

It is possible to use the \texttt{\textbackslash doc ... \textbackslash 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 \texttt{doc.opm} macros like in this documentation.

### 2.2.4 Name spaces for package writers

Package writer should use internal names in the form \texttt{\_\{pkg\}\_\{sequence\}}, where \texttt{\{pkg\}} is a package label. For example: \texttt{\textbackslash qr\_utfstring} from \texttt{qrcode.opm} package.

The package writer needs not write repeatedly \texttt{\_\{pkg\}\_\{pkg\}\_bar} etc. again and again in the macro file.\footnote{We have not adatped the idea from expl3 language:)} When the \texttt{\\_namespace \{\{pkg\}\}} is declared at the beginning of the macro file then all occurrences of \texttt{\_\{foo\}} will be replaced by \texttt{\_\{pkg\}\_\{foo\}} at the input processor level. The macro writer can write (and backward can read his/her code) simply with \texttt{\_\{foo\}}, \texttt{\_\{bar\}} control sequences and \texttt{\_\{pkg\}\_\{foo\}}, \texttt{\_\{pkg\}\_\{bar\}} control sequences are processed internally. The scope of the \texttt{\\_namespace} command ends at the \texttt{\\_endnamespace} command or when another \texttt{\\_namespace} is used. This command checks if the same package label is not declared by the \texttt{\\_namespace} twice.

The \texttt{\\_nspublic} macro does \texttt{\let\texttt{\{foo\} = \_\{\{pkg\}\_\{foo\}\}} when \texttt{\\_namespace \{\{pkg\}\}} is declared. And the \texttt{\\_nsprivate} macro does reverse operation to it. Example: you can define \texttt{\def \\_macro{...}} and then set it to the user name space by \texttt{\\_nspublic \_macro};

Don’t load another packages (which are using their own name space) inside your name space. Do load them before your \texttt{\\_namespace \{\{pkg\}\}} is initialized. Or close your name space by \texttt{\\_endnamesp} and open it again (after other packages are loaded) by \texttt{\\_resetnamespace \{\{pkg\}\}}.

If the package writer needs to declare a control sequence by \texttt{\\_newif\_\{if\}} then there is an exception of the rule described above. Use \texttt{\\_newif\_\{if\}\_\{if\} \_\{if\}} for example \texttt{\\_newif\_\{if\}\_\{if\}\_\{if\}}. Then the control sequences \texttt{\_\{qr\}_incornertrue} and \texttt{\_\{qr\}_incornersfalse} can be used (or the sequences \texttt{\_\{incornertrue} and \texttt{\_\{incornersfalse} when \texttt{\\_namespace \{qr\}} is used).

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

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

- Don’t use a control sequences from user name space in the macro bodies if there is not explicit and documented reason to do this.
- Don’t declare control sequences in the user name space if there is not explicit and documented reason to do this.
- Use control sequences from OpTeX and primitive name space in read only mode if there is not explicit and documented reason to redefine them.
- Use \texttt{\_\{\{pkg\}\_\{name\}\}} for your internal macros or \texttt{\_\{name\}} if the \texttt{\\_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 and/or write more documentation after the \_endcode command.

If the macro file accepts these recommendations then it should be named by \(\langle\text{filename}\rangle\).opm where \(\langle\text{filename}\rangle\) differs from file names used directly in OpTEX and from other published macros. This extension .opm has a precedence before .tex when the \load macro is used.

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

2.2.6 The implementation of the name spaces

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

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

\_let\_directlua = \directlua
\_directlua {
% enable all \TeX primitives with _ prefix
\tex.enableprimitives('_', \tex.extraprimitives('tex'))
% enable all primitives without prefixing
\tex.enableprimitives('', \tex.extraprimitives())
% enable all primitives with _ prefix
\tex.enableprimitives('_', \tex.extraprimitives())
}
\ea
\expandafter
\_let \_ea = \expandafter % usefull shortcut
\_long\_def \_xargs #1#2{\_ifx #2;\_else \_ea#1\_ea#2\_ea\_xargs \_ea #1\_fi}
\_def \_pkglabel{}
\_def \_public {\_xargs \_publicA}
\_def \_publicA #1{\_ea\_let \csname _\_csstring #1\_endcsname =#1}
\_public \public \private \xargs \ea ;

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

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

\_def \_codedecl #1#2{\_ifx #1\_undefined \_wterm{#2}\_else \_expandafter \_endinput \_fi}
\_def \_wterm {\_immediate \_write16 }
\_public \public \private \xargs \ea ;

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

\_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.

```
\def \pkglabel{}
\def \namespace #1{% 
  \ifcsname namesp:#1\endcsname \errmessage{The name space "#1" is used already, it cannot be used twice}% 
  \else \resetnamespace{#1}\fi 
}
\def \resetnamespace #1{% 
  \ea \gdef \csname namesp:#1\endcsname {}% 
  \gdef \pkglabel{_#1}% 
  \directlua{ 
    callback.add_to_callback("process_input_buffer", 
    function (str) 
      return string.gsub(str, \_nbb\[.\](\[a-zA-Z\]), \_nbb _#1\_pcent 1") 
    end, "_namespace") 
  }% 
} 
\def \endnamespace {% 
  \directlua{ callback.remove_from_callback("process_input_buffer", "_namespace") }% 
  \gdef \pkglabel{}% 
} 
\def \nspublic {\xargs \nspublicA} 
\def \nspublicA #1{\ea \let \csname \pkglabel _\csstring #1\endcsname} 
\def \nsprivate {\xargs \nsprivateA} 
\def \nsprivateA #1{\ea \let \csname \pkglabel _\csstring #1\endcsname =#1} 
```

2.3 pdfTEx initialization

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

```
\codedecl \pdfprimitive {LuaTeX initialization code <2020-02-21>} % preloaded in format
\let \pdfpagewidth \pagewidth
\let \pdfpageheight \pageheight
\let \pdfadjustspacing \adjustspacing
\let \pdfprotrudechars \protrudechars
\let \pdfnoligatures \ignoreligaturesinfont
\let \pdffontexpand \expandglyphsinfont
\let \pdfxform \saveboxresource
\let \pdflastxform \lastsavedboxresourceindex
\let \pdfximage \saveimageresource
\let \pdflastximage \lastsavedimageresourceindex
\let \pdfximagerepages \lastsavedimageresourcepages
\let \pdfoutput \outputmode
\let \pdffontpagewidth \pagewidth
\let \pdffontpageheight \pageheight
\let \pdffontadjustspacing \adjustspacing
\let \pdffontprotrudechars \protrudechars
\let \pdffontnoligatures \ignoreligaturesinfont
\let \pdffontexpand \expandglyphsinfont
\let \pdffontxform \saveboxresource
\let \pdffontlastxform \lastsavedboxresourceindex
\let \pdffontximage \saveimageresource
\let \pdffontlastximage \lastsavedimageresourceindex
\let \pdffontximagerepages \lastsavedimageresourcepages
\let \pdffontoutput \outputmode
\let \pdfnormaldeviate \normaldeviate
\let \pdfuniformdeviate \uniformdeviate
\let \pdfsetrandomseed \setrandomseed
\let \pdfrandomseed \randomseed
```
\_directlua {tex.enableprimitives('pdf', {'tracingfonts'})}
\_protected\_def \_pdftexversion {\_numexpr 140\_relax}
\_protected\_def \_pdflastlink {\_numexpr \_pdffeedback lastlink \_relax}
\_protected\_def \_pdfreval {\_numexpr \_pdffeedback reval \_relax}
\_protected\_def \_pdflastobj {\_numexpr \_pdffeedback lastobj \_relax}
\_protected\_def \_pdflastannot {\_numexpr \_pdffeedback lastannot \_relax}
\_def \_pdfxformname {\_pdffeedback xformname}
\_protected\_edef \_pdfcreationdate {\_pdffeedback creationdate}
\_protected\_def \_pdffontname {\_pdffeedback fontname}
\_protected\_def \_pdffontobjnum {\_pdffeedback fontobjnum}
\_protected\_def \_pdffontsize {\_pdffeedback fontsize}
\_def \_pdfpageref {\_pdffeedback pageref}
\_protected\_def \_pdfcolorstackinit {\_pdffeedback colorstackinit}
\_protected\_def \_pdfliteral {\_pdffeedback literal}
\_protected\_def \_pdfcolorstack {\_pdffeedback colorstack}
\_protected\_def \_pdfsetmatrix {\_pdffeedback setmatrix}
\_protected\_def \_pdfsave {\_pdffeedback save \_relax}
\_protected\_def \_pdfrestore {\_pdffeedback restore \_relax}
\_protected\_def \_pdfobj {\_pdffeedback obj}
\_protected\_def \_pdfrefobj {\_pdffeedback refobj}
\_protected\_def \_pdfannot {\_pdffeedback annot}
\_protected\_def \_pdfstartlink {\_pdffeedback startlink}
\_protected\_def \_pdfendlink {\_pdffeedback endlink \_relax}
\_protected\_def \_pdfoutline {\_pdffeedback outline}
\_protected\_def \_pdfdest {\_pdffeedback dest}
\_protected\_def \_pdfthread {\_pdffeedback thread}
\_protected\_def \_pdfstartthread {\_pdffeedback startthread}
\_protected\_def \_pdfendthread {\_pdffeedback endthread \_relax}
\_protected\_def \_pdfinfo {\_pdffeedback info}
\_protected\_def \_pdfcatalog {\_pdffeedback catalog}
\_protected\_def \_pdfnames {\_pdffeedback names}
\_protected\_def \_pdfincludechars {\_pdffeedback includechars}
\_protected\_def \_pdffontattr {\_pdffeedback fontattr}
\_protected\_def \_pdfmapfile {\_pdffeedback mapfile}
\_protected\_def \_pdfmapline {\_pdffeedback mapline}
\_protected\_def \_pdftrailer {\_pdffeedback trailer}
\_protected\_def \_pdfglyphbounctable {\_pdffeedback glyptable}
\_protected\_edef \_pdfcompresslevel {\_pdffeedback compresslevel}
\_protected\_edef \_pdfobjcompresslevel {\_pdffeedback objcompresslevel}
\_protected\_edef \_pdfdecimaldigits {\_pdffeedback decimaldigits}
\_protected\_edef \_pdfgamma {\_pdffeedback gamma}
\_protected\_edef \_pdfimageresolution {\_pdffeedback imageresolution}
\_protected\_edef \_pdfimageapplygamma {\_pdffeedback imageapplygamma}
\_protected\_edef \_pdfimagegamma {\_pdffeedback imagegamma}
\_protected\_edef \_pdfimagembicolor {\_pdffeedback imagembicolor}
\_protected\_edef \_pdfimageaddfilename {\_pdffeedback imageaddfilename}
\_protected\_edef \_pdfpkresolution {\_pdffeedback pkresolution}
\_protected\_edef \_pdfinclusioncopyfonts {\_pdffeedback inclusioncopyfonts}
\_protected\_edef \_pdfinclusionerrorlevel {\_pdffeedback inclusionerrorlevel}
\_protected\_edef \_pdfgentounicode {\_pdffeedback gentounicode}
\_protected\_edef \_pdfpagebox {\_pdffeedback pagebox}
\_protected\_edef \_pdfminorversion {\_pdffeedback minorversion}
\_protected\_edef \_pdfuniquerename {\_pdffeedback uniquerename}
\_protected\_edef \_pdfhorigins {\_pdffeedback horigins}
\_protected\_edef \_pdfvorigins {\_pdffeedback vorigins}
\_protected\_edef \_pdflinkmargin {\_pdffeedback linkmargin}
2.4 Basic macros

We define first bundle of basic macros.

\public

\pdftexversion \pdftexrevision \pdflastlink \pdflastobj

\pdflastannot \pdfxformname \pdfcreationdate \pdffontname \pdffontobjnum

\pdffontsize \pdfpageref \pdfcolorstackinit \pdfliteral \pdfcolorstack

\pdfsetmatrix \pdfsave \pdfrestore \pdfobj \pdfrefobj \pdfannot

\pdfstartlink \pdfendlink \pdfoutline \pdfdest \pdfthread \pdfstartthread

\pdfendthread \pdfinfo \pdfcatalog \pdfnames \pdfincludechars \pdffontattr

\pdfmapfile \pdfmapline \pdftrailer \pdfglyphtounicode \pdfcompresslevel

\pdfobjcompresslevel \pdfdecimaldigits \pdfgamma \pdfimageresolution

\pdfimageapplygamma \pdfimagegamma \pdfimagehicolor \pdfimageaddfilename

\pdfpkresolution \pdfinclusioncopyfonts \pdfinclusionerrorlevel

\pdfgentounicode \pdfpagebox \pdfminorversion \pdfuniqueresname \pdfhorigin

\pdfvorigin \pdflinkmargin \pdfdestmargin \pdfthreadmargin \pdfpagesattr

\pdfpageattr \pdfpageresources \pdfxformattr \pdfxformresources \pdfpkmode ;

_\pdfminorversion = 5
_\pdfobjcompresslevel = 2
_\pdfcompresslevel = 9
_\pdfdecimaldigits = 3
_\pdfpkresolution = 600

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

\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⟩}.

\sdef {⟨char⟩}{⟨body⟩} puts the ⟨char⟩ as active character and defines it as {⟨body⟩}. You can declare a macro with parameters too. For example \sdef @#1{...$1...}.
\texttt{\_cs \{⟨text⟩\}\endcsname} is only a shortcut to \texttt{\csname ⟨text⟩\endcsname}, but you need one more \_ea if you need to get the real control sequence \texttt{⟨text⟩}.

\texttt{\_trycs \{⟨csname⟩\}\{⟨text⟩\}} expands to \texttt{\csname ⟨csname⟩\endcsname} if it is defined else to the \texttt{⟨text⟩}.

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

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

\texttt{\loggingall} and \texttt{\tracingall} are defined similarly as in \LaTeX, but they print more logging information to the log file and to the terminal.

Write a warning if the user did not to load a Unicode Font or if there were unresolved references. \texttt{\_byehook} is used in the \texttt{\bye} macro.

### 2.5 Allocators for \TeX registers

Like plain\TeX, the allocators \texttt{\newcount, \newwrite, etc.} are defined. The registers are allocated from 256 to the \texttt{\_mai\langle type\rangle} which is 65535 in Lua\TeX.

Unlike in Plain\TeX, the mentioned allocators are not \texttt{\outer}.

User can use \texttt{\dimen0} to \texttt{\dimen255} and similarly for \texttt{\skip, \muskip, \box and \toks} directly.

User can use \texttt{\count20} to \texttt{\count255} directly too. This is the same philosophy like in old plain\TeX, but the range of directly used registers is wider.

Inserts are allocated form 254 to 201 using \texttt{\newinsert}.

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

```latex
\texttt{\newcount \_maicount} \% redefine maximal allocation index as variable
\texttt{\_maicount = \maicount} \% first value is top of the array
\texttt{\def\newcountarray #1[#2]{% \newcountarray \foo[#1]
   \global\advance\_maicount by \#2\relax
   \ifnum\_countalloc > \_maicount
     \errmessage{No room for a new array of \string\count}\\fi
   \else
     \global\chardef#1=\_maicount
   \fi}
\texttt{\def\usecount #1[#2]{% \usecount \foo[#1]
   \count\numexpr#1+\#2\relax}
```
The limits are set first.

Each allocation macro needs its own counter.

The common allocation macro \texttt{\_allocator \sequence \{\texttt{\_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.

The allocation macros \texttt{\_newcount, \_newdimen, \_newskip, \_newmuskip, \_newbox, \_newtoks, \_newread, \_newwrite} and \texttt{\_newfam} are defined.

The \texttt{\_newinsert} macro is defined differently than others.

% loaded in format

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

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

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

\_public \_newcount \_newdimen \_newskip \_newmuskip \_newbox \_newtoks \_newread \_newwrite \_newfam ;

\_newinsert \_insertalloc=255
\_chardef\_insertmin = 201
\_def\_newinsert #1{\_ifnum\_insertalloc <\_insertmin
\_errmsg\_No\_room\_for\_a\_new\_\_string\insert\_\_endcsname\%\_else
\_global\_chardef#1=\_insertalloc
\_vlog{\_string#1=\_string\insert\_\_endcsname\_the\_\cs{\_#2alloc))\%}
\_fi
}
Other allocation macros $\texttt{\newattribute}$ and $\texttt{\newcatcodetable}$ have their counter allocated by the $\texttt{\newcount}$ macro.

\begin{verbatim}
\_newcount \_attributealloc \_attributealloc=0
\_chardef\_maiatribute=\_maicount
\_def\_newattribute #1{\_allocator #1{attribute}\_attributedef}
\_newcount \_catcodetablealloc \_catcodetablealloc=10
\_chardef\_maicatcodetable=32767
\_def\_newcatcodetable #1{\_allocator #1{catcodetable}\_chardef}
\_public \newattribute \newcatcodetable ;
\end{verbatim}

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

\begin{verbatim}
\_newcount \_tmpnum \_newcount \_tmpnum
\_newdimen \_tmpdim \_newdimen \_tmpdim
\end{verbatim}

A few registers are initialized like in plain\TeX. We absolutely don’t support the $\$\texttt{\z@}$ category dance, so $\texttt{\z@skip}$, $\texttt{\p@}$ etc. are not defined in Op\TeX. If you need such control sequences then you can initialize them by $\texttt{\load[plain-at]}$.

Only the $\texttt{\zo}$ and $\texttt{\zoskip}$ (equivalents to $\texttt{\z@}$ and $\texttt{\z@skip}$) are declared here and used in some internal macros of Op\TeX for improving speed.

\begin{verbatim}
\_newdimen \maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
\_newdimen \zo \zo=0pt
\_newskip \hideskip \hideskip=-1000pt plus 1fill % negative but can grow
\_newskip \centering \centering=0pt plus 1000pt minus 1000pt
\_newskip \zoskip \zoskip=0pt plus0pt minus0pt
\_newbox \voidbox % permanently void box register
\_public \maxdimen \hideskip \centering \voidbox ;
\end{verbatim}

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

\begin{verbatim}
\_codedecl \newif {Special if-macros, is-macros and loops <2020-05-22>} % preloaded in format
\_def \newif #1{\_ea \newifA \_string #1 \_relax #1}
\_ea \def \_ea \newifA \_string \_if #1 \_relax #2{\%}
\_sdef {#1true}{\_let#2=\_iftrue}%
\_sdef {#1false}{\_let#2=\_iffalse}%
\_let#2=\_iffalse
\_def \newifi #1{\_ea \newifiA \_string \_if #1 \_relax #2{\%}
\_sdef {#1true}{\_let#2=\_iftrue}%
\_sdef {#1false}{\_let#2=\_iffalse}%
\_let#2=\_iffalse
\_public \newif ;
\end{verbatim}

\$\texttt{\afterfi}$ \{\texttt{what to do}\} \{\texttt{ignored}\} $\texttt{\fi}$ closes condition by $\texttt{\fi}$ and processes \{\texttt{what to do}\}. Usage:

\begin{verbatim}
\texttt{\if<something> \afterfi{<result is true>} \else \afterfi{<result is false>} \fi}
\end{verbatim}

2.6.2 Loops

The \loop \ifsomething \codeA \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.

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

\foreach \langle list \rangle \do (parameter-mask) \{\langle what \rangle\} reads parameters from \langle list \rangle repeatedly and does \langle what \rangle for each such reading. The parameters are declared by \langle parameter-mask \rangle. Examples:

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

Note that \foreach \langle list \rangle \do {\langle what \rangle} is equivalent to \foreach \langle list \rangle \do #1{\langle what \rangle}.

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

\fornum \langle from \rangle \ldots \langle to \rangle \do {\langle what \rangle} or \fornumstep \langle num \rangle: \langle from \rangle \ldots \langle to \rangle \do {\langle what \rangle} repeats \langle what \rangle for each number from \langle from \rangle to \langle to \rangle (with step \langle num \rangle or with step one). The \langle what \rangle can include \#1 which is substituted by current number. The sequence \langle from \rangle \ldots \langle to \rangle can be decreasing too. The macro is expandable.
The \texttt{foreach} and \texttt{fornum} macros can be nested and arbitrary combined. When they are nested then use \texttt{##1} for the variable of nested level, \texttt{####1} for the variable of second nested level etc. Example:

\begin{verbatim}
\foreach ABC \do {\fornum 1..5 \do {letter:#1, number: ##1. }}
\end{verbatim}

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 \texttt{foreach} and \texttt{fornum}. The \_putforstack is used when \texttt{for*} is initialized and \_getforstack is used when the \texttt{for*} macro ends. The \_forlevel variable keeps the current nesting level. If it is zero, then we need not save nor restore any data.

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

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

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

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

\subsection{2.6.3 Is-macros}

There are a collection of macros \texttt{\isempty, \istokempty, \isequal, \ismacro, \isdefined, \isinlist} and \texttt{\isfile} with common syntax:

\begin{verbatim}
\issomething \{params\} \iftrue \{codeA\} \else \{codeB\} \fi
\item or
\issomething \{params\} \iffalse \{codeB\} \else \{codeA\} \fi
\end{verbatim}

The \texttt{\else} part is optional. The \texttt{\{codeA\}} is processed if \texttt{\issomething\{params\}} generates true condition. The \texttt{\{codeB\}} is processed if \texttt{\issomething\{params\}} generates false condition.

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

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

\begin{verbatim}
\isempty \{\texttt{\texttt{text}}\}\iftrue is true if the \{\texttt{\texttt{text}}\} is empty. This macro is expandable.
\istokempty \{\texttt{\texttt{tokens variable}}\}\iftrue is true if the \{\texttt{\texttt{tokens variable}}\} is empty. It is expandable.
\end{verbatim}
\_isempty \_istoksempty \isempty \_istoksempty ;

\_isempty \_istoksempty is true if the \textit{textA} and \textit{textB} are equal, only from string point of view, category codes are ignored. The macro is expandable.

\_isempty \_istoksempty ;

\_isempty \_istoksempty is true if macro is defined as \textit{text}. Category codes are ignored in this testing. The macro is expandable.

\_isempty \_istoksempty ;

\_isdefined \_isdefined is true if \textit{csname} is defined. The macro is expandable.

\_isdefined \_isdefined is true if \textit{csname} is defined. The macro is expandable.

\_isinlist \_isinlist is true if the \textit{text} is included the macro body of the \texttt{list}. The category code are relevant here. The macro is not expandable.

\_isinlist \_isinlist is true if the \textit{filename} exists and are readable by TeX.

\_isfile \_isfile is true if given font exists. The result of this testing is saved to the \_ifexistfam.

\_isnextchar \_isnextchar has different syntax than all others is-macros. It executes \textit{codeA} if next character is equal to \textit{char}. Else the \textit{codeB} is executed. The macro is not expandable.
2.7 Setting parameters

The behavior of document processing by Op\TeX is controlled by parameters. The parameters are

- primitive registers used in build-in algorithms of \TeX,
- registers declared and used by Op\TeX macros.

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

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

2.7.1 Primitive registers

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

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

Note that \_topskip and \_splitoppskip are changed when first \_typosize sets the main values (default font size and default \_baselineskip).

\begin{verbatim}
\_topskip=10pt % top edge of page-box to first baseline distance
\_splitoppskip=10pt
\end{verbatim}

42
2.7.2 Plain Te\TeX registers

Declared registers used in plain Te\TeX:

\begin{verbatim}
% We also define special registers that function like parameters:
\_newskip\_smallskipamount \_smallskipamount=3pt plus 1pt minus 1pt
\_newskip\_medskipamount \_medskipamount=6pt plus 2pt minus 2pt
\_newskip\_bigskipamount \_bigskipamount=12pt plus 4pt minus 4pt
\_newskip\_normalbaselineskip \_normalbaselineskip=12pt
\_newskip\_normallineskip \_normallineskip=1pt
\_newdimen\_normallineskiplimit \_normallineskiplimit=0pt
\_newdimen\_jot \_jot=3pt
\_newcount\_interdisplaylinepenalty \_interdisplaylinepenalty=100
\_newcount\_interfootnotelinepenalty \_interfootnotelinepenalty=100
\_def\_normalbaselines{\_lineskip=\_normallineskip
\_baselineskip=\_normalbaselineskip \_lineskiplimit=\_normallineskiplimit}
\_def\_frenchspacing{\_sfcode`\.=1000 \_sfcode`\?=1000 \_sfcode`\!=1000
\_sfcode`\:=1000 \_sfcode`\;=1000 \_sfcode`\,=1250 }
\_def\_nonfrenchspacing{\_sfcode`\.=3000 \_sfcode`\?=3000 \_sfcode`\!=3000
\_sfcode`\:=2000 \_sfcode`\;=1500 \_sfcode`\,=1250 }
\_public \_normalbaselines \_frenchspacing \_nonfrenchspacing
\_emergencystretch=20pt \_emergencystretch=20pt % we want to use third pass of aparagraph building algorithm
\_clubpenalty=10000 \_clubpenalty=150 % after first line of paragraph
\_widowpenalty=10000 \_widowpenalty=150 % before last line of paragraph
\_showboxbreadth=150 \_showboxbreadth=7 % for tracing boxes
\_showboxdepth=7 % errorcontextlines=15 \_errorcontextlines=15
\_tracinglostchars=2 \_tracinglostchars=2 % missing character warnings on terminal too
\_outputmode=1 \_outputmode=1 % PDF output
\_pdfvorigin=0pt \_pdfvorigin=0pt % origin is exactly at left upper corner
\_pdfhorigin=0pt \_pdfhorigin=0pt
\_hoffset=25mm \_hoffset=25mm % margins are 2.5cm, no 1in
\_voffset=25mm \_voffset=25mm
\_hsize=160mm \_hsize=210mm (from A4 size) - 2*25mm (default margins)
\_vsize=244mm \_vsize=297mm (from A4 size) - 2*25mm (default margins) -3mm baseline correction
\_pagewidth=210 true mm \_pagewidth=210 true mm
\_pageheight=297 true mm
\_pageheight=297 true mm
\_def\_plaintexsetting{\_emergencystretch=0pt \_emergencystretch=0pt % we need not to keep the compatibility with old documents
\_clubpenalty=150 \_clubpenalty=150
\_widowpenalty=150 \_widowpenalty=150}
\end{verbatim}

2.7.3 Different settings than in plain Te\TeX

Default “baseline setting” is for 10 pt fonts (like in plain Te\TeX). But \texttt{\textbackslash typsize} and \texttt{\textbackslash typscale} macros re-declare it if another font size is used.

The \texttt{\textbackslash nonfrenchspacing} is not set by default because the author of OpTe\TeX is living in the Europe. If you set \texttt{\textbackslash enlang} hyphenation patterns then \texttt{\textbackslash nonfrenchspacing} is set.

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

\begin{verbatim}
\_normalbaselines \_normalbaselines % baseline setting, 10 pt font size
\_emergencystretch=20pt \_emergencystretch=20pt % we want to use third pass of aparagraph building algorithm
\_clubpenalty=10000 \_clubpenalty=150 % after first line of paragraph
\_widowpenalty=10000 \_widowpenalty=150 % before last line of paragraph
\_showboxbreadth=150 \_showboxbreadth=7 % for tracing boxes
\_showboxdepth=7 \_errorcontextlines=15 \_tracinglostchars=2 % missing character warnings on terminal too
\_outputmode=1 \_outputmode=1 % PDF output
\_pdfvorigin=0pt \_pdfvorigin=0pt % origin is exactly at left upper corner
\_pdfhorigin=0pt \_pdfhorigin=0pt
\_hoffset=25mm \_hoffset=25mm % margins are 2.5cm, no 1in
\_voffset=25mm \_voffset=25mm
\_hsize=160mm \_hsize=210mm (from A4 size) - 2*25mm (default margins)
\_vsize=244mm \_vsize=297mm (from A4 size) - 2*25mm (default margins) -3mm baseline correction
\_pagewidth=210 true mm \_pagewidth=210 true mm
\_pageheight=297 true mm \_pageheight=297 true mm
\_def\_plaintexsetting{\_emergencystretch=0pt \_emergencystretch=0pt % we need not to keep the compatibility with old documents
\_clubpenalty=150 \_clubpenalty=150
\_widowpenalty=150 \_widowpenalty=150}
\end{verbatim}

If you insist on plain Te\TeX values of these parameters then you can call the \texttt{\textbackslash plaintexsetting} macro.

\begin{verbatim}
\_def\_plaintexsetting{\_emergencystretch=0pt \_emergencystretch=0pt % we need not to keep the compatibility with old documents
\_clubpenalty=150 \_clubpenalty=150
\_widowpenalty=150 \_widowpenalty=150}
\end{verbatim}
The main principle how to configure OpTeX is not to use only parameters. A designer can copy macros from OpTeX and re-define them as required. This is a reason why we don’t implement dozens of parameters, but we keep OpTeX macros relatively simple. Example: do you want another design of section titles? Copy macros \_printsec and \_printsecc from sections.opm file to your macro file and re-define them.

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

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

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

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

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

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

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

The \ttindent gives default indentation of verbatim lines printed by \begtt...\endtt pair or by \verbatiminput. The \ttshift gives the amount of shift of all verbatim lines to right. Despite to the \ttindent, it does not shift the line numbers, only the text.

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

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

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

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

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

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

Each line in the Table of contents is printed in a group. The \everytocline tokens list is processed here before the internal \tocitem⟨num⟩ macro which starts printing the line.
The \bibtexhook tokens list is used inside the group when \usebib command is processed after style file is loaded and before printing bib-entries. You can re-define a behavior of style file here or you can modify the more declaration for printing (fonts, baselineskip, etc.) or you can define a specific macros used in your .bib file.

\newtoks\bibtexhook
\public \bibtexhook ;
\newtoks\everycaptiont \newtoks\everycaptionf
\public \everycaptiont \everycaptionf ;

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

\newtoks\everyii
\public \everyii ;

The \mnote is used in the \mnote group before \noindent which immediately precedes marginal note text.
The \mnotesize is horizontal size of the marginal notes.
The \mnoteindent is horizontal space between body-text and marginal note.

\newtoks\everymnote
\newdimen\mnotesize \mnotesize=20mm % the width of the mnote paragraph
\newdimen\mnoteindent \mnoteindent=10pt % ditance between mnote and text
\public \everymnote \mnotesize \mnoteindent ;

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

\newtoks\everytable \newtoks\thistable
\newtoks\tabiteml \newtoks\tabitemr \newtoks\tabstrut
\newdimen\tablinespace \newdimen\vvkern \newdimen\hhkern \newdimen\tsize
\newskip\tabskipl \newskip\tabskipr
\everytable={} % code used after settings in \vbox before table processing
\thistable={} % code used when \vbox starts, is is removed after using it
\tabstrut=\{(\strut
\tabiteml=\{(\enspace
\tabitemr=\{(\enspace
\tablinespace=2pt % additional vertical space before/after horizontal rules
\vvkern=1pt % space between double vertical line and used in \frame
\hhkern=1pt % space between double horizontal line and used in \frame
\tabskip=\opt\relax % \tabskip used before first column
\tabskipr=\opt\relax % \tabskip used after the last column
\public \everytable \thistable \tabiteml \tabitemr \tabstrut \tablinespace
\vvkern \hhkern \tsize \tabskipl \tabskipr ;

The \eqalign macro can be configured by \eqlines and \eqstyle tokens lists. The default values are set in order this macro beahives like in Plain TeX. The \eqspace is horizontal space put between equation systems if more columns in \eqalign is used.
\_lmfil is “left matrix filler” (for matrix columns). The default value does centering because right matrix filler is directly set to \hfil.

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

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

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

### Parameters
\newdimen\headlinedist \headlinedist=14pt
\newdimen\footlinedist \footlinedist=24pt
\newskip\pgbottomskip \pgbottomskip=0pt \relax
\newtoks\ovalparams \newtoks\circleparams
\newtoks\nextpages

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

### \nextpages
\nextpages={\headline={\rmfixed \firstmark \hfil}}

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

### \pgbackground
\pgbackground={\Yellow \hrule height Opt depth\pdfpageheight width\pdfpagewidth}

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.
2.8 More OpTEX macros

The second bundle of OpTEX macros is here.

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

In order to achieve this, we declare \optexcatcodes catcode table and \plaintexcatacodes. 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 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, \restorectable pops the stacks and returns to the catcodes used before \opinput is run.

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.

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 \optexcatcodes table and sets it to the main catcode table 0.

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

If the \afterload macro is defined then it is run after \opinput. The catcode setting should be here. Note that catcode setting done in the loaded file is forgotten after the \opinput.
The declarator `\optdef\macro \[⟨opt default⟩\] ⟨params⟩{⟨replacement text⟩}` defines the \macro with the optional parameter followed by normal parameters declared in ⟨params⟩. The optional parameter must be used as the first first parameter in brackets [...]. If it isn’t used then ⟨opt default⟩ is taken into account. The ⟨replacement text⟩ can use \the\opt because optional parameter is saved to the \opt tokens register. Note the difference from L\AT\EX concept where the optional parameter is in #1. Op\TEX 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.

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

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

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

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

The verbatim scanner of \scantoeol keeps category 7 for ^ in order to be able to use ^\^J as comment chracter which means that the next line continues.
The \replstring macro{\langle textA\rangle}{\langle textB\rangle} replaces all occurrences of \langle textA\rangle by \langle textB\rangle in the \macro body. The \macro must be defined without parameters. The occurrences of \langle textA\rangle are not replaced if they are “hidden” in braces, for example \ldots{\ldots\langle textA\rangle\ldots}. The category codes in the \langle textA\rangle must exactly match.

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

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

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

You can use expandable \_bp\langle dimen\rangle convertor from TEX \langle dimen\rangle (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\langle expression\rangle for analogical purposes. It expands to the value of the \langle expression\rangle at expand processor level with \_decdigits digits after decimal point. The \langle expression\rangle can include +-*() and decimal numbers in common syntax.

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

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 \langle documentation\rangle (and \langle ignored text\rangle too) must be \langle balanced text\rangle. It means that you cannot document only the \{ but you must document the \} too.
2.9 Using key=value format in parameters

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

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

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

You can define \myframe as follows:

\def\myframedefaults{% defaults:
  frame-color=Black, % color of frame rules
  text-color=Black, % color 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
  \ea\addto\ea\myframedefaults\ea{\the\opt}%
  \readkv\myframedefaults
  \rulewidth=\kv{rule-width}
  \hhkern=\kv{margins}\vvkern=\kv{margins}\relax
  \kv{frame-color}\frame{\kv{text-color}\strut #1}%
  \egroup}

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

You can use keys without values in the parameters list too, but with additional care. For example,
suppose draft option without parameter. If user write \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/{\bgroup
  \ea\addto\ea\myframedefaults\ea{\the\opt}%
  \replstring\myframedefaults{draft}{DRAFTv=1}%
  \readkv\myframedefaults
  ...
  \ifnum\kv{DRAFTv}=1 draft mode\else normal mode\fi
  ...
}

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

\def\_readkv#1{\_ea\_def\_ea\_tmpb\_ea{#1}%
\_replstring\_tmpb{= }{=}\_replstring\_tmpb{ =}{=}\
\_replstring\_tmpb{, }{,}\_replstring\_tmpb{,,}{,}\
\_ea \_kvscan \_tmpb,,=,}
\_def\_kvscan #1#2=#3,\{_ifx#1,\_else \_sdef{\_kv:#1#2}{#3}\_ea\_kvscan\fi}
\_def\_kv#1\{_trycs{\_kv:#1}{\_kvunknown}\}
\_def\_kvunknown{???}
\public \readkv \kv ;
2.10 Plain TeX macros

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

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\_\}
```

The shortcuts \chardef \@one is not defined in OpTEX. Use normal numbers instead of such obscurities.

The \magstep and \magstephalf are defined with \space, (no \relax), in order to be expandable.

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

Plain TeX classical \obeylines and \obeyspaces.

Spaces. \thinspace, \negthinspace, \enspace, \enskip, \quad, \qquad, \smallskip, \medskip, \bigskip, \nointerlineskip, \offinterlineskip, \topglue, \vglue, \hglue, \slash.
Penalties macros: \break, \nobreak, \allowbreak, \filbreak, \goodbreak, \eject, \supereject, \dosupereject, \removelastskip, \smallbreak, \medbreak, \bigbreak.

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

Alignment. \hang \ialign \multispan.

Tabbing macros are omitted because they are obsolete. \textindent, \item, \itemitem, \narrower, \raggedright, \ttraggedright, \leavevmode.
Few character codes are set for backward compatibility. But old obscurities (from plain TeX) based on \mathhexbox are not supported – an error message and recommendation to directly using of the desired character is implemented by the \_usedirectly macro. The user can re-define these control sequences of course.

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

The accents commands like \v, \., \H, etc. are not defined. Use the accented characters directly – it is best solution. But you can use the macro \oldaccents which defines accented macros. Much more usable is to define these control sequences to other purposes.
The plain TeX macros \hrulefill, \dotfill, \rightarrowfill, \leftarrowfill, \downbracefill, \upbracefill. The last four are used in non-Unicde variants of \overrightarrow, \overleftarrow, \overbrace and \underbrace macros, see section 2.15.

\def \braceld#1#2{\leaders#1\hrulefill#2}
\def \bracerd#1#2{\leaders\hbox{$#1$}\hrulefill\hbox{$#2$}}
\def \bracelu#1#2{\leaders\vrule\ht0\depth.0\hrulefill\vrule\ht0\depth#1#2}
\def \braceru#1#2{\leaders\vrule\ht0\depth.0\hrulefill\vrule\ht0\depth#1#2}
\public \hrulefill \dotfill \rightarrowfill \leftarrowfill \downbracefill \upbracefill
2.11 Preloaded fonts for text mode

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

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

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

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

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:

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

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

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

\begin{verbatim}
  Here is 10 pt text, \setfontsize\{at12pt\} 10 pt text here unchanged...
  \resizethefont and 12 pt text is here.
\end{verbatim}
The \setfontsize command acts like font modifier. It means that it saves information about fonts but does not change the font actually until variant selector or \resizethefont is used.

The following example demonstrates the mag format of \setfontsize parameter. It is only a curious example probably not used in practical typography.
\def\smaller{\setfontsize{mag.9}\resizethefont}
Text \smaller text \smaller text \smaller text.

2.12.2 The \font primitive

If you load a font directly by \font primitive and you want to create a size-dependent selector for such font then you can use \resizethefont:
\font\tencomfortaa=Comfortaa-Regular-T1 at10pt 
\def\comfortaa{\tencomfortaa\resizethefont}

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

The example above uses the 8-bit tfm font. You can use Unicode font too, of course. The \fontfam macro initializes the extended \font primitive features for LuaTeX (see section 2.13.14). If you didn’t use this command, you must to initialize these features by the \initunifonts command explicitly, for example:
\initunifonts
\font\tencyklop=[cyklop-regular] at10pt % the font cyklop-regular.otf is loaded 
\def\cyklop{\tencyklop\resizethefont}

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

2.12.3 The \fontdef declarator

You can declare \langle newfont \rangle by the \fontdef command.
\fontdef \langle newfont \rangle \{\langle font modifiers \rangle \langle variant-selector \rangle} 
example:
\fontdef \bigfont \{\setfontsize{at15pt}\bf}

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

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

The parameter of the \fontdef macro must be exactly finished by the variant selector. More information about font modifiers and variant selectors are in the section 2.13.

2.12.4 The \fontlet declarator

We have another command for scaling: \fontlet which is able to resize arbitrary font given by its font switch. This font switch was declared it by the \font primitive or the \fontdef macro.
\fontlet \langle newfont \rangle = \langle fontswitch \rangle \langle sizespec \rangle
element:
\fontlet \bigfont = \_tenbf at15pt

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

Summary: you can declare font switches:
• by the \fnt primitive if you know the font file,
• by the \fontlet command if you know the font switch and the size, or
• by the \fontdef command if you know the variant and modifiers.
2.12.5 Optical sizes

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

Font collections with optical sizes must be registered by the \regtfm for tfm files or \regoptsizes
for Unicode fonts. \TeX{} registers 8bit Latin Modern fonts in the format (fonts-resize.opm file) and
OTF Latin Modern fonts in the \lfmfonts.opm file.

2.12.6 Implementation notes

The \setfontsize \{\sizespec\} saves the \sizespec to the \_sizespec macro. The \_optsize value is
calculated from the \sizespec. If the \sizespec 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.

The \_setsizespec macro is used. It works in TFM mode (\doresizetfmfont) or OTF mode (\doresizeunifont). In both
modes, it does

\_font \_tenXX = \{\fontname\} \_sizespec

The \fontname is generated by the \fontname \TeX{} primitive where \_rfontskipat removes the
at⟨dimen⟩ part of the \fontname output. The \fontname is generated differently in OTF mode, see
\doresizeunifont macro. In both modes, it does

The \whatresize is defined as \{\variant-name\}.

\def\doresizefont#1#2{\edef\whatresize{#1} \ifx \fontselector \undefined \doresizefont#2\else \ea \fontselector \fi
\let\_tryload#1 \relax \lastmagsize=\_zo \_slet{\_tryload#1}{\relax}}
\let\doresizefont=\doresizetfmfont
\def\logfont#1{} % default is no logging of used fonts
\fontdef \{font switch\}\{\{modifiers\}\{variant selector\}\} opens group, runs \{modifiers\}\{variant selector\} (i.e. it runs #2 parameter). The font switch #1 saved in the \_fontselector macro is re-declared because the variant selector runs the \_resizefont. Now, we need to keep the current meaning of the font switch #1 but we must leave the opened group. This is done by the \_keepmeaning macro.

\fontlet \{font switch A\} \{font switch B\} \{size spec\} does

The \{fontname\} is extracted using the primitive command \_fontname \{font switch B\}.

\newcurrfontsize \{size spec\} sets current font size to the \{size spec\}. It is implemented by \fontlet. The font switch of the current font is extracted by \_the\_font. We must re-create the control sequence \_the\_font because its original meaning is set to “inaccessible” by \font primitive is started.

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

\currvar is defined by \protected\def\currvar\{\cs{_currvar:}\ea\csstring \the\font\}\. This meaning is activated by the \_reloading macro.

The font selection system allows to use \currvar instead explicitly specified variant selector. The current variant is extracted from \the\font output which could be \_tenXX control sequence. Then \currvar expands to \_rm or \_it etc.
The \_regtfm ⟨font id⟩ ⟨optical size data⟩ saves the ⟨optical size data⟩ concerned to ⟨font id⟩. The ⟨optical size data⟩ is in the form as shown below in the code where \_regtfm is used.

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

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

Optical sizes data for preloaded 8bit Latin Modern fonts:

```

def\regtfm lmr 0 #2 *{\def \csname _#1:reg\endcsname{#2 16380 \relax}}
\def\tmpa{#1}\reversetfm #2 *

def\reversetfm #1 #2 {% we need this data for \setmathfamily
 \let \csname _\tmpa:reg\endcsname
 \if*#2\else \ea\reversetfm \fi}

def\whichtfm #1{%
 \ifcsname _#1:reg\endcsname
 \ea\ea\ea\dowichtfm \csname _#1:reg\endcsname
 \else
  #1
 \fi
}

def\dowichtfm #1 #2 {%
 \ifdim \optsize<#2pt #1\ea\ignoretfm \else \ea\dowichtfm \fi
}

def\ignoretfm #1\relax{}
```

Optical sizes data for preloaded 8bit Latin Modern fonts:

```
\regtfm lmr 0 ec-lmr5 5.5 ec-lmr6 6.5 ec-lmr7 7.5 ec-lmr8 8.5 ec-lmr9 9.5
\regtfm lmbx 0 ec-lmbx5 5.5 ec-lmbx6 6.5 ec-lmbx7 7.5 ec-lmbx8 8.5 ec-lmbx9 9.5
\regtfm lmri 0 ec-lmri7 7.5 ec-lmri8 8.5 ec-lmri9 9.5 ec-lmri10 11.1 ec-lmri12 *
\regtfm lmtt 0 ec-lmtt8 8.5 ec-lmtt9 9.5 ec-lmtt10 11.1 ec-lmtt12 *
\SetFontSize {at10pt} % default font size
```

### 2.13 The Font Selection System

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

#### 2.13.1 Terminology

We distinguish between

- **font switchers**, they are declared by the \font primitive or by \fontlet or \fontdef macros, they select given font.
- **variant selectors**, there are four basic variant selectors \rm, \bf, \it, \bi, there is a special selector \currvar. More variant selectors can be declared by the \famvardef macro. They select the font depending on given variant and on the font context (i.e. on current family and on more features given by font modifiers). In addition, OpTeX defines \tt as variant selector independent of chosen font family. It selects typewriter-like font.
- **font modifiers** are declared in a family (\cond, \caps) or are “build in” (\SetFontSize{⟨size spec⟩}, \SetFont{⟨features⟩}). They do appropriate change in the font context but do not select the font.
- **family selectors** (for example Terms, 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 selected font and the \textit{font context} are returned back to the values used when the group was opened. They have the following features:

- The \textit{font context} is a set of macro values which will affect the selection of real font when variant selector is processed. It includes the value of \textit{current family}, current font size and more values stored by font modifiers.

- The \textit{family context} is the current family value stored in the font context. The variant selectors declared by \texttt{\textbackslash famvardef} and font modifiers declared by \texttt{\textbackslash moddef} are dependent on the \textit{family context}. They can have the same names but different behavior in different families.

The fonts registered in Op\TeX\ have their macros in the \textit{font family files}, each family is declared in one font family file with the name \texttt{f-famname.opm}. All families are collected in \texttt{fams-ini.opm} and user can give more declarations in the file \texttt{fams-local.opm}.

### 2.13.2 Font families, selecting fonts

The \texttt{\fontfam \[\langle Font Family\rangle\]} opens the relevant font family file where the \langle Font Family\rangle 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 \texttt{\textbackslash rm} variant selector is processed.

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

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

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

\langle family selector\rangle \langle font modifiers\rangle \langle variant selector\rangle

For example:

\begin{verbatim}
\fontfam [Heros] % Heros family is active here, default \texttt{\textbackslash rm} variant.
\fontfam [Termes] % Termes family is active here, default \texttt{\textbackslash rm} variant.
\{\Heros \caps \cond \it The \caps+condensed italics in Heros family is here.\}
\textsl{The Termes roman is here.}
\end{verbatim}

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

You can use the \texttt{\setfontsize \{\langle sizespec\rangle\}} command in the same sense as other font modifiers. It saves information about font size to the font context. See section 2.12. Example:

\begin{verbatim}
\texttt{\textbackslash rm} default size \setfontsize{at14pt}\texttt{\textbackslash rm} here is 14pt size \textit{\it italic} is in 14pt size too \textbf{\bf bold} too
\end{verbatim}

Much more comfortable way to resize fonts is using OPmac-like commands \texttt{\textbackslash typosize} and \texttt{\textbackslash typoscale}. These commands prepare the right sizes for math fonts too and they re-calculate many internal parameters like \texttt{\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 Op\TeX\ format then you can use command \texttt{\noloadmath} before you load a first font family.

If you want to use your specially selected Unicode-math font then use \texttt{\loadmath \{\langle font_file\rangle\}} or \texttt{\loadmath \{\langle font_name\rangle\}} before first \texttt{\fontfam} is used.
2.13.4 Declaring font commands

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

- Font switches can be declared by \font primitive (see section 2.12.2) or by \fontlet command (see section 2.12.4) or by \fontdef command (see sections 2.13.5 and 2.12.3). When the font switches are used then they select given font independently of 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 \ttt selector is documented in section 2.13.7.
- Font modifiers are “build in” or declared by \moddef command. They do modifications in the font context but don’t select any font.
  - “build-in” font modifiers are \setfontsize (see section 2.12), \setff (see section 2.13.9), \setfontcolor, \setletterspace and \setwordspace (see section 2.13.10). They are independent of font family.
  - Font modifiers declared by \moddef depend on the font family and they are typically declared in font family files, see section 2.13.11.
- Family selectors set the given font family as current and re-set data used by 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 arbitrary by \def primitive by user. See an example in section 2.13.8.

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

2.13.5 The \fontdef declarator in detail

The general format for \fontdef usage is

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

where \{family selector\} and \{font modifiers\} are optional and \{variant selector\} is mandatory.

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

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

2.13.6 The \famvardef declarator

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

\famvardef\{new variant selector\} \{family selector\} \{font modifiers\} \{variant selector\}

where \{family selector\} and \{font modifiers\} are optional and \{variant selector\} is mandatory. The \{new variant selector\} should be used in the same sense as \rm, \bf etc. It can be used as the final command in next \fontdef or \famvardef declarators too. When the \{new variant selector\} is used in normal text then it does 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 warning is printed on the terminal “\{var selector\} is undeclared in current family“ and nothing happens. But you can declare the same variant selector by \famvardef macro in the context of new family. Then the same command 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:
Now, you can use six independent variant selectors \rm, \bf, \it, \bi, \mf and \mi in the selected font family.

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

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

\begin{verbatim}
\fontfam[Heros] \fontfam[Termes]
\def\exhcorr\{\setfontsize{mag.88}}
\famvardef\rmsans\{\Heros\exhcorr\rm\}
\famvardef\itsans\{\Heros\exhcorr\it\}
\end{verbatim}

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 name space can be here. For example: if \mf is defined by user and then \fontfam[Roboto] is used then \famvardef\mf is performed for Roboto family and the original meaning of \mf is lost. But \TeX prints warning about it. There are two cases:

\begin{verbatim}
\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.
\end{verbatim}

2.13.7 The \tt variant selector

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

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

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

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

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 special feature only for \tt selector. New definition is used in \_ttfont and \_urlfont too. It is recommended to use \setff{-liga;-tlig} in order 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. User can define them.

The following example shows how to define a “title-font selector”. Titles are not only bigger but they are typically in bold variant. When the 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:
OpTeX defines similar internal commands \titfont, \chapfont, \secfont and \seccfont, see section 2.26. The commands \typsize 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 supports given font feature then font is reloaded without warning nor error, silently. The font feature is not activated.

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

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

The \setfontsize command is described in the section 2.12. The \setff command was described in previous subsection.

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

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

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

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

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

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

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

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

2.13.11 How to create the font family file

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

```latex
\f-heros.opm

\famdecl [Heros] \Heros {TeX Gyre Heros fonts based on Helvetica}
\{(\caps \cond) \{\rn \bf \it \bl\} \{FiraMath\}
\{[texgyreheros-regular]\}
\{\def\fontnamegen{[texgyreheros-\condV-\currV]\_capsV\_fontfeatures}\}
\wlog{\detokenize{\
\caps ...... caps & small caps```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````}

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

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

```latex
\famdecl \{(Name of family) \{Familyselector\} \{\{comments\}\}\{\{modifiers\}\}\{\{variant selectors\}\}\{\{comments about math fonts\}\}\{\{font-for-testing\}\}\{\{font name or font file name generated\}\}\}
```

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

- You can use \wlog{\detokenize{...} to write additional information into log file.

- You can declare optical sizes using \regoptsizes if there are more font files with different optical sizes (like in Latin Modern). See f-lmfonts.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 does not produce any space in horizontal mode. For example check: X\caps Y, the letters XY must be printed without any space.

- Optionally, declare new variants by the \famvardef macro.

- Run \initfontfamily in order to start the family (it is mandatory).

- If math font should be loaded, use \loadmath{\{math font\}}.

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

```latex
\font \{\{selector\}\}\{\{\fontnamegen\}\}\{\{size spec\}\}
```

Note that the extended \font syntax \font{\{selector\}\{\{font name\}:\{font features\}\}\{\{size spec\}\} or \font{\{\\{selector\\}\{\{font file name\}:\{font features\}\}\{\{size spec\}\} is expected here.
Example 1
Assume an abstract font family with fonts `xx-Regular.otf`, `xx-Bold.otf`, `xx-Italic.otf` and `xx-BoldItalic.otf`. Then you can declare the \resetmod (for initialising 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;+onum} and \def \_condV{cn} were processed by these font modifiers. If user needs the \bf variant at 11pt now then the
```
\font {texgyreheroscn-bold:+smcp;+onum;+tlig;} at11pt
```
is processed. We assume that a font file texgyreheroscn-bold.otf is present in your \TeX{} system.

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 font modifier) then it is re-defined with 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 modifiers declaration. It runs the \{Familyselector\} and it runs \_rm, so first font from new family is loaded and it is ready to use it.

**Name conventions**

Create font modifiers, new variants and the \{Familyselector\} only as public, i.e. in user name space without _ prefix. We assume that if user re-defines them then he/she needs not them, so we have no problems. If user’s definition done before loading font family file is re-defined then \TeX{} warns about it. See the end of section 2.13.4.

The name of \{Familyselector\} should begin with uppercase letter.

Please, look at Op\TeX{} font catalogue before you will create your own font family file and use the same names for analogical font modifiers (like \cond{}, \caps{}, \sans{} 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 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 another modifiers or macros, for example), use the name \_wordM{}. You can be sure that such name does not influence the private name space used by \TeX{}.

**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 a documentation about it.

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

### 2.13.12 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 \internaltemplate{} 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 \internaltemplate{} by the \regoptsizes macro. The syntax is shown for one real case:

```
\regoptsizes lmr.r lmrroman?-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 \internaltemplate{} \general-output-template{} \resizing-data{}
```

Suppose our example above. Then \optname{lmr.r} expands to lmrroman?-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 \texttt{11.1 < \optsize \leq 15} then 12 is substituted instead question mark. The \resizing-data{} virtually begins with zero <0, but it is not explicitly written. The right part of \resizing-data{} must be terminated by <*> which means "less than infinity".

If \optname gets an argument which is not registered \internaltemplate{} then it expands to \failedoptname{} which typically ends to error message about missing font. You can redefine \failedoptname{} macro to some existing font if you find it useful.

We are using a special macro \LMregfont{} in \f-lmfonts.opm. It sets the file names to lowercase and enables to use shortcasts instead real \resizing-data{}. There are shortcats \regoptFS{}, \regoptT{}, etc.
etc. here. The collection of \textit{internal-templates} are declared, each of them covers a collection of real file names.

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

The \texttt{\_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 \langle internal-templates\rangle and do an alias from each \langle internal-template\rangle to \langle real-font-name\rangle. For example, you can do it as follows:

\begin{verbatim}
\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}
\end{verbatim}

Another example of font family with optical sizes is Antykwa Półtawskiego. The optical sizes feature is deactivated by default and it is switched on by \texttt{\_size} font modifier:

\begin{verbatim}
\famdecl [Poltawski] [Antykwa Półtawskiego, Polish traditional font family]
\{\light \noexpd \expd \expd \cond \ccond \osize \caps\} {\rm \bf \it \bi} {}
\{\[antpolt-regular\]\}
{\def \\fontnamegen {\[antpolt\]\_lv\\\_condV\\\_currV}\_capsV\_\fontfeatures}
\_wlog {\_detokenize{
Modifers:``
``\light ..... light weight, \bf,\bi=semibold``
``\noexpd .... no expanded, no condensed, designed for 10pt size (default)``
``\expd ..... expanded, designed for 6pt size``
``\cond ...... semi expanded, designed for 8pt size``
``\ccond ..... condensed, designed for 12pt size``
``\osize ...... auto-sitches between \ccond \cond \noexpd \expd \by size``
\caps ...... caps & small caps``
}}
\_moddef \resetmod {\_fsetV li={},cond={},caps=} \_fvars regular bold italic bolditalic 
\_moddef \light {\_fsetV li=lt }
\_moddef \noexpd {\_fsetV cond={} }
\_moddef \expd {\_fsetV cond=semiexpd }
\_moddef \cond {\_fsetV cond=cond }
\_moddef \ccond {\_fsetV cond=cond }
\_moddef \caps {\_fsetV caps=+smcp;+onum; }
\_moddef \nocaps {\_fsetV caps={} }
\_moddef \osize {\def \_fontnamegen{\[antpolt\]\_lv\\_optname(s)\-\_currV}\_capsV\_\fontfeatures}
\_regoptsizes x ? expd <7 semiexpd <9 {<11.1 semicond <15 cond <*>}
\_initfontfamily % new font family must be initialized
\end{verbatim}

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{} is able to see it in your filesystem then you can type \texttt{\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 \texttt{\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 \texttt{\fontfam [LM Fonts]} is equivalent to \texttt{\fontfam [LMfonts]} and both commands load the file f-lmfonts.opm.

You can use your font file in sense of previous paragraph without registering it. But problem is that such families are not listed when \texttt{\fontfam[]} is used and it is not included in font catalogue when

68
\fontfam[catalog] is printed. The list of families taken in the catalogue and listed on the terminal is declared in two files: \fams-ini.opm and \fams-local.opm. The second file is optional. User can create it and write to it the information about user-defined families using the same syntax as in existed file \fams-ini.opm.

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

The \fams-ini.opm and \fams-local.opm files use the macros \_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 \_famtext {Special name for printing a catalogue:}
5 \_faminfo [Catalogue] {Catalogue of all registered font families} {fonts-catalog} {} 
6 \_famtext {Computer Modern like family:}
7 \_faminfo [Latin Modern] {TeX Gyre fonts based on Computer Modern} {f-lmfonts} 
8 { -,\nbold,\sans,\sans\nbold,\slant,\ttset,\ttset\slant,\ttset\caps,\%
9 \ttprop,\ttprop\bolder,\quotset: {\rm\bf\it\bi}
10 \caps: {\rm\it}
11 \ttlight,\ttcond,\dunhill: {\rm\it} \upital: {\rm} }
12 \_famalias {LMfonts} \_famalias {Latin Modern Fonts} \_famalias {lm}
13 \_famtext {TeX Gyre fonts based on Adobe 35:}
14 \_faminfo [Termes] {TeX Gyre Termes fonts based on Times} {f-termes}
15 { -,\caps: {\rm\bf\it\bi} }
16 \_famalias {Times}
17 \_faminfo [Heros] {TeX Gyre Heros fonts based on Helvetica} {f-heros}
18 { -,\caps,\cond,\caps\cond: {\rm\bf\it\bi} }
19 \_famalias {Helvetica}
```

... etc.

The \_faminfo command has the syntax:

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

The \{mod-plus-vars\} data is used only when printing catalogue. It consists with one or more pairs \{mods\} : \{vars\}. For each pair: each modifiers (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 to 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 in order to print \_famfrom info into the catalogue. The \_famfrom data is applied to each following declared families until new \_famfrom is given. Use \_famfrom {} if the information is not known.

2.13.14 Notices about extension of \font primitive

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

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

The OpTeX format is initialized by luatex engine by default but you can initialize it by luahbtex engine too. Then the harfbuzz library is ready to use for font rendering as an alternative of build-in
font renderer from Luaotfload. The harfbuzz library gives more features for rendering indic and arabic scripts. But it is not used as default, you need to specify `mode=harf` in 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 luahbtex engine is not used then `mode=harf` is ignored. See Luaotfload documentation for more information.

2.13.15 Implementation of the Font Selection System

\famdecl \[\text{Family Name}\] \[\text{Famselector}\] \{\langle comment\rangle\} \{\langle modifiers\rangle\} \{\langle variants\rangle\} \{\langle math\rangle\} \{\langle font for testing\rangle\} \{\text{\texttt{\fontnamegen}}\} \{\text{\texttt{\mathfaminfo}}\} runs \texttt{\initunifonts}, then checks if \texttt{\langle Famselector\rangle} is defined. If it is true, then closes the file by \texttt{\endinput}. Else it defines \texttt{\langle Famselector\rangle} and saves it to the internal \texttt{\_f:\langle currfamily\rangle:main.fam} command. The macro \texttt{\initfontfamily} needs it. The \texttt{\_currfamily} is set to the \texttt{\langle Famselector\rangle} because the following \texttt{\moddef} commands need to be in the right font family context. The \texttt{\_currfamily} is set to the \texttt{\langle Famselector\rangle} by the \texttt{\langle Famselector\rangle} too, because \texttt{\langle Famselector\rangle} must set the right font family context. The font family context is given by the current \texttt{\_currfamily} and by the actual meaning of the \texttt{\fontnamegen} macro. The \texttt{\mathfaminfo} is saved for usage in the catalogue.
The `\moddef \{modifier\} \{data\}` simply speaking does `\def\{modifier\}\{\{data\}\}`, but we need to respect the family context. In fact, `\protected\def\f:\{current\\ family}:\{modifier\}\{\{data\}\}` is performed and the `\{modifier\}` is defined as \_famdepend\{modifier\} \_f: \_currfamily:\{modifier\}. It expands to \_f:\_currfamily:\{modifier\} value if it is defined or it prints warning. When the `\_currfamily` value is changed then we can declare the same \_f:\_currfamily:\{family\}.

When user declare a prefixed variant of the `\{modifier\}` then unprefixed modifier name is used in internal macros, this is reason why we are using the `\_remifirstunderscore` \_tmp (where \_tmp expands to something or to something). The `\remifirstunderscore` redefines \_tmp in the way that it expands only to something without the first _.
The $\_addtomodlist$ \{\textit{font modifier}\} adds given modifier to $\_\modlist$ macro. This list is used after $\resetmod$ when new family is selected by a family selector, see $\_\resetfam$ macro. This allows to reinitialize the same current modifiers in the font context after family is changed.

\begin{verbatim}
\def \moddef #1#2{\edef \_tmp{\csstring#1}\%\edef \tmp\moddef \moddef \_reloading}\%
\protected \edef \_tmp\ama\_noexpand\_famdepend\_noexpand\_fam\_noexpand\_currfamily\_noexpand\_tmp}{\_noexpand\_famdepend\_noexpand\_currfamily\_noexpand\_tmp}\%
\setnewmeaning \#1=\_tmp \moddef \}
\protected \def \_resetmod \{\cs\fam\_amma\fam\_resa\_m\_m\_t\_modf\}
\def \_resetfam\{\def \_addtomodlist\#1\#2\_reloading\%\}
\protected \edef \_modlist\#1\moddef \}
\let \_addtomodlist=\_addtomodlistb\%
\def \_currfamily{} \% default current family is empty
\def \_modlist{} \% list of currently used modifiers
\def \_addtomodlist\#1{\addtomodlist \#1}
\let \_addtomodlistb=\_addtomodlist\%
\def \_famdepend\#1\#2{\ifcsname \#2\_endcsname \csname \#2\_ama\_endcsname \else \ifx \_addtomodlist \addtomodlistb \opwarning{\string \#1 is undeclared in family \_currfamily, ignored}\fi \fi \}
\def \setnewmeaning \#1=\#2{\ifx \#1 \undefined \else \ifx \#1 \_tmpa \opwarning{\string \#1 is redefined by \string \#2} \fi \fi \let \#1=\_tmpa \}
\protected \def \_tryload:\#1\_\ten\#2\_\tryload:\#1\_\ten\#2\{\fontdef \_ten\#1\{\#2\}}\%
\protected \def \_currvar:_\ten\#1\{\#1\}
\public \moddef ;
\end{verbatim}

The $\famvardef \{\langle data\rangle\}$ uses analogical trick like $\moddef$ with the $\_famdepend$ macro. The auxiliary $\famvardefA \{\langle data\rangle\}$ \ten\langle data\rangle\_tryload\langle data\rangle uses it. It does:

- $\def \_tryload:\langle \textit{currfam}\rangle:\langle \textit{data}\rangle\{\fontdef \_ten\langle \textit{data}\rangle\{\langle \textit{data}\rangle\}}\}$ loads font \ten\langle \textit{data}\rangle,
- $\protected \def \langle \textit{data}\rangle\{\_famdepend \langle \textit{data}\rangle\{\_f:\langle \textit{currfam}\rangle:\langle \textit{data}\rangle\}}\}$,  
- $\def \_f:\langle \textit{currfam}\rangle:\langle \textit{data}\rangle\{\_tryload:\langle \textit{currfam}\rangle:\langle \textit{data}\rangle\} \_\ten\langle \textit{data}\rangle\} \_\currvar\_\ten\langle \textit{data}\rangle\}$ keeps family dependent definition,  
- $\def \_currvar:\_\ten\langle \textit{data}\rangle\{\langle \textit{data}\rangle\}$ in order to the $\currvar$ macro work correctly.

$\famvardef\tt$ behaves somewhat differently: it doesn’t re-define the $\tt$ macro which is defined as $\_\tryloadtt\_\ten\tt$ in sections 2.14 and 2.16.2. It only re-defines the internal $\_\tryloadtt\tt$ macro.

\begin{verbatim}
\def \famvardef\#1\{\edef \_tmp\csstring\#1\%\edef \tmp\famvardef \famvardef \_reloading\%\%
\edef \_famvardef\#1\#2\#3\#4{% \#1=\X\#2=\_ten\#2\#3=\_\tryload\_\ten\#3\#4=\data\%
\_isinlist{.\_\bf\_\it\_\bi\currvar\_\currvar}\#1\_\ifttrue\%
\opwarning{\string \famvardef: \You cannot re-declare standard variant selector \string \#1}\%
\_\else\%
\_\def\#3\{\_\fontdef\#2\#4\%\%
\protected \edef \_tmp\ama\_noexpand\_famdepend\_noexpand\_fam\_noexpand\_currfamily\_noexpand\_tmp\ ama\_noexpand\_tmp\ ama\_noexpand\_fam\_noexpand\_currfamily\_noexpand\_tmp\%
\setnewmeaning \#1=\_tmp \famvardef \_\fi\%
\_\def\_\currvar:\_\currvar\_\csstring\#2\{\\}
\_\fi\%
\_\let\#1=\_\tmpa\%
\protected \def \_famvardef\#1\{\_famvardef\#1\}\%
\public \famvardef ;
\end{verbatim}

The $\fontfam \{\langle Font Family\rangle\}$ does:

- Convert its parameter to lower case and without spaces, e.g. \langle fontfamily\rangle.
- If the file \langle fontfamily\rangle\_opm exists read it and finish.
- Try to load user defined \fams-local\_opm.
- If the \langle fontfamily\rangle is declared in \fams-local\_opm or \fams-ini\_opm read relevant file and finish.
- Print the list of declared families.
The `fams-local.opm` is read by the `_tryloadfamslocal` macro. It sets itself to `_relax` because we need not to load this file twice. The `_listfamnames` macro prints registered font families to the terminal and to the log file.

```
\def\fontfam[#1]{}% 
\lowercase{\edef\famname{\ea\removespaces #1 {} }}% 
\isfile {f-\famname.opm}\_iftrue \opinput {f-\famname.opm}\
\else 
  \tryloadfamslocal 
  \edef\famname{\trycs{\famf:\famname}()}% 
  \ifx\famfile\empty \listfamnames 
  \else \opinput {\famfile.opm}% 
  \fi fi 
}
\def\tryloadfamslocal{% 
  \isfile {fams-local.opm}\_iftrue 
  \opinput {fams-local.opm}\famfrom={}% 
  \fi 
  \let \tryloadfamslocal=\relax % need not to load fams-local.opm twice 
}
\def\listfamnames {%  
\wterm{===== List of font families ======}  
\begingroup  
\let\famtext=\wterm  
\def\faminfo [##1]##2##3##4{%  
\wterm{ \space\noexpand\fontfam [##1] -- ##2}  
\let\famalias=\famaliasA}  
\opinput {fams-ini.opm}%  
\isfile {fams-local.opm}\_iftrue \opinput {fams-local.opm}\_fi 
\_message{^^J}% 
\endgroup 
}
\def\famalias [##1]{% 
\wterm{ \space\space\space\space -- alias:} 
\def\famf:##1{\famfile} \famalias 
}
\newtoks\famfrom 
\input fams-ini.opm 
\let\famfile=\undefined 
\famfrom={}
```

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:{{\famname}} {\famfile} 
```

prints information on the terminal.

`\famalias` [[Family Alias]] does `\def \famf:{{\famalias}} {\famfile}` where `\famfile` 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.

```
\def\famtext #1{}% 
\def\faminfo [##1]##2##3##4{%  
\lowercase{\edef\tmp{\ea\removespaces #1 {} }}% 
\sdef{\famf:\tmp}{#3}% 
\def\famfile{#3} 
} 
\def\famalias [##1]{% 
\lowercase{\edef\famname{\ea\removespaces #1 {} }}% 
\sdef{\famf:\famname\ea}{\ea\famfile}% 
} 
\newtoks\famfrom 
\input fams-ini.opm 
\let\famfile=\undefined 
\famfrom={}
```

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

```latex
\newtoks \catalogsample
\newtoks \catalogmathsample
\newtoks \catalogonly
\newtoks \catalogexclude
\catalogsample={ABCDabcd Qsty fi fl áéíóúüů řžč ÁÉÍÓÚ ŘŽČ 0123456789}
\public \catalogonly \catalogexclude \catalogsample \catalogmathsample
```

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

```latex
\_def \_defaultfontfeatures {+tlig;}
\_def \_setff #1{%
    \_ifx^#1^\_let \_fontfeatures=\_defaultfontfeatures
    \else \edef \_fontfeatures{\_fontfeatures #1;}\_fi
    \reloading
}
\_setff {} % default font features: +tlig;
\_def \_removefeature #1{%
    \isinlist \_fontfeatures{#1}\_iftrue
    \_def \_tmp ##1#1##2;##3\_relax{\_def \_fontfeatures{##1##3}}%
    \_ea \_tmp \_fontfeatures \_relax
    \_fi
}
\_public \setff ;
```

The `\setfontcolor` and `\setletterspace` are macros based on the special font features provided by LuaTeX (and by XeLaTeX too but it is not our business). The `\setwordspace` recalculates the `\fontdimen2,3,4` of the font using the `\_doresizeunifont` macro which is used by the `\font` primitive (with independent `\fontdimen` registers).

```latex
\_def \_savedfontcolor{}
\_def \_savedletterspace{}
\_def \_savedwsp{}
\_def \_setfontcolor #1{\_removefeature{color=}\_edef \_tmp{\_calculatefontcolor{#1}}\_ifx \_tmp \_empty \_else \_edef \_fontfeatures{\_fontfeatures color=\_tmp;}\_fi
    \reloading
}
\_def \_setletterspace #1{\_removefeature{letterspace=}\_if^#1^\_else \_edef \_fontfeatures{\_fontfeatures letterspace=#1;}\_fi
    \reloading
}
\_def \_setwordspace #1{\_if^#1^\_def \_setwsp##1{}\_removefeature{+Ws}\
    \_else \_def \_setwsp\_setwspA{#1}\_setff{+Ws}\_fi
    \reloading
}
\_def \_setwspA #1#2{\_fontdimen2#2=#1\_fontdimen2#2
    \_fontdimen3#2=#1\_fontdimen3#2\_fontdimen4#2=#1\_fontdimen4#2}
\_def \_calculatefontcolor#1{\_trycs{fcolor:#1}{#1}} % you can define more smart macro ...
\_sdef{fcolor:red}{FF0000FF} \_sdef{fcolor:green}{00FF00FF} \_sdef{fcolor:blue}{0000FFFF}
\_sdef{fcolor:yellow}{FFFF00FF} \_sdef{fcolor:cyan}{00FFFFFF} \_sdef{fcolor:magenta}{FF00FFFF}
\_sdef{fcolor:white}{FFFFFFFF} \_sdef{fcolor:grey}{00000080} \_sdef{fcolor:lgrey}{00000025}
\_sdef{fcolor:black}{} % ... you can declare more colors...
\_public \setfontcolor \setletterspace \setwordspace ;
```
2.14 Preloaded fonts for math mode

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

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

The general rule of usage the math fonts in different sizes in OpTeX says: set three sizes by the macro \setmathsizes \[⟨text-size⟩/⟨script-size⟩/⟨scriptscript-size⟩\] and then load all math fonts in given sizes by \normalmath or \boldmath macros. For example \setmathsizes[12/8.4/6]\normalmath ... math typesetting at 12 pt is ready.

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

\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.

\chardef\bffam = 8
\chardef\bifam = 9
%\chardef\ttfam = 10
%\chardef\itfam = 11
\protected\def \_rm {\tryloadrm \tenrm \fam0 }
\protected\def \_bf {\tryloadbf \tenbf \fam\bffam}
\protected\def \_it {\tryloadit \tenit \fam1 }
\protected\def \_bi {\tryloadbi \tenbi \fam\bifam}
The optical sizes of Computer Modern fonts, AMS and other fonts are declared here.

\#% CM math fonts, optical sizes:
\_regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
cmmi10 11.1 cmmi12 *
\_regtfm cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
\_regtfm cmtex 0 cstex8 8.5 cstex9 9.5 cstex10 *
\_regtfm cmsy 0 cmsy5 5.5 cmsy6 6.5 cmsy7 7.5 cmsy8 8.5 cmsy9 9.5 cmsy10 *
\_regtfm cmbsy 0 cmbsy5 5.5 cmbsy6 6.5 cmbsy7 7.5 cmbsy8 8.5 cmbsy9 9.5 cmbsy10 *
\_regtfm cmex 0 cmex8 8.5 cmex9 9.5 cmex10 *
\_regtfm cmexb 0 cmexb10 *
\_regtfm cmr 0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
cmr10 11.1 cmr12 15 cmr17 *
\_regtfm cmbr 0 cmbrx5 5.5 cmbrx6 6.5 cmbrx7 7.5 cmbrx8 8.5 cmbrx9 9.5
cmr10 11.1 cmbrx12 *
\_regtfm cmmi 0 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5 cmmi10 *
\_regtfm cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
\_loadmathfamily \langle number\rangle \langle font\rangle loads one math family, i.e. the triple of fonts in the text size, script size and script-script size. The \langle font\rangle is \langle font-id\rangle used in the \_regtfm parameter or the real TFM name. The family is saved as \fam\langle number\rangle.
\_setmathfamily \langle number\rangle \langle font-switch\rangle loads one math family like \_loadmathfamily does it. But the second parameter is a \langle font-switch\rangle declared previously by the \font primitive.
\_corrmsizes should be used in the \normalmath and \boldmath macros if you need a size correction when a selected math family is loaded. It is similar as ex-height correction but for math fonts.
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 10pt typesetting). The \skewchar of some math families are set here too.

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 format is generated. Maybe there are not all fonts in the TeX distribution installed.

2.15 Math macros

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

There is a problem: The \textbf{x_n} is tokenized as \textbf{x}, n and it works without problem. But \textbf{\int_a^b} is tokenized as \textbf{\int_a}, \textbf{^b}. The control sequence \textbf{\int_a} isn’t defined. We must write \textbf{\int_a^b}.

The Lua code presented here solves this problem. But you cannot set our own control sequence in the form \langle \textbf{\word} \rangle \langle \textbf{\nonletter} \rangle (where \textbf{\word} is sequence of letters) because such control sequences are inaccessible: preprocessor rewrites it.

The \mathsbon macro activates the rewriting rule \langle \textbf{\word} \rangle \langle \textbf{\nonletter} \rangle to \langle \textbf{\word} \rangle \langle \textbf{\nonletter} \rangle \langle \textbf{\letter} \rangle \langle \textbf{\nonletter} \rangle \langle \textbf{\letter} \rangle \langle \textbf{\nonletter} \rangle at input processor level. The \mathsboff deactivates it. You can ask by \textbf{\ifmathsbof} 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 second line of the document.
All mathcodes are set to equal values as in plain\TeX. But all encoding-dependend declarations (like these) will be set to different values when Unicode-math font is used.

\begin{verbatim}
\_mathcode`\^\^@="2201 % \cdot
\_mathcode`\^\^A="3223 % \downarrow
\_mathcode`\^\^B="010B % \alpha
\_mathcode`\^\^C="010C % \beta
\_mathcode`\^\^D="225E % \land
\_mathcode`\^\^E="023A % \lnot
\_mathcode`\^\^F="3232 % \in
\_mathcode`\^\^G="0119 % \pi
\_mathcode`\^\^H="0115 % \lambda
\_mathcode`\^\^I="010D % \gamma
\_mathcode`\^\^J="010E % \delta
\_mathcode`\^\^K="3222 % \uparrow
\_mathcode`\^\^L="2206 % \pm
\_mathcode`\^\^M="2208 % \oplus
\_mathcode`\^\^N="0231 % \infty
\_mathcode`\^\^O="0140 % \partial
\_mathcode`\^\^P="321A % \subset
\_mathcode`\^\^Q="321B % \supset
\_mathcode`\^\^R="225C % \cap
\_mathcode`\^\^S="225B % \cup
\_mathcode`\^\^T="0238 % \forall
\_mathcode`\^\^U="0239 % \exists
\_mathcode`\^\^V="220A % \otimes
\_mathcode`\^\^W="3224 % \leftrightarrow
\_mathcode`\^\^X="3220 % \leftarrow
\_mathcode`\^\^Y="3221 % \rightarrow
\_mathcode`\^\^Z="8000 % \ne
\_mathcode`\^[="2205 % \diamond
\_mathcode`\^[="3214 % \le
\_mathcode`\^[="3215 % \ge
\_mathcode`\^[="3211 % \equiv
\_mathcode`\^[="225F % \lor
\_mathcode`\^="8000 % \space
\_mathcode`\^="5021 % \prime
\_mathcode`\^="402B % \ast
\_mathcode`\^="5029 % \ast
\_mathcode`\^="2203 % \ast
\_mathcode`\^="202B % \ast
\_mathcode`\^="613B % \ast
\_mathcode`\^="2200 % \ast
\_mathcode`\^="013A % \ast
\_mathcode`\^="013D % \ast
\_mathcode`\^="303A % \ast
\_mathcode`\^="603B % \ast
\_mathcode`\^="313C % \ast
\_mathcode`\^="303D % \ast
\_mathcode`\^="313E % \ast
\_mathcode`\^="503F % \ast
\_mathcode`\^="405B % \ast
\end{verbatim}
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.

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

\mathchardef\alpha="010B \mathchardef\beta="010C \mathchardef\gamma="010D \mathchardef\delta="010E \mathchardef\epsilon="010F \mathchardef\zeta="0110 \mathchardef\eta="0111 \mathchardef\theta="0112 \mathchardef\iota="0113 \mathchardef\kappa="0114 \mathchardef\lambda="0115 \mathchardef\mu="0116 \mathchardef\nu="0117 \mathchardef\xi="0118 \mathchardef\pi="0119 . . . etc. (see \texttt{math-macros.opm})

\protected\def\log{\mathop{\text{log}}\nolimits} \protected\def\lg{\mathop{\text{lg}}\nolimits} \protected\def\ln{\mathop{\text{ln}}\nolimits} \protected\def\lim{\mathop{\text{lim}}} \protected\def\limsup{\mathop{\text{lim\thinspace sup}}} \protected\def\liminf{\mathop{\text{lim\thinspace inf}}} \protected\def\sin{\mathop{\text{sin}}\nolimits} \protected\def\arcsin{\mathop{\text{arcsin}}\nolimits} \protected\def\sinh{\mathop{\text{sinh}}\nolimits} \protected\def\cos{\mathop{\text{cos}}\nolimits} \protected\def\arccos{\mathop{\text{arccos}}\nolimits} \protected\def\cosh{\mathop{\text{cosh}}\nolimits} \protected\def\tan{\mathop{\text{tan}}\nolimits} \protected\def\arctan{\mathop{\text{arctan}}\nolimits} \protected\def\tanh{\mathop{\text{tanh}}\nolimits} \protected\def\cot{\mathop{\text{cot}}\nolimits} \protected\def\csc{\mathop{\text{csc}}\nolimits} \protected\def\max{\mathop{\text{max}}} \protected\def\min{\mathop{\text{min}}} \protected\def\sup{\mathop{\text{sup}}} \protected\def\inf{\mathop{\text{inf}}} \protected\def\arg{\mathop{\text{arg}}\nolimits}
These macros are defined similarly as in plain\TeX. Only internal macro names from plain\TeX with \@ character are re-written in more readable form. \\sp is alternative for ^, The \sb alternative for _ was defined at the line 27 of the file math-macros.opm. 

Active \texttt{\textbackslash prime} character is defined here. 

Math relations defined by the \texttt{\jointrel} plain \TeX macro:
\ldots, \cdots, \vdots, \ddots from plain \TeX
\ldots \cdots \vdots \ddots ;
ispired by plain \TeX
\ldots \cdots \vdots \ddots ;

Math accents (encoding dependent declarations).
Macros based on \delimiter, \*witdelims and \radical primitives.

\def\moustache{\delimiter"437A340 } % top from (, bottom from )
\def\moustache{\delimiter"537B341 } % top from , bottom from ()
\def\group{\delimiter"462833A } % extensible ( with sharper tips
\def\group{\delimiter"562933B } % extensible ) with sharper tips
\def\arrowvert{\delimiter"26A33C } % arrow without arrowheads
\def\Arrowvert{\delimiter"26B33D } % double arrow without arrowheads
\def\bracevert{\delimiter"77C33E } % the vertical bar that extends braces
\def\Vert{\delimiter"26B30D } %let\|\=\Vert
\def\vert{\delimiter"26A30C }
\def\uparrow{\delimiter"3222378 }
\def\downarrow{\delimiter"3223379 }
\def\updownarrow{\delimiter"326C33F }
\def\Uparrow{\delimiter"322A37E }
\def\Downarrow{\delimiter"322B37F }
\def\Updownarrow{\delimiter"326D377 }
\def\backslash{\delimiter"26E30F } % for double coset G\backslash H
\def\rangle{\delimiter"526930B }
\def\langle{\delimiter"426830A }
\def\rbrace{\delimiter"5267309 } %let\}\=\rbrace
\def\lbrace{\delimiter"4266308 } %let\{\=\lbrace
\def\rceil{\delimiter"5265307 }
\def\lceil{\delimiter"4264306 }
\def\rfloor{\delimiter"5263305 }
\def\lfloor{\delimiter"4262304 }
\def\choose{\atopwithdelims()}
\def\brack{\atopwithdelims\[\]}
\def\brace{\atopwithdelims\lbrack\rbrack}
\def\sqrt{\radical"270370 } %public \sqrt ;
\mathstrut, \vphantom, \hphantom, \phantom, \mathstrut, and \smash macros from plain \TeX.
\cong, \notin, \rightleftharpoons, \buildrel, \doteq, \bmod and \pmod macros from plain \TeX.

\texttt{math-macros.opm}

569 \_protected\_def\_cong\{\_mathrel\{\_mathpalette\_overeq\_sim\}\} \% congruence sign
570 \_def\_overeq\#2\{\_lower.05em\_vbox\{\_lineskiplimit\_maxdimen\_lineskip=-.05em
571 \_ialign{$\_math\#1\_hfil\&\_hfil\#2\_crcr\#2\_crcr\#1\_crcr}}\}
572 \_protected\_def\_notin\{\_mathrel\{\_mathpalette\_cancel\_in\}\}
573 \_def\_cancel\#1\#2\{\_math\_ooalign{$\_hfil\#1\_mkern1mu/\_hfil$\#2}
574 \_protected\_def\_rightleftharpoons\{\_vcenter\{\_math\_hbox\{\_oalign\{\_raise.2em
575 \_hbox{$\#1\_rightharpoonup$}\_crcr$\#1\_leftharpoondown$\}}\}
576 \_protected\_def\_buildrel\#1\over\#2\{\_mathrel\{\_kern\zo \#2\}_limits^{\#1}\}
577 \_protected\_def\_doteq\{\_buildrel\_textstyle.\over=\}
578 \_private \in \sim ;
579 \_public \cong \notin \rightleftharpoons \buildrel \doteq ;
580 \_protected\_def\_bmod\{\_nonscript\_mskip-\_medmuskip\_mkern5mu
581 \_mathbin\{\_rm mod\}_penalty900\_mkern5mu\_nonscript\_mskip-\_medmuskip\}
582 \_protected\_def\_pmod\#1\{\_allowbreak\_mkern18mu({\_rm mod}\_thinsk\_thinsk\#1)\}

\texttt{math-macros.opm}

596 \_protected\_def\_matrix\#1\{\_null\_thinsk
597 \_edef\_stylenum\{\_the\_numexpr\_mathstyle/2\_relax\%
598 \_vcenter\{\_matrixbaselines\_math
599 \_ialign{$##\_hfil$&\_quad{##\_unsskip}\_hfil\_crcr
600 \_mathstrut\_crcr\noalign{\_kern-\_baselineskip}
601 #1\_crcr\_noalign{\_kern-\_baselineskip}}\}
602 \_thinsk\}
603 \_def\_matrixscriptbaselines\{\_baselineskip=.7\_baselineskip
604 \_def\_quad \{\_hskip.7em\_relax\}
605 \_let\_matrixstyle=\_scriptstyle
606 \_def\_lt\_matrixstyle=\_scriptscriptstyle
607 \_protected\_def\_pmatrix\#1\{\_left(\_matrix{\#1}\_right)\}
608 \_public \_matrix \pmatrix ;
609
610 \_protected\_long\_def\_cases\#1\{\_left\{\_thinsk\_vcenter\{\_normalbaselines\_def\_matrixstyle\%
611 \_let\_matrixbaselines=\_relax \% matrix inside matrix does not change size again
612 \_ifcase\_stylenum \_or \_matrixscriptbaselines \_or \_matrixscriptbaselines
613 \_or \_scriptscriptstyle \_or \_scriptstyle
614 \_fi
615 \_protected\_def\_\_matrix\#1\{\_left(\_matrix{\#1}\_right)\}
616 \_public \_matrix \_matrix ;
617
\texttt{math-macros.opm}

The \texttt{\cases} and \texttt{\bordermatrix} macros are identical from plain \TeX.
The \texttt{\textbackslash eqalign} macro behaves like in Plain \TeX{} by default. It creates the \texttt{\textbackslash vcenter} in the math mode. The contents is two column \texttt{\textbackslash halign} with right aligned left column and left aligned right column. The table items are in \texttt{\textbackslash displaystyle} and the \texttt{\textbackslash baselineskip} is advanced by \texttt{\textbackslash jot} (3pt in plain \TeX{}). It follows from the default settings of \texttt{\textbackslash eqlines} and \texttt{\textbackslash eqstyle} parameters.

In Op\TeX{}, this macro is more flexible. See section 4.4 in the Typesetting Math with Op\TeX{}. The \texttt{\textbackslash baselineskip} value is set by the \texttt{\textbackslash eqlines} parameter and math style by the \texttt{\textbackslash eqstyle} parameter.

There are more possible columns than two (used in classical Plain \TeX{}): \texttt{rlcrlcrlc} etc. where \texttt{r} and \texttt{l} columns are without spaces and \texttt{c} column (if used) has the space \texttt{\textbackslash eqspace}/2 at its both sides.

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

Op\TeX{} offers \texttt{\textbackslash displaylines} as an alternative case of usage \texttt{\textbackslash displaylines}. See section 4.3 in the Typesetting Math with Op\TeX{}. The centered formulas are in \texttt{\textbackslash vcenter} in this case, so lines cannot be broken to more pages, but this case enables to use \texttt{\textbackslash leqno} or \texttt{\textbackslash leqno or \textbackslash eqmark}.

\texttt{\textbackslash openup}, \texttt{\textbackslash eqalignno} and \texttt{\textbackslash leqalignno} macros are copied from Plain \TeX{} unchanged.

These macros are inspired from \texttt{ams-math.tex} file.
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.

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

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

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

The \mathbox{⟨text⟩} macro is copied from OPmac trick 078. It behaves like \hbox{⟨text⟩} but the ⟨text⟩ is scaled to smaller size if it is used in scriptstyle or scriptscript style.
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 new UnicodeMath font isn’t loaded until \doloadmath is used.

\loadboldmath \{bold-font\} \to \{normal-font\} loads bold variant only if \{normal-font\} was sucessfully loaded by the \loadmath. For example:

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

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

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

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

The \loadmath macro was succesfully tested on:

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

2.16.1 Unicode-math macros preloaded in the format

\loadmath \{Unicode-math fonts <2020-06-06>\} % preloaded in format math-unicode.opm

\loadmath \{Unicode-math font\} loads given font. It does:

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

\noloadmath disallows Unicode-math loading by \_mathloadingfalse.
\doloadmath allows Unicode-math loading by \_mathloadingtrue.
\loadboldmath \{\textbf{\texttt{bold-font}}} \to \{\textit{\texttt{normal-font}}\} defines \unimathboldfont as \textbf{\texttt{bold-font}} only if \unimathfont is defined as \textit{\texttt{normal-font}}. It is used when \texttt{\boldmath} macro is run. When no \unimathboldfont is defined then the \texttt{\boldmath} macro use “fake bold” generated by \texttt{embolden} Lua\TeX\ font feature.

\begin{verbatim}
\loadboldmath \{\texttt{\boldmath}\to \texttt{\normalmath}\} \defines \unimathboldfont as \texttt{\boldmath} only if \unimathfont is defined as \texttt{\normalmath}. It is used when \texttt{\boldmath} macro is run. When no \unimathboldfont is defined then the \texttt{\boldmath} macro use “fake bold” generated by \texttt{embolden} Lua\TeX\ font feature.
\end{verbatim}

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

\begin{verbatim}
\normalunimath\{\setmathsizes\} \normalmath \{\setmathsizes\}
\end{verbatim}

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

\begin{verbatim}
\umathrange \{\from\} \{\to\} \{\class\} \{\family\} \{\first\} sets \Umathcodes of the characters in the interval
\end{verbatim}
\umahrangegreek \langle \text{first} \rangle \to \langle \text{omega} \rangle \langle \text{first} \rangle + \text{1} \langle \text{first} \rangle + \text{2} \text{ etc.}, but \umathcharholes are skipped (\umathcharholes are parts of the Unicode table not designed for math alphabets but they causes that the math alphabets are not continuously spread out in the table; I mean that the designers were under the influence of drugs when they created this part of the Unicode table). The \langle \text{from} \rangle \to \langle \text{to} \rangle \ clause includes normal letters like A-Z.

\umahrangegreek \langle \text{first} \rangle \text{ is the same as } \umathrange \{ \langle \text{alpha} \rangle \to \langle \text{omega} \rangle \} \langle \text{first} \rangle.

\umahrangegreek \langle \text{first} \rangle \text{ is the same as } \umathrange \{ \langle \text{Alpha} \rangle \to \langle \text{Omega} \rangle \} \langle \text{first} \rangle.

\greekdef \langle \text{control sequences} \rangle \relax \text{ defines each control sequence as a normal character with codes \umathnumB, \umathnumB+1, \umathnumB+2 etc. It is used for redefining the contol sequences for math Greek} \\alpha, \\beta, \gamma \text{ etc.}

\begin{verbatim}
\newcount\umathnumA \newcount\umathnumB
\def\umathcorr#1#2{\ea#1\ea{\the#2}}
\def\umathprepare#1{\def\tmp{#2 #3 }\umathrangea#1}
\def\umathrangea#1-#2{\umathnumA=`#1\relax }
\loop
  \umathcorr\umathprepare\umathnumB
  \Umathcode \umathnumA = \tmp \umathcorrtemp\umathvalue(\umathnumB)
  \ifnum\umathnumA<`#2\relax
    \advance\umathnumA by1
  \repeat
\def\umathrangegreek{\umathrange{\^\^0391-\^\^03a9}}
\def\umathrangeGREEK{\umathrange{\^\^03B1-\^\^03d6}}
\def\greekdef#1{\ifx#1\relax \else
  \begingroup \lccode`X=\umathnumB \\
  \lowercase{\endgroup \def#1{X}}\relax
\fi}
\end{verbatim}

2.16.2 Macros and codes set when \loadmatfont is processed

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

\begin{verbatim}
\codedecl \ncharrmA \{Uni math codes <2020-11-13>} \% preloaded on demand by \loadmath
\end{verbatim}

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

\begin{verbatim}
\umathnumB="0391
\greekdef {\text{Alpha} \beta \gamma \delta \epsilon \zeta \eta \theta \iota \kappa \lambda \mu \nu \xi \omicron \pi \rho \sigma \tau \upsilon \phi \chi \psi \omega} \relax
\umathnumB="03B1
\greekdef {\text{alpha} \beta \gamma \delta \varepsilon \zeta \eta \theta \iota \kappa \lambda \mu \nu \xi \omicron \pi \rho \sigma \varsigma \tau \upsilon \phi \varphi \chi \psi \omega \vartheta \varkappa \varepsilon} \relax
\end{verbatim}

The math alphabets are declared here using the \umathrange{\{range\}}{\{class\}}{\{family\}}{\{starting-code\}} macro.
The \texttt{\cal}, \texttt{\bbchar}, \texttt{\frak}, \texttt{\script} and the \texttt{\rm}, \texttt{\bf}, \texttt{\it}, \texttt{\bi}, \texttt{\tt} are defined here. Their “8bit definitions” from the file \texttt{math-preload.om} (section 2.14) are removed.

You can redefine them again if you need different behavior (for example you don’t want to use sans serif bold in math). What to do:
Each Unicode slot carries information about math type. This is saved in the file `mathclass.txt` which is copied to `mathclass.opm`. The file has the following format:

```plaintext
<table>
<thead>
<tr>
<th>Code</th>
<th>Math Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>002E</td>
<td>P</td>
</tr>
<tr>
<td>002F</td>
<td>B</td>
</tr>
<tr>
<td>0030-0039</td>
<td>N</td>
</tr>
<tr>
<td>003A</td>
<td>P</td>
</tr>
<tr>
<td>003B</td>
<td>P</td>
</tr>
<tr>
<td>003C</td>
<td>R</td>
</tr>
<tr>
<td>003D</td>
<td>R</td>
</tr>
<tr>
<td>003E</td>
<td>R</td>
</tr>
<tr>
<td>003F</td>
<td>P</td>
</tr>
<tr>
<td>0040</td>
<td>N</td>
</tr>
<tr>
<td>0041-005A</td>
<td>A</td>
</tr>
<tr>
<td>005B</td>
<td>O</td>
</tr>
<tr>
<td>005C</td>
<td>B</td>
</tr>
<tr>
<td>005D</td>
<td>C</td>
</tr>
<tr>
<td>005E</td>
<td>N</td>
</tr>
<tr>
<td>005F</td>
<td>N</td>
</tr>
</tbody>
</table>
```

We have to read this information and convert it to the `\Umathcode` macros.
Each math symbol has its declaration in the file `unicode-math-table.tex` which is copied to `unimath-table.opm`. The file has following format:

```latex
70 \UnicodeMathSymbol{00394}{\mupDelta}{\mathalpha}{capital delta, greek}\
71 \UnicodeMathSymbol{00395}{\mupEpsilon}{\mathalpha}{capital epsilon, greek}\
72 \UnicodeMathSymbol{00396}{\mupZeta}{\mathalpha}{capital zeta, greek}\
73 \UnicodeMathSymbol{00397}{\mupEta}{\mathalpha}{capital eta, greek}\
74 \UnicodeMathSymbol{00398}{\mupTheta}{\mathalpha}{capital theta, greek}\
75 \UnicodeMathSymbol{00399}{\mupIota}{\mathalpha}{capital iota, greek}\
76 \UnicodeMathSymbol{0039A}{\mupKappa}{\mathalpha}{capital kappa, greek}\
77 \UnicodeMathSymbol{0039B}{\mupLambda}{\mathalpha}{capital lambda, greek}\
78 \UnicodeMathSymbol{0039C}{\mupMu}{\mathalpha}{capital mu, greek}\
79 \UnicodeMathSymbol{0039D}{\mupNu}{\mathalpha}{capital nu, greek}\
80 \UnicodeMathSymbol{0039E}{\mupXi}{\mathalpha}{capital xi, greek}\
81 \UnicodeMathSymbol{0039F}{\mupOmicron}{\mathalpha}{capital omicron, greek}\
82 \UnicodeMathSymbol{003A0}{\mupPi}{\mathalpha}{capital pi, greek}\
83 \UnicodeMathSymbol{003A1}{\mupRho}{\mathalpha}{capital rho, greek}\
84 \UnicodeMathSymbol{003A3}{\mupSigma}{\mathalpha}{capital sigma, greek}\
85 \UnicodeMathSymbol{003A4}{\mupTau}{\mathalpha}{capital tau, greek}\
```

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

```latex
171 \begingroup % input unimath-table.opm (it is a copy of unicode-math-table.tex):
172 \def\UnicodeMathSymbol{\#1\#2\#3}[\#4]{%\global\Umathcharnumdef\#2=\Umathcodenum\#1\relax%\ifx\#3\mathopen\gdef\#2{\Udelimiter 4 1 \#1}\fi%\ifx\#3\mathclose\gdef\#2{\Udelimiter 5 1 \#1}\fi%\ifx\#3\mathaccent\gdef\#2{\Umathaccent fixed 7 1 \#1}\fi%}
173 \input unimath-table.opm
174 \endgroup
```

Many special characters must be declared with care...

```latex
185 \begingroup % input unimath-codes.opm
186 \def\UnicodeMathSymbol{\#1\#2\#3}[\#4]{%\global\Umathcharnumdef\#2=\Umathcodenum\#1\relax%\ifx\#3\mathopen\gdef\#2{\Udelimiter 4 1 \#1}\fi%\ifx\#3\mathclose\gdef\#2{\Udelimiter 5 1 \#1}\fi%\ifx\#3\mathaccent\gdef\#2{\Umathaccent fixed 7 1 \#1}\fi%}
187 \def\sqrt{\Uradical 1 \#21A}\def\cuberoot{\Uradical 1 \#21B}\def\fourthroot{\Uradical 1 \#21C}\def\intwithnolimits#1#2{
188 \ifx#1\relax\else\ea\let\csname csstring#1op\endcsname=#1%\ea\def#1{\csname csstring#1op\endcsname \nolimits}\bgroup\lccode`~=#2\lowercase\egroup\mathcode`~="8000\let~=#1}\ea\intwithnolimits\fi}
189 \protected\def\sqrt{\Uradical 1 \#21A}\def\cuberoot{\Uradical 1 \#21B}\def\fourthroot{\Uradical 1 \#21C}
190 \protected\def\intwithnolimits{\intwithnolimits\#1\#2}{\intwithnolimits\#1\#2\relax\else\ea\let\csname csstring#1op\endcsname=#1%\ea\def#1{\csname csstring#1op\endcsname \nolimits}\bgroup\lccode`~=#2\lowercase\egroup\mathcode`~="8000\let~=#1}\ea\intwithnolimits\fi}
191 \intwithnolimits{\intwithnolimits\#1\#2\relax\else\ea\let\csname csstring#1op\endcsname=#1%\ea\def#1{\csname csstring#1op\endcsname \nolimits}\bgroup\lccode`~=#2\lowercase\egroup\mathcode`~="8000\let~=#1}\ea\intwithnolimits\fi}
192 \protected\def\sqrt{\Uradical 1 \#21A}\def\cuberoot{\Uradical 1 \#21B}\def\fourthroot{\Uradical 1 \#21C}
193 \protected\def\intwithnolimits{\intwithnolimits\#1\#2}{\intwithnolimits\#1\#2\relax\else\ea\let\csname csstring#1op\endcsname=#1%\ea\def#1{\csname csstring#1op\endcsname \nolimits}\bgroup\lccode`~=#2\lowercase\egroup\mathcode`~="8000\let~=#1}\ea\intwithnolimits\fi}
```

We have to read this information and convert it to the Unicode math codes.
Aliases are declared here. They are names not mentioned in the unimath-table.opm file but commonly used in \TeX.
The \texttt{\not} macro is redefined here. If the \texttt{\_not\langle char\rangle} is defined (by \texttt{\_negationof}) then this macro is used. Else centered / is printed over the \langle char\rangle.

\begin{verbatim}
\def\_negationof #1#2{\ea\let _csname _not\csstring#1\endcsname =#2}
\negationof = \neq \negationof < \nless \negationof > \ngtr \negationof \rightarrow \nrightarrow \negationof \leftrightarrow \nleftrightarrow \negationof \Leftarrow \nLeftarrow \negationof \Rightarrow \nRightarrow \negationof \exists \nexists \negationof \ni \nni \negationof \parallel \nparallel \negationof \sim \nsim \negationof \approx \napprox \negationof \equiv \nequiv \negationof \asymp \nasymp \negationof \lesssim \nlesssim \negationof \gtrsim \ngtrsim \negationof \lessgtr \nlessgtr \negationof \preceq \npreceq \negationof \succeq \nsucceq \negationof \subset \nsubset \negationof \supset \nsupset \negationof \subseteq \nsubseteq \negationof \supseteq \nsupseteq \negationof \vdash \nvdash \negationof \vDash \nvDash
\end{verbatim}
Newly declared public control sequences are used in internal macros by \texttt{Op\TeX}. We need to get new meanings of these control sequences in private name space.

2.16.3 A few observations

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

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

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

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

3. You can combine more Unicode math fonts in single formula using \texttt{Op\TeX} trick 0030.

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 \texttt{\fontfam[termes]} or by \texttt{\loadmath{⟨math-font⟩}}) and then you can do \texttt{\input print-unimath.opm}. The big table with all math symbols is printed.
2.17 Scaling fonts in document (high-level macros)

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

\texttt{\verbatimline{3} \verbatimline{4} \verbatimline{5} \verbatimline{6} \verbatimline{7} \verbatimline{8} \verbatimline{9} \verbatimline{10} \verbatimline{11} \verbatimline{12} \verbatimline{13} \verbatimline{14} \verbatimline{15} \verbatimline{16} \verbatimline{17} \verbatimline{18} \verbatimline{19} \verbatimline{20} \verbatimline{21} \verbatimline{22} \verbatimline{23} \verbatimline{24} \verbatimline{25} \verbatimline{26} \verbatimline{27} \verbatimline{28} \verbatimline{29} \verbatimline{30} \verbatimline{31} \verbatimline{32} \verbatimline{33} \verbatimline{34} \verbatimline{35} \verbatimline{36} \verbatimline{37} \verbatimline{38} \verbatimline{39} \verbatimline{40} \verbatimline{41} \verbatimline{42} \verbatimline{43} \verbatimline{44} \verbatimline{45} \verbatimline{46} \verbatimline{47} \verbatimline{48} \verbatimline{49} \verbatimline{50} \verbatimline{51} \verbatimline{52} \verbatimline{53} \verbatimline{54} \verbatimline{55} \verbatimline{56} \verbatimline{57} \verbatimline{58} \verbatimline{59} \verbatimline{60} \verbatimline{61} \verbatimline{62} \verbatimline{63} \verbatimline{64} \verbatimline{65} \verbatimline{66} \verbatimline{67} \verbatimline{68} \verbatimline{69} \verbatimline{70} \verbatimline{71} \verbatimline{72} \verbatimline{73} \verbatimline{74} \verbatimline{75} \verbatimline{76} \verbatimline{77} \verbatimline{78} \verbatimline{79} \verbatimline{80} \verbatimline{81} \verbatimline{82} \verbatimline{83} \verbatimline{84} \verbatimline{85} \verbatimline{86} \verbatimline{87} \verbatimline{88} \verbatimline{89} \verbatimline{90} \verbatimline{91} \verbatimline{92} \verbatimline{93} \verbatimline{94} \verbatimline{95} \verbatimline{96} \verbatimline{97} \verbatimline{98} \verbatimline{99} \verbatimline{100} \verbatimline{101} \verbatimline{102} \verbatimline{103} \verbatimline{104} \verbatimline{105} \verbatimline{106} \verbatimline{107} \verbatimline{108} \verbatimline{109} \verbatimline{110} \verbatimline{111} \verbatimline{112} \verbatimline{113} \verbatimline{114} \verbatimline{115} \verbatimline{116} \verbatimline{117} \verbatimline{118} \verbatimline{119} \verbatimline{120} \verbatimline{121} \verbatimline{122} \verbatimline{123} \verbatimline{124} \verbatimline{125} \verbatimline{126} \verbatimline{127} \verbatimline{128} \verbatimline{129} \verbatimline{130} \verbatimline{131} \verbatimline{132} \verbatimline{133} \verbatimline{134} \verbatimline{135} \verbatimline{136} \verbatimline{137} \verbatimline{138} \verbatimline{139} \verbatimline{140} \verbatimline{141} \verbatimline{142} \verbatimline{143} \verbatimline{144} \verbatimline{145} \verbatimline{146} \verbatimline{147} \verbatimline{148} \verbatimline{149} \verbatimline{150} \verbatimline{151} \verbatimline{152} \verbatimline{153} \verbatimline{154} \verbatimline{155} \verbatimline{156} \verbatimline{157} \verbatimline{158} \verbatimline{159} \verbatimline{160} \verbatimline{161} \permalink{codes/latex}
\def \settmpdim#1#2{%  \tmpdim=#1pt \divide \tmpdim by1000  \tmpdim=\ea \ignorept \the#2\tmpdim}  
\public \typoscale ;

\setbaselineskip \{\langle \baselineskip \rangle\} sets new \baselineskip and more values of registers which are dependent on the {\baselineskip} including the \strutbox.

\def \setbaselineskip #1{\if$#1$\else  \tmpdim=#1ptunit  \baselineskip=\tmpdim \relax  \bigskipamount=\tmpdim plus.33333\tmpdim minus.33333\tmpdim  \medskipamount=.5\tmpdim plus.16666\tmpdim minus.16666\tmpdim  \smallskipamount=.25\tmpdim plus.08333\tmpdim minus.08333\tmpdim  \normalbaselineskip=\tmpdim  \jot=.25\tmpdim  \maxdepth=.33333\tmpdim  \setbox \strutbox=\hbox{\vrule height.709\tmpdim depth.291\tmpdim width0pt}  \fi}  

\setmainvalues sets the current font size and \baselineskip values to the \mainfosize and \mainbaselineskip registers. It redefines itself in order to set the main values only first. \scalemain returns to these values if they were set. Else they are set to 10/12pt.

\def \setmainvalues {\mainbaselineskip=\baselineskip  \mainfosize=\optsize  \topskip=\mainfosize \splittopskip=\topskip  \ifmmode \else \bf \it \bi \rm \fi % load all basic variants of the family  \normalmath % load fonts if \typosize is running first  \let \setmainvalues =\setmainvaluesL}  
\def \setmainvaluesL {\ifdim \mainfosize=\zoo  \mainfosize=10pt \mainbaselineskip=12pt  \let \setmainvalues=\setmainvaluesL  \fi}  
\public \scalemain \mainfosize \mainbaselineskip ;

\thefontsize \langle \langle \size \rangle \rangle \and \thefontscale \langle \langle \factor \rangle \rangle do modification of the size of the current font. They are implemented by the \newcurrfontsize macro.

\protected \def \thefontsize[#1]{\if$#1$\else  \tmpdim=#1ptunit  \newcurrfontsize{at \tmpdim}  \fi}  
\protected \def \thefontscale[#1]{\ifx$#1$\else  \tmpdim=#1pt \divide \tmpdim by1000  \tmpdim=\ea \ea \ea \ignorept \pdffontsize \font \tmpdim  \newcurrfontsize{at \tmpdim}  \fi}  
\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.
The \boldify macro does \let\it\bi and \let\normalmath=\boldmath.

We need to use a font selector for default pagination. Because we don’t know what default font size will be selected by the user, we use this \_rmfixed macro. It sets the \rm font from default font size (declared by first \_typosize command and redefines itself be only the font switch for next pages.

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 –
  - background box, if \pgbackground is non-empty,
  - headline box by \_makeheadline, if the \headline is non-empty,
  - \vbox to\_vsiz of \_pagecontents which consists of –
    - \_pagedest, the page destination pg:(\gpageno) for hyperlinks is created here,
    - \topins box if non-empty (from \_topinserts),
    - \box255 with completed vertical material from main vertical mode,
    - \_footnoterule and \footins box if non-empty (from \fnote, \_footnote),
    - \pgbottomskip (default is 0pt),
  - footnote box by \_makefootline, if the \footline is non-empty
- \_endoutput which does:
  - increments \pageno using \_advancemageno
  - runs output routine repeatedly if \dosupereject is activated.

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

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

97
The \_prepoffsets can set \hoffset differently for left or right page. It is re-defined by the \margins macro.

The \_regmark tokens list includes accumulated #2 from the \regmacro. Logos and another macros are re-defined here (locally) for their usage in headlines or footlines.

The \_completepage is similar what plain \TeX does in its output routine. New is only \_backgroundbox. It is \vbox with zero height with its contents (from \pgbackground) lapped down. It is shifted directly to the left-upper corner of the paper.

The \_ensureblack sets the typesetting of its parameter locally to \Black color. We needn’t do this if colors are never used in the document. So, default value of the \_ensureblack macro is empty. But first usage of color macros in the document re-defines \_ensureblack. See the section 2.20 for more details.

The \_pagecontents is similar as in plain \TeX. The only difference is that the \pagedest is inserted at the top of \_pagecontents and \_ensureblack is applied to the \topins and \footins material. The \_footnoterule is defined here.
\texttt{\textbackslash pagedest} is a destination defined in \texttt{output.opm} that is used for pages. The \texttt{\textbackslash def\_pagedest} command is used to define the destination of the page.

\begin{verbatim}
\_def\_pagedest % destination of the page \_ifvoid\_topins \_else \_ensureblack(\unbox\_topins)\fi
\_def \_footnoterule {
_\kern-3pt \_hrule width 2truein \_kern 2.6pt }
\end{verbatim}

\texttt{\textbackslash pageno}, \texttt{\textbackslash folio}, \texttt{\textbackslash nopagenumbers}, \texttt{\textbackslash advancepageno} and \texttt{\textbackslash normalbottom} used in the context of the output routine from plain \TeX{} is defined here. Only the \texttt{\textbackslash raggedbottom} macro is defined differently. We use the \texttt{\textbackslash pgbottomskip} register here which is set to 0pt by default.

Macros for footnotes are the same as in plain \TeX{}. There is only one difference: \texttt{\textbackslash vfootnote} is implemented as \texttt{\_opfootnote} with empty parameter \texttt{#1}. This parameter should do a local settings inside the \texttt{\_footins} group and it does it when \texttt{\fnote} macro is used.

The \texttt{\_opfootnote} nor \texttt{\_vfootnote} don't take the footnote text as a parameter. This is due to user can do catcode settings (like inline verbatim) in the footnote text. This idea is adapted from plain \TeX{}.

The \texttt{\_footnote} and \texttt{\_footstrut} is defined as in plain \TeX{}.

The \texttt{\_topins} macros \texttt{\topinsert}, \texttt{\midinsert}, \texttt{\pageinsert}, \texttt{\endinsert} are the same as in plain \TeX{}.

The \texttt{\_topins} macros \texttt{\topinsert}, \texttt{\midinsert}, \texttt{\pageinsert}, \texttt{\endinsert} are the same as in plain \TeX{}.
The \draft macro is an example of usage \pgbackground to create water color marks.

2.19 Margins

The \margins macro is documented in the section 1.2.1.

\newdimen\pgwidth \newdimen\pgheight \pgwidth=0pt
\newdimen\shiftoffset
\def\margins#1#2#3#4#5#6#7{
  \def\tmp{#7}
  \ifx\tmp\empty
    \opwarning{\string\margins: missing unit, mm inserted}\def\tmp{mm}\fi
  \setpagedimensions #2 \relax
  \if#3\relax\hoffset = \dimexpr (#3, #4, #5, #6) \relax \else \hoffset = \dimexpr #3 - #4 \relax \fi
  \if#4\relax\hsize = \dimexpr \pgwidth - #3 - #4 \relax \else \hsize = \dimexpr #3 \relax \fi
  \if#5\relax\voffset = \dimexpr (#5, #6, #7) \relax \else \voffset = \dimexpr #5 \relax \fi
  \if#6\relax\vsize = \dimexpr \pgheight - #5 - #6 \relax \else \vsize = \dimexpr #5 \relax \fi
  \if#7\relax\shiftoffset = 0pt \else \if#8\relax\shiftoffset = \dimexpr \pgwidth - #7 \relax \else \shiftoffset = \dimexpr #7 \relax \fi\fi
}
The common page dimensions are defined here.

\magscale \[
\langle \text{factor} \rangle
\]
does \text{mag} = \langle \text{factor} \rangle and recalculates page dimensions to their true values.

\section*{2.20 Colors}

The colors have different behavior than fonts. A marks (whatsits) with color information are stored into PDF output and \TeX{} doesn’t interpret them. The PDF viewer (or PDF interpreter in a printer) reads these marks and switches colors according to them. This is totally independent on \TeX{} group mechanism. You can declare \texttt{\nolocalcolor} at the beginning of the document, if you want this behavior. In this case, if you set a color then you must to return back to black color using \texttt{\Black} manually.

By default, Op\TeX{} sets \texttt{\localcolor}. It means that the typesetting returns back to a previous color at the end of current group, so you cannot write \texttt{\Black} explicitly. This is implemented using \texttt{\aftergroup} feature. There is a limitation of this feature: when a color selector is used in a group of a box, which is saved by \texttt{\setbox}, then the activity or reconstruction of previous color are processed at \texttt{\setbox} time, no in the box itself. You must correct it by double group:

\begin{verbatim}
\setbox0=\hbox{\Red text}  % bad: \Black is done after \setbox
\setbox0=\hbox{\Red text}  % good: \Black is done after group inside the box
\end{verbatim}

The implementation of colors is based on colorstack, so the current color can follow across more pages. It is not so obvious because PDF viewer (or PDF interpreter) manipulates with colors locally at each PDF page and it initializes each PDF page with black on white color.

Macros \texttt{\setcmykcolor{\langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle}} or \texttt{\setrgbcolor{\langle R \rangle \langle G \rangle \langle B \rangle}} or \texttt{\setgreycolor{\langle Grey \rangle}} should be used in color selectors or user can specify these macros explicitly.

The color mixing processed by the \texttt{\colordef} is done in the subtractive color model CMYK. If the result has a component greater than 1 then all components are multiplied by a coefficient in order to maximal component is equal to 1.

You can move a shared amount of CMY components (i.e. their minimum) to the \( K \) component. This saves the color tonners and the result is more true. This should be done by \texttt{\useK} command at the end of a linear combination used in \texttt{\colordef}. For example

\begin{verbatim}
\colordef \myColor {.3\Green + .4\Blue \useK}
\end{verbatim}
The \useK command exactly does:

\[ k' = \min(C, M, Y), \]
\[ C = (C - k')/(1 - k'), \quad M = (M - k')/(1 - k'), \quad Y = (Y - k')/(1 - k'), \]
\[ K = \min(1, K + k'). \]

You can use minus instead plus in the linear combination in \colordef. The given color is subtracted in such case and the negative components are rounded to zero immediately. For example

\colordef \Color {\Brown-\Black}

can be used for removing black component from the color. You can use the \(~\Black\) trick after \useK command in order to remove grey components occured during color mixing.

Finally, you can use ~ immediately preceeded before macro name of the color. Then the complementary color is used here.

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

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

If a CMYK to RGB or RGB to CMYK conversion is needed then the following simple formulae are used (ICC profiles are not supported):

CMYK to RGB:
\[ R = (1 - C)(1 - K), \quad G = (1 - M)(1 - K), \quad B = (1 - Y)(1 - K). \]

RGB to CMYK:
\[ K' = \max(R, G, B), \quad C = (K' - R)/K', \quad M = (K' - G)/K', \quad Y = (K' - B)/K', \quad K = 1 - K'. \]

The RGB to CMYK conversion is invoked when a color is declared using \setrgbcolor and it is used in \colordef or if it is printed when \onlyrgb 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 \onlycmyk is declared.

We declare internal boolean value \_iflocalcolor ad do \localcolor as default.

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

By default, the \setcmykcolor \setrgbcolor and \setgreycolor macros with \{\textit{component}\} parameter expand to \_setcolor\{\textit{pdf-primitive}\} using \_formatcmyk or \_formatrgb or \_formatgrey expandable macros. For example \setcmykcolor\{1 0 0\} expands to \_setcolor\{1 0 0 \Grey 1 0 0 \Red\}. We set both types of colors (for lines (K or RG or G) and for fills (r or rg or g) together in the \textit{pdf-primitive} command. This is the reason why the \_fillstroke uses both its parameters. If only fills are needed you can do \def\_fillstroke#1#2{#1}. If only strokes are needed you can do \def\_fillstroke#1#2{#2}.
The \onlyrgb declaration redefines \formatcmyk in order it expands to its conversion to RGB \langle pdf-primitive \rangle. This conversion is done by the \cmyktorgb macro. Moreover, \onlyrgb re-defines three basic RGB colors for RGB color space and re-declares \colordef as \rgbcolordef. The \onlycmyk macro does a similar work, it re-defines \formatrgb macro. The Grey color space is unchanged and works in both main settings (RGB or CMYK) without collisions.

The \setcolor macro redefines empty \ensureblack macro (used in output routine for headers and footers) to \ensureblackA which sets Black at the start of its parameter and returns to the current color at the end of its parameter.

The current color is saved into \currentcolor macro and colorstack is pushed. Finally, the \colorstackpop is initialized by \aftergroup if \localcolor is declared.

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

The colorstack is initialized here and the basic macros \colorstackpush, \colorstackpop and \colorstackset are defined here.

We need to open a special color stack for footnotes, because footnotes can follow on next pages and their colors are independent on colors used in the main page-body. The \openfnotestack is defined as \openfnotestackA when the \setcolor is used first. The \fnotestack is initialized in in \everyjob because the initialization is not saved to the format.

We use lua codes for RGB to CMYK or CMYK to RGB conversions and for addition color components in the \colordef macro. The \rgbtocmyk \langle R \rangle \langle G \rangle \langle B \rangle expands to \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle and the \cmyktorgb \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle expands to \langle R \rangle \langle G \rangle \langle B \rangle. The \colorcrop, \colordefFin and \douseK are auxiliary macros used in the \colordef. The \colordefFin 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.
We have a problem with the `.3f` directive in Lua code. It prints trailed zeros: (0.300 instead desired 0.3) but we want to save PDF file space. The macro \_stripzeros removes these trailing zeros at expand processor level. So \_stripzeros 0.300 0.400 0.560 ; expands to .3 .4 .56.

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

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

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

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

Public versions of \colordef and \useK macros are declared using \def, because the internal versions \colordef and \useK are changed during processing.

The \x11nam.def file 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 \typsize[11/14]
\morecolors
\endmulti
2.21 The .ref file

The .ref file has the name \jobname.ref and it saves information about references, TOC lines, etc. All data needed in next \TeX run are saved here. Opt\TeX reads this file at the beginning of the document (using \everyjob) if such file exists. The .ref file looks like:

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

where ⟨gpageno⟩ is internal page number globally numbered from one and ⟨pageno⟩ is a page number (\the\pageno) used in pagination (they may be differ). Each page begins with \_Xpage. The ⟨label⟩ is a label used by user in \label[⟨label⟩] and ⟨text⟩ is a text which should be referenced (the number of section or table, for example 2.3.14). The ⟨title⟩ is a title of the chapter (⟨level⟩=1, ⟨type⟩=chap), section (⟨level⟩=2, ⟨type⟩=sec), subsection (⟨level⟩=3, ⟨type⟩=secc). The \_Xpage is written at beginning of each page, the \_Xtoc is written when chapter or section or subsection title exists on the page and \_Xlabel when labeled object prefixed by \label[⟨label⟩] exists on the page.

The .ref file is read when the processing of the document starts using \everyjob. It is read, removed and opened to writing immediately. But the .ref file should be missing. If none forward references are needed in the document then .ref file is not created. For example, you only want to test a simple plain \TeX macro, you create test.tex file, you do optex test and you don’t need to see empty test.ref file in your directory.

The \_inputref macro is used in \everyjob. It reads \jobname.ref file if it exists. After the file is read then it is removed and opened to write a new contents to this file.

If the file does not exists then it is not created by default. It means that if you process a document without any forward references then no \jobname.ref file is created because it is unusable. The \_wref macro is dummy in such case.
If a macro needs to create and to use `.ref` file then such macro must use `\openref`. When the file is created (using internal `\_openref`) then the `\_wref {macro}{data}` is redefined in order to save the line `\wref{macro}{data}` to the `.ref` file using asynchronous `\write` primitive. Finally, the `\openref` destroys itself, because we need not to open the file again.

```
\def\openref {%
  \_def\_openref {%
    \_ifx \_wref \_wrefrelax \_openrefA{\_string\openref}\_fi
  \_def\_openrefA #1{%\_openref
    \_immediate\_openout\_reffile=\_jobname.ref\_relax
    \_def\_wref ##1##2{\_write\_reffile{\_string##1##2}}%
    \_immediate\_write\_reffile {\_pcent\_pcent\_space OPTeX <\_optexversion> - REF file (#1)}%
  }
  \_def\_openref {\_openrefA}
}\_def\_openrefA #1{%\_openref
  \_immediate\_openout\_reffile=\_jobname.ref\_relax
  \_def\_wref ##1##2{\_write\_reffile{\_string##1##2}}%
  \_immediate\_write\_reffile {\_pcent\_pcent\_space OPTeX <\_optexversion> - REF file (#1)}%
  \_def\_openref {\_openrefA}
}\_def\openref {\_openref}
```

We are using convention that the macros used in `.ref` file are named \_X{foo}. If there is a new version of OpTEX with different collection of such macros then we don’t want to read the `.ref` files produced by an old version of OpTEX or by OPmac. So first line of `.ref` file is in the form

```
\Xrefversion{⟨version⟩}
```

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

```
\_def\_REFversion{4} % actual version of .ref files in OpTeX
\_def\_Xrefversion#1{\_ifnum #1=\_REFversion\_relax \_else \_endinput \_fi}
\_public \Xrefversion ; % we want to ignore .ref files generated by OPmac
```

You cannot define your special `.ref` macros before `.ref` file is read because it is read in `\everyjob`. But you can define such macros using `\refdecl{⟨definitions of your ref macros⟩}`. This command sends to `.ref` file your ⟨definitions of your ref macros⟩ immediately. Next lines in `.ref` file should include our macros. Example from CTUstyle2:

```
\refdecl{%
  \def\totlist{} \def\toflist{}^^J
  \def\Xtab#1#2#3{\addto\totlist{\totline{#1}{#2}{#3}}}^^J
  \def\Xfig#1#2#3{\addto\toflist{\tofline{#1}{#2}{#3}}}
}
```

We must read ⟨definition of your ref macros⟩ when catcode of # is 12 because we needn’t to duplicate each # in the `.ref` file.

```
\_def\_refdecl{(\_hgroup \_catcode\#=12 \_refdeclA)
\_def\_refdeclA #1{(\_egroup \_openref
  \_immediate\_write\_reffile {\_pcent\_space \_string \_refdecl:\%}
  \_immediate\_write\_reffile {\_detokenize(#1)}%
  \_def\_refdeclA #1{(\_egroup \_openref}
\_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.

```
\_codedecl \ref {References <2020-03-03>} % preloaded in format
\_Xpage {gpageno}{pageno} saves the parameter pair into \_currpage. Resets \_lfnotenum; it is used if footnotes are numbered from one at each page.
```

Counter for number of unresolved references `\_ unresolvedrefs`. 107
\label[⟨label⟩] saves the decalred label to \_lastlabel and \wlabel{⟨text⟩} uses the \_lastlabel and activates \wref\_Xlabel{⟨label⟩}{⟨text⟩}.

\ref[⟨label⟩] uses saved \_lab:⟨label⟩ and prints (linked) ⟨text⟩. If the reference is backwarded then we know \_lab:⟨label⟩ without any need to read REF file. On the other hand, if the reference is forwarded, then we doesn’t know \_lab:⟨label⟩ in first run of TEX and we print warning and do \_openref.

\pgref[⟨label⟩] uses {⟨gpageno⟩} from \_pgref:⟨label⟩ and prints (linked) ⟨pageno⟩ using \_ilink macro.

2.23 Hyperlinks

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

- ref:⟨label⟩ – the destination is created when \label[⟨label⟩] is used, see also the section 2.22.
- toc:⟨tocrefnum⟩ – the destination is created at chap/sec/secct 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:⟨bibnum⟩ – the destination is created in bibliography reference, see also the section 2.32.1.
- url:⟨url⟩ – used by \url or \ulink, see also the end of this section.
The \textit{tocrefnum}, \textit{gpageno} and \textit{bibnum} are numbers starting from one and globally incremented by one in whole document. The registers \texttt{tocrefnum}, \texttt{gpageno} and \texttt{bibnum} are used for these numbers.

When a chap/sec/secc title is prefixed by \texttt{\label{[\texttt{\label}]}, then both types of internal links are created at the same destination place: \texttt{toc:}{\textit{tocrefnum}} and \texttt{ref:}{\textit{bibnum}}.

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

\texttt{\hyperlinks{<2020-04-22>}} \% preloaded in format

The \texttt{\hyperlinks} macro is redefined here to its “active” version. The \texttt{\hyperlinks} command in not used, then \texttt{\hyperlinks} does nothing else it is set to \texttt{\hyperlinks}. The \texttt{\hyperlinks} primitive is implemented by \texttt{\_destactive}. It creates a box in which the destination is shifted by \texttt{\_destactive}. The reason is that the destination is exactly at top border of the PDF viewer but we want to see the line where destination is. The destination box is positioned by different way depending on current vertical or horizontal mode.

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format

\texttt{\hyperlinks} \% preloaded in format
\url{⟨url⟩} does approximately the same as \ulink\[⟨url⟩\]{⟨url⟩}, but more work is done before the \ulink is processed. The link-version of ⟨url⟩ is saved to _tmpa and the printed version in _tmpb. The printed version is modified in order to set a breakpoints to special places of the ⟨url⟩. For example // is replaced by _urlskip/_urlskip/_urlbskip where _urlskip adds a small nobreakable glue between these two slashes and before them and _urlbskip adds a breakable glue after them.

The text version of the ⟨url⟩ is printed in _urlfont.

\url{⟨url⟩} does approximately the same as \ulink\[⟨url⟩\]{⟨url⟩}, but more work is done before the \ulink is processed. The link-version of ⟨url⟩ is saved to _tmpa and the printed version in _tmpb. The printed version is modified in order to set a breakpoints to special places of the ⟨url⟩. For example // is replaced by _urlskip/_urlskip/_urlbskip where _urlskip adds a small nobreakable glue between these two slashes and before them and _urlbskip adds a breakable glue after them.

The text version of the ⟨url⟩ is printed in _urlfont.

\url{⟨url⟩} does approximately the same as \ulink\[⟨url⟩\]{⟨url⟩}, but more work is done before the \ulink is processed. The link-version of ⟨url⟩ is saved to _tmpa and the printed version in _tmpb. The printed version is modified in order to set a breakpoints to special places of the ⟨url⟩. For example // is replaced by _urlskip/_urlskip/_urlbskip where _urlskip adds a small nobreakable glue between these two slashes and before them and _urlbskip adds a breakable glue after them.

The text version of the ⟨url⟩ is printed in _urlfont.

2.24 Making table of contents

\_Xtoc {⟨level⟩}{⟨type⟩}{⟨number⟩}{⟨title⟩} (in .ref file) reads the specified data and appends them to the \_toclist as \_tocline{⟨level⟩}{⟨type⟩}{⟨number⟩}{⟨title⟩}{⟨gpageno⟩}{⟨pageno⟩} where:

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

The last two parameters are restored from previous \_Xpage{⟨pgageno⟩}{⟨pgageno⟩}, data were saved in the \_currpage macro.

We read the ⟨title⟩ parameter by \scantoeol from .ref file because the ⟨title⟩ can include something like `{`.
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 \langle title\rangle) looks like:

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

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

\_tocrefnum is global counter of all TOC records (used by hyperlinks).

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

Parameters are \{\langle number\rangle\}\{\langle title\rangle\}\{\langle pageno\rangle\}.

The auxiliary macros are:

- \_llaptoclink\langle text\rangle does \_noindent \llap{\langle linked text\rangle}.
- \_tocdotfill creates dots in the TOC.
- \_nofirst\_macro applies the \_macro only if we don’t print the first record of the TOC.
- \_pgn\langle pageno\rangle creates \langle pageno\rangle as link to real \langle gpage\rangle saved in \#6 of \_tocline. This is temporarily defined in the \_tocline.

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

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

Nesting PDF outlines

The problem is that PDF format needs to know the number of direct descendants of each outline if we need to create the tree of structured outlines. But we know only the level of each outline. The required data should be calculated from TOC data. We use two steps over TOC data saved in the \_toclist where each record is represented by one \_tocline.

First step, the \_outlines macro sets \_tocline to \_outlinesA and calculates the number of direct descendants of each record. Second step, the \_outlines macro sets \_tocline to \_outlinesB and it uses prepared data and create outlines.

Each outline is mapped to the control sequence of the type \_ol:\langle num\rangle or \_ol:\langle num\rangle:\langle num\rangle or \_ol:\langle num\rangle:\langle num\rangle:\langle num\rangle etc. The first one is reserved for level 0, the second one for level 1 (chapters), third one for level 2 (sections) etc. The number of direct descendants will be stored in these macros after first step is finished. Each new outline of given level increases the \langle num\rangle at given level. When the first step is processed then (above that) the \_ol:.. sequence of the parent increase 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:

\begin{verbatim}
\advance \count2 by 1
% increases the mapping pointer of the type
\advance \langle\_ol:\langle\_count0:\langle\_count1:\langle\_count2\rangle of this section
\end{verbatim}

When second step is processed, then we only read the stored data about the number of descendants. Ad we use it in count parameter of \_pdfoutline primitive.

For linking, we use the same links as in TOC, i.e. the toc:\_the\_tocrefnum labels are used.

\begin{verbatim}
\insertoutline \{\langle text\rangle\} inserts one outline with zero direct descendants. It creates link destination of the type ou1:\langle num\rangle into the document (where \insertoutline is used) and the link itself is created too in the outline.
\end{verbatim}
2.25.2 Strings in PDF outlines

There are only two encodings for PDF strings (used in PDFoutlines, PDFinfo etc.). First one is PDFDocEncoding which is one-byte encoding, but most Czech or Slovak characters are missing here.

The second encoding is PDFunicode encoding which is implemented in this file. This encoding is TeX-discomfortable, because it looks like

```
\376\377\000C\000v\000i\001\015\000e\000n\000\355\000\040\000j\000\040
\000t\001\033\001\176
```

This example is a real encoding of the string “Cvičení je zátěž”. You can see that this is UTF-16 encoding (two bytes per character) with two starting bytes FEFF. Moreover, each byte is encoded by three octal digits preceded by backslash. The only exception is the visible ASCII character encoding: such a character is encoded by its real byte preceded by \000.

The `\octalprint` is lua script which prints the character code in the octal notation.

```
3 \octalprint
```

| \pdfunidef \macro{⟨text⟩} | \pdfunidef \macro{⟨text⟩} does more things than only converting to octal notation. The \langle text \rangle can be scanned in verbatim mode (it is true because \_Xtoc reads the \langle text \rangle 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 \langle text \rangle is detokenized, spaces are preprocessed using \replstring and then the \_pdfunidefB is repeated on each character. It calls the \directlua chunk to print octal numbers in the macro \_octalprint.

```
\def\_pdfunidefB{\_octalprint\_}{% $x$ -> x
```

```
\pdfunidef\macro{⟨text⟩}
```
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 in-line verbatim.

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

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.
You can re-define \printchap, \printsec or \printsecc macros if another design of section titles is needed. These macros get the \langle title \rangle text in its parameter. The common recommendations for these macros are:

- Use \abovetitle{\langle penaltyA\rangle}{\langle skipA\rangle} and \belowtitle{\langle skipB\rangle} for inserting vertical material above and below the section title. The arguments of these macros are normally used, i.e. \abovetitle inserts \langle penaltyA\rangle{\langle skipA\rangle} and \belowtitle inserts \langle skipB\rangle. But there is an exception: if \belowtitle{\langle skipB\rangle} is immediately followed by \abovetitle{\langle penaltyA\rangle}{\langle skipA\rangle} (for example section title is immediately followed by subsection title), then only \langle skipA\rangle is generated, i.e. \langle skipB\rangle{\langle penaltyA\rangle}{\langle skipA\rangle} is reduced only to \langle skipA\rangle. The reason of such behavior: we don’t want to duplicate vertical skip and we don’t want to use negative penalty in such cases. Moreover, \abovetitle{\langle penaltyA\rangle}{\langle skipA\rangle} takes whatever vertical skip (other than from \belowtitle) and generates only greater from this pair of skips. It means that \langle whatever-skip\rangle{\langle penaltyA\rangle}{\langle skipA\rangle} is transformed to \langle penaltyA\rangle\max(\langle whatever-skip\rangle{\langle skipA\rangle}). The reason of such behavior: we don’t want to duplicate vertical skips (from \belowlistskip, for example) above the title.

- Use \printrefnum{\langle pre\rangle}{\langle ref-num\rangle}{\langle post\rangle} in horizontal mode. It prints \langle pre\rangle{\langle ref-num\rangle}{\langle post\rangle}. The \langle ref-num\rangle is \thechapnum or \theseccnum or \theseccnum depending on what type of title is processed. If \nonum prefix is used then \printrefnum prints nothing. The macro \printrefnum does more work: it creates destination of hyperlinks (if \hyperlinks{} is used) and saves references from label (if \label{} precedes) and saves references for table of contents (if \maketoc is used).

- Use \nobreak for closing the paragraph for printing title. This command inserts \nobreak between each line of such paragraph, so the title cannot be broken to more pages.

- You can use \firstnoindent in order to the first paragraph after the title is not indented.

\begin{verbatim}
72 \def\printchap #1{\vfill\supereject
73 \vglue\medskipamount % shifted by topkip+\medskipamount
74 \chapfont \noindent \text{(chap) \printrefnum[0]\par
75 \nobreak \smallskip
76 \noindent \raggedright #1\nobpar}\mark{}%}
77 \firstnoindent
78 \abovetitle{\bigskip}
79 }
80 \def\printsec#1{\par
81 \abovetitle{\langle penalty-400\rangle}{\bigskip
82 \langle secfont \noindent \raggedright \printrefnum[0]{\quad}#1\nobpar}\insertmark{}%}
83 \firstnoindent
84 \abovetitle{\medskip}
85 }
86 \def\printsecc#1{\par
87 \abovetitle{\langle penalty-200\rangle}{\medskip}\smallskip
88 \langle seccfont \noindent \raggedright \printrefnum[0]{\quad}#1\nobpar}
89 \firstnoindent
90 }
\end{verbatim}

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)

\begin{verbatim}
104 \newcount\sectionlevel
105 \def\secinfo {\ifcase \sectionlevel
106 \part\or \chap\or \sec\or \secc\or \seccx\fi
107 }
\end{verbatim}

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}.

115
Default concept: Tables, figures and display maths are numbered from one in each section – subsections don’t reset these counters. Footnotes declared by \fnotenumchapters are numbered in each chapter from one.

The \_the* macros \_thechapnum, \_theseccnum, \_theseccnum, \_thetnum, \_thefnum and \_thednum include the format of numbers used when the object is printing. If chapter is never used in the document then \_chapnum=0 and \_oth\_chapnum. expands to empty. Sections have numbers ⟨num⟩ and subsections ⟨num⟩. On the other hand, if chapter is used in the document then \_chapnum>0 and sections have numbers ⟨num⟩. ⟨num⟩. ⟨num⟩.

\newcount \chapnum % chapters
\newcount \seccnum % sections
\newcount \seccnum % subsections
\newcount \tnum % table numbers
\newcount \fnum % figure numbers
\newcount \dnum % numbered display maths
\def \_chapx \{\_secx \_seccnum=0 \_lfnotenum=0 \}
\def \_secx \{\_seccx \_seccnum=0 \_tnum=0 \_fnum=0 \_dnum=0 \_resetABCDE \}
\def \_seccx \{}\def \_thechapnum \{\_the\_chapnum\}
\def \_thesecnum \{\_othe\_chapnum.\_the\_seccnum\}
\def \_theseccnum \{\_othe\_chapnum.\_the\_seccnum.\_the\_seccnum\}
\def \_thetnum \{\_othe\_chapnum.\_the\_seccnum.\_the\_tnum\}
\def \_thefnum \{\_othe\_chapnum.\_the\_seccnum.\_the\_fnum\}
\def \_thednum \{(\_the\_dnum)\}
\def \_othe #1. {\_ifnum#1>0 \_the#1. \_fi}
\def \_incr #1 {\_global\_advance #1 by 1}
\def \notoc \nonum ;
\chap, \sec and \secc macros are implemented here. The \_inchap, \_insec and \_insecc macros does the real work. First, we read the optional parameter \[⟨label⟩], if it exists. 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.
The \_printrefnum\{\_pre\}@\_post\} macro is used in \_print* macros. The \_wtotoc\{\_level\}\{\_info\}\{\_ref-num\}\{\_title-text\} macro expands its parameters and does \_wref. Note that the \_title-text\ is \_detokenized before \_wref, so the problem of "fragile macros" from old \_L\_T\_E\_X never occurs.

\_def \_printrefnum [#1@#2]{\_leavevmode % we must be in horizontal mode
\_ifnonum \_else #1\_therefnum #2\_fi
\_wlabel \_therefnum % references, if \_label\{\_ltext\}\ is declared
\_ifnotoc \_else \_incr \_tocrefnum
\_dest[\_toc:\_the\_tocrefnum]\%
\_wtotoc\{\_the\_sectionlevel\}\{\_secinfo\}%
\_therefnum\{\_detokenize\_ea\{\_savedtitle\}\}%
\_fi
}
\_def \_wtotoc #1#2#3#4{\_edef\_tmp{{#1}{#2}{#3}{#4}}\_ea\_wtotocA\_tmp}
\_def \_wtotocA #1#2#3#4{\_wref\_Xtoc{{#1}{#2}{#3}#4}}

The \_abovetitle\{\penalty\_A\}\{\skip\_A\} and \_belowtitle\{\skip\_B\} pair communicates using a special penalty 11333 in vertical mode. The \_belowtitle puts the vertical skip (its value is saved in \_savedtitleskip) followed by this special penalty. The \_abovetitle reads \_lastpenalty and if it has this special value then it removes the skip used before and don’t use the parameter. The \_abovetitle creates \{\skip\_A\} only if whatever previous skip is less or equal than \{\skip\_A\}. We must save \{\whatever-skip\} before \_belowtitle, remove it, create \{\penalty\_A\} (if \_belowtitle does not preceded) and create \{\whatever-skip\} or \{\skip\_A\} depending on what is greater. The amount of \{\skip\_A\} is measured using \setbox0=\vbox.

\_newskip \_savedtitleskip
\_newskip \_savedlastskip
\_def \_abovetitle #1#2{\_savedlastskip=\_lastskip % <whatever-skip>
\_ifdim\_lastskip>\zo \_vskip-\_lastskip \_fi
\_ifnum\_lastpenalty=11333 \_vskip-\_savedtitleskip \_else #1\_fi
\_ifdim\_savedlastskip=\zo \_setbox0=\vbox#2\_global\_tmpdim=\_lastskip\%
\_else \_tmpdim=\maxdimen \_fi
\_ifdim\_savedlastskip=\_tmpdim \_vskip\_savedlastskip \_else #2\_fi
}
\_def \_belowtitle #1{#1\_global\_savedtitleskip=\_lastskip % \_penalty11333 }

\nbpar sets \_interlinepenalty value. \_nl is “new line” in text (or titles), but space in toc or headlines or outlines.

\_def \_nbpar{\{\_interlinepenalty=10000\_endgraf}}
\_protected\_def \_nl{\_hfil\_break}
\regmacro \_def\_nl{\_unskip\_space}{\_def\_nl{}}
\regmacro \_def\_nl{\_unskip\_space}{\_def\_nl{}}
\_public \_nl \_nl ;

\_firstnoindent puts a material to \everypar in order to next paragraph will be without indentation. It is useful after titles. If you dislike this feature then you can say \_let\_firstnoindent=\relax. The \_wipeepar removes the material from \everypar.

\_def \_firstnoindent {\_global\everypar=\_wipeepar \_setbox7=\_lastbox}
\_def \_wipeepar {\_global\everypar=\_}

The \_mark (for running heads) is used in \_printsection only. We suppose that chapters will be printed after \_fill\_break, so user can implement chapter titles for running headers directly by macros, no \_mark mechanism is needed. But sections need \_marks. And they can be mixed with chapter’s running heads, of course.

The \_insertmark\{\_title-text\}\ saves \_mark in the format {\_title-num} \{\_title-text\}, so it can be printed “as is” in \headline (see the space between them), or you can define a formatting macro with two parameters for processing these data, if you need it.
OpTeX sets `\headline={} by default, so no running headings are printed. You can activate the running headings by following code, for example:

```latex
\addto\chappx {
  \edef\runningchap {\thechapnum: \unexpanded\ea\savedtitle}
  \def \formathead #1#2{\isempty{#1}\iffalse #1: #2\fi}
  \headline = {
    \ifodd \pageno
      \hfil \ea\formathead\firstmark{}{}
    \else
      Chapter: \runningchap \hfil
    \fi
  }
}
```

The \caption/{⟨letter⟩} uses \langleletter⟩num counter. The group opened by \caption is finalized by first \par from empty line or from \vskip or from \endinsert. The \_printcaption{⟨letter⟩} is called, it starts with printing of the caption.

The \_cskip macro inserts nobjeakable vertical space between caption and the object.

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).

\_noindent \wlabel{\thecapnum} \_{bf}\thecaptitle\_\thecapnum\_\enspace
\_narrowlastlinecentered\_\indent
\_let \_printcaptionf = \_printcaptiont \% caption of figures = caption of tables

The default format of \caption text is paragraph in block narrower by \_\indent and with the last line is centered. This setting is done by the \_\narrowlastlinecentered macro.

\_eqmark is processed in display mode (we add \_eqno primitive) or in internal mode when \_equaligno is used (we don’t add \_eqno).

\_numberedpar ⟨letter⟩\{⟨name⟩\} is implemented here.
The \texttt{\printnumberedpar} \texttt{\theXnum \{\textit{name}\}} opens numbered paragraph and prints it. The optional parameter is in \texttt{\_the\_opt}. You can re-define it if you need another design. \texttt{\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:

\begin{verbatim}
\def\printnumberedpar #1#2{\par
  \noindent \wlabel #1\
  \bf #2 \#1\istoksempty\opt\iffalse \space \the\opt \fi.}\space
  \ignorespaces
  }
\end{verbatim}

\begin{verbatim}
\def\numberedpar#1#2{\ea \ incr \ cname \counter#1\endcname
  \def\tmpa{#1}\def\tmpb{#2}\numberedparparam}
\optdef\numberedparparam[\]{%  \ea \printnumberedpar \cname \the\tmpnum \ea\endcname\ea{\tmpb}}
\end{verbatim}

\begin{verbatim}
\def\theAnum {\othe\_chapnum.\othe\_secnum.\the\_counterA}
\def\theBnum {\othe\_chapnum.\othe\_secnum.\the\_counterB}
\def\theCnum {\othe\_chapnum.\othe\_secnum.\the\_counterC}
\def\theDnum {\othe\_chapnum.\othe\_secnum.\the\_counterD}
\def\theEnum {\othe\_chapnum.\othe\_secnum.\the\_counterE}
\end{verbatim}

2.27 Lists, items

\begin{verbatim}
\def\begitems {Lists: begitems, enditems <2020-04-21>} % preloaded in format
\end{verbatim}

\begin{verbatim}
\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.
\end{verbatim}

The \texttt{\itemnum} is locally reset to zero in each group declared by \texttt{\begitems}. So nested lists are numbered independently. User can set initial value of \texttt{\itemnum} to another value after \texttt{\begitems} if he/she want. Each level of nested lists is indented by new \texttt{\indent} from left. Default item mark is \texttt{\printitem}. The \texttt{\begitems} runs \texttt{\aboveliskip} only if we are not near below a title, where a vertical skip is placed already and where the \texttt{\penalty} 11333 is. It activates * and defines it as \texttt{\startitem}. The \texttt{\enditems} runs \texttt{\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).
Various item marks are saved in \_item:<letter> macros. You can re-define them or define more such macros. The \style:<letter> does \_printitem=\_item:<letter> first, then \style:<letter> does \_printitem=\_item:<letter> when it is used and finally, \_startitem alias * uses \_printitem.

\_athename{⟨num⟩} returns the ⟨num⟩s lowercase letter from the alphabet. \_fullrectangle{⟨dimen⟩} prints full rectangle with given ⟨dimen⟩.

### 2.28 Verbatim, listings

#### 2.28.1 Inline and “display” verbatim

The internal parameters \_ttskip, \_ttpenalty, \_viline, \_vifile and \_ttfont for verbatim macros are set.
\texttt{\detokenize{⟨text⟩}} expands to \texttt{⟨text⟩} when \texttt{\escapechar=-1}. In order to do it more robust when it is used in \texttt{\write} then it expands as noexpanded \texttt{\code⟨text⟩} (followed by space in its csname). This macro does the real work.

The \texttt{\printinverbatim{⟨text⟩}} macro is used for \texttt{\code⟨text⟩} printing and for ‘⟨text⟩’ printing. It is defined as \texttt{\box}, so the in-verbatim ⟨text⟩ will be never broken. But you can re-define this macro.

When \texttt{\code} occurs in PDF outlines then it does the same as \texttt{\detokenize}. The macro for preparing outlines sets \texttt{\escapechar} to −1 and uses \_\texttt{regoul} token list before \_\texttt{edef}.

The \texttt{\code} is not \texttt{\protected} because we want it expands to \texttt{\unexpanded{\code⟨space⟩}} in \texttt{\write} parameters. This protect the expansions of the \texttt{\code} parameter (like \texttt{\}}, \texttt{\^} etc.).

The \_\texttt{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 \texttt{\catcodetable\verbatimcatcodes}. After the group is finished then original catcode table is restored.

\texttt{\activettchar⟨char⟩} saves original catcode of previously declared ⟨char⟩ (if such character was declared) using \texttt{\_\savedttchar} and \texttt{\_\savedttcharc} values. Then new such values are stored. The declared character is activated by \_\texttt{adef} as a macro (active character) which opens a group, does \_\texttt{setverb} and other settings and reads its parameter until second the same character. This is done by the \_\texttt{readverb} macro. Finally it prints scanned ⟨text⟩ by \_\texttt{printinverbatim} and closes group. Suppose that \texttt{\activettchar"} is used. Then the following work is schematically done:

\begin{verbatim}
\def "\begin{group} \_\texttt{setverb} ... \_\texttt{readverb} \end{group}
\end{verbatim}

Note that the second occurrence of " is not active because \_\texttt{setverb} deactivates it.

\texttt{\begtt} is defined only as public. We don’t need private \_\texttt{begtt} variant. This macro is defined by \_\texttt{edef}, so user can put a parameter at the same line where \_\texttt{begtt} is. This #1 parameter is used after \_\texttt{everytt} parameters settings, so user can change them locally.

The \_\texttt{begtt} macro opens group, does \_\texttt{setverb} and another preprocessing, sets \_\texttt{endlinechar} to "\_\texttt{J} and reads the following text in verbatim mode until \_\texttt{endtt} occurs. This scanning is done by \_\texttt{startverb} macro which is defined as:

\begin{verbatim}
\def \_\texttt{startverb} #1\_\texttt{endtt} #2"\_\texttt{J} {...}
\end{verbatim}
We must 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 `\startverb` and end of the same line and it is simply ignored.

The `\startverb` puts the scanned data to `\prepareverbdata`. It sets the data to `\tmpb` without changes by default, but you should re-define it in order to do special changes, if you want. (For example, `\hisyntax` redefines this macro.) The scanned data have `\^[` at each end of line and all spaces are active characters (defined as `\␣`). Other characters have normal category 11 or 12.

When `\prepareverbdata` finishes then `\startverb` runs `\printverb` loop over each line of the data and does a final work: last skip plus `\noindent` in the next paragraph.

```
verbatim.opm
117 \_eoldef \begtt#1{\_par \_wipeepar \_setxhsize
118 \vskip\parskip \ttskip
119 \begingroup \_setverb
120 \_ifnum\ttline<0 \_let\_printverblinenum=\relax \else \_initverblinenum \_fi
121 \_adef{ }\_parindent=\ttindent \parskip=0pt
122 \_def\t{\hskip \dimexpr\tabspaces em/2\relax}
123 \_the\everytt \relax #1\relax \ttfont
124 \_def\testcommentchars##1\_iftrue{\_iffalse}\_let\_hicomments=\relax
125 \_startverb
126 }
127 \_ea\def\ea\_startverb \_ea#1\_ea\csstring\endtt#2\^[}{
128 \_prepareverbdata\tmpb{#1\^[}\
129 \_ea\_printverb \_tmpb \_end
130 \_par
131 \_endgroup \_ttskip
132 \_isnextchar\par{}\_noindent\_fi
133 \_startverb
134 }
135 \_def\prepareverbdata\#1\#2\{\def\#1{\#2}
136 \_par
137 }
138 \_def\_printverb #1\^[#2\{\
139 \_ifx\_printverblinenum\relax \else \_global\_advance\_ttline by1 \_fi
140 \_testcommentchars \_def\_vcomments\empty\_else \_printcomments \_def\_vcomments{}\_fi
141 \_ifx\_end#2 \_printcomments\_fi
142 \_else
143 \_ifx\_vcomments\empty\_else \_printcomments \_def\_vcomments{}\_fi
144 \_ifx\_end#2
145 \_bgroup \_adef{ }\_def\t{}% if the last line is empty, we don’t print it
146 \_else
147 \_ifcat&#1&\_egroup \_else\egroup \_printverbline{#1}\_fi
148 \_else
149 \_printverbline{#1}\_fi
150 \_fi
151 \_fi
152 \_ifx\_end#2 \_let\_next=\_relax \_else \_def\_next{\_printverb#2}\_fi
153 \_next
154 }
155 \_def\_printverbline{\_penalty \_ttpenalty \_indent \_printverblineunum \_kern\_ttshift \_printverblineunum\_par
156 \_testcommentchars \_def\_printverblineunum{\_preventline}\
157 \_def\_printverblinenum{\_llap{\_sevenrm \_the\ttline \_kern.9em}}
```

Macro `\verbinput` uses a file read previously or opens the given file. Then it runs the parameter scanning by `\viscannparameter` and `\viscannminus`. Finally the `\doverbinput` is run. At beginning of `\doverbinput`, we have `\viline` = number of lines already read using previous `\verbinput`, `\vinolines` = the number of lines we need to skip and `\vidolnes` = the number of lines we need to print. Similar preparation is done as in `\begtt` after the group is opened. Then we skip `\vinolines`
lines in a loop a and we read \_vidolines lines. The read data is accumulated into \_tmpb macro. The next steps are equal to the steps done in \_startverb macro: data are processed via \_prepareverbdata and printed via \_printverb loop.
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 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 single command \verbinput (4-) optex.lua here and the \commentchars -- was set before it.

If you need to set a special character by \commentchars then you must set the catcode to 12 (and space to 13). Examples:

\commentchars // % C++ comments
\commentchars -- % Lua comments
\catcode\%=12 \catcode\%=9 % used in \commentchars comments
\ea\printverb \% used in \commentchars comments
\commentchars%% % TeX comments
\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 comments block does not work. The character \% is ignored in comments but you can use \% for printing or \% alone for de-activating \_endpar from empty comment lines.

Implementation: The \commentchars⟨first⟩⟨second⟩ redefines the \_testcommentchars used in \_printverb in order to it removes the following \_iftrue and returns \_iftrue or \_iffalse depending on the fact that the comment characters are or aren’t present at the beginning of tested line. If it is true (\_ifnum expands to \_ifnum 10>0) then the rest of the line is added to the \_vcomments macro.

The \_hicomments is \relax by default but it is redefined by \commentchars in order to keep no-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 main idea must be kept: it is printed in the group, \reloading \_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.
The \texttt{\_visiblesp} sets spaces as visible characters \texttt{␣}. It redefines the \texttt{\_dsp}, so it is useful for verbatim modes only.

The \texttt{\_dsp} is equivalent to \texttt{␣} primitive. It is used in all verbatim environments: spaces are active and defined as \texttt{\_dsp} here.

```verbatim
\_def \_visiblesp{\ifx \_initunifonts \relax \_def \_dsp{\_char9251 }\% \\
\_else \_def \_dsp{\_char32 }\_fi} \\
\_let \_dsp=\% primitive "direct space"
```

2.28.2 Listings with syntax highlighting

The user can write

\begin{verbatim}
... \end{verbatim}

and the code is colorized by C syntax. The user can write \texttt{\everytt={\hisyntax{C}}} and all verbatim listings are colorized.

The \texttt{\hisyntax{⟨name⟩}} reads the file \texttt{hisyntax-⟨name⟩.opm} where the colorization is declared. The parameter \texttt{⟨name⟩} is case insensitive and the file name must include it in lowercase letters. For example the file \texttt{hisyntax-c.opm} looks like:

```hisyntax-c.opm
\_codedecl \_hisyntaxc {Syntax highlighting for C sources <2020-04-03>} \\
\_newtoks \_hisyntaxc \_newtoks \_hicolorsc \\
\_global \_hicolorsc={% colors for C language \\
\_hicolor K \Red % Keywords \\
\_hicolor S \Magenta % Strings \\
\_hicolor C \Green % Comments \\
\_hicolor N \Cyan % Numbers \\
\_hicolor P \Blue % Preprocessor \\
\_hicolor D \Blue % Non-letters \\
} \\
\_global \_hisyntaxc={% \\
\_the \_hicolorsc \\
\_let\c=\_relax \_let\e=\_relax \_let\o=\_relax \\
\_replfromto /*\{\_string#}\} */ {\z K{#1}}% /*...*/ \\
\_replfromto (()\_string\_string\} &&\_string\_string\} } {\z P{\_string\_string\}}}% //... \\
\_replfromto {\_string"\} {\_string\_string\}}% \" protected inside strings \\
\_replfromto {\_string\_string\} {\z N{\_string\_string\}}}% \"..." \\
\% \\
\_edef \_tmpa {()\_string\_string\}+/*=/<>;::;\_pcent\_string\_string\}!?%} % non-letters \\
\_ea \_foreach \_tmpa \\
\_do {\z \_repthis{\_string\_string\}+/*=/<>;::;\_pcent\_string\_string\}!?% \_repthis{\_string\_string\}+/*=/<>;::;\_pcent\_string\_string\}!?%} % non-letters \\
\% numbers \\
\_foreach \_auto\break\case\char\continue\default\do\double\% \\
\_else\_entry\_enum\_extern\_float\_for\_goto\_if\_int\_long\_register\% \\
\_return\_short\_sizeof\_static\_struct\_switch\_typedef\_union\% \\
\_unsigned\void\while\% \\
\_do {\z \_repthis{\_string\_string\}+/*=/<>;::;\_pcent\_string\_string\}!?%} % non-letters \\
\% numbers \\
\_foreach 0123456789 \\
\_do {\z \_repthis{\_string\_string\}+/*=/<>;::;\_pcent\_string\_string\}!?%} % non-letters \\
\_replthis{\_string\_string\}+/*=/<>;::;\_pcent\_string\_string\}!?%} % non-letters
```

125
Op\TeX provides \texttt{hisyntax-\{c,python,\tex,html\}.opm} files. You can take inspiration from these files and declare more languages.

User can re-declare colors by \verb+\hicolors\{\ldots\}+ This value has precedence before \verb+\hicolors\{name\}+ values declared in the \texttt{hicolors-\{name\}.opm} file. What exactly to do: copy \verb+\hicolors\{name\}=\{\ldots\}+ from \texttt{hicolors-\{name\}.opm} to your document, rename it as \verb+\hicolors\{\ldots\}+ and do your own colors modifications.

Another way to set non-default colors is to declare \verb+\newtoks\hicolors\{name\}+ (without the \verb+_+ prefix) and set the colors palette here. It has precedence before \verb+\hicolors\{name\}+ (with the \verb+_+ prefix) declared in the \texttt{hicolors-\{name\}.opm} file. This is useful when there are more hi-syntax languages used in one document.

Notes for hi-syntax macro writers

The file \texttt{hisyntax-\{name\}.opm} is read only once in the \TeX group. If there are definitions then they must be declared as global.

The \texttt{hisyntax-\{name\}.opm} file must (globally) declare \verb+_hisyntax\{name\}_+ tokens string where the action over verbatim text is declared typically by \verb+_replfromto_+ or \verb+_replthis_+ macros.

The verbatim text is prepared by \texttt{pre-processing phase}, then the \verb+_hisyntax\{name\}_+ is applied and then \texttt{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 \verb+\\\\an+, so \verb+\\word\an+ should be a pattern finding whole words (no subwords). The \verb+\an+ sequence is removed in the post-processing phase.
- Each end of line is represented by \verb+\an^J\an+.
- The \verb+_start_+ control sequence is added before the verbatim text and \verb+_end_+ control sequence is appended to the end of the verbatim text. These control sequences are removed in post-processing phase.

There are special macros working only in a group when processing the verbatim text.

- \verb+\an+ means noting but it should be used as a boundary of words as mentioned above.
- \verb+\t+ means a tabulator. It is prepared as \verb+\\t\an+ because it can be at a boundary of a word.
- \verb+\x \langle letter \rangle \{\langle text \rangle\}+ can be used as replacing text. Suppose the example

\begin{verbatim}
\replfromto{/*}{*/}\{\x C{/*#1*/}\}
\end{verbatim}

This replaces all C comments \verb+/*...*/+ by \verb+\x C/*...*/+. But the C comments may span more lines, i.e. the \verb+^J+ should be inside it.

The macro \verb+\x \langle letter \rangle \{\langle text \rangle\}+ is replaced by one or more \verb+\z \langle letter \rangle \{\langle text \rangle\}+ in post-processing phase where each parameter \verb+\langle text \rangle+ of \verb+\z+ keeps inside one line. Inside-line parameters are represented by \verb+\x C\{\langle text \rangle\}+ and they are replaced to \verb+\z C\{\langle text \rangle\}+ without any change. But:

\begin{verbatim}
\x C\{\langle text1\rangle^J\langle text3\rangle^J\langle text3\rangle\}
\end{verbatim}

is replaced by

\begin{verbatim}
\z C\{\langle text1\rangle^{\langle text2\rangle}^\langle text2\rangle\}^\langle text2\rangle\z C\{\langle text3\rangle\}
\end{verbatim}

The \verb+\z \langle letter \rangle \{\langle text \rangle\}+ is expanded to \verb+\z:\langle letter \rangle \{\langle text \rangle\}+ and if \verb+\hicolor \langle letter \rangle \langle color \rangle+ is declared then \verb+\z:\langle letter \rangle \{\langle text \rangle\}+ expands to \verb+\{\langle color \rangle\{\langle text \rangle\}+). So, required color is activated at all lines (separately) where C comment spans.

- \verb+\y \{\langle text \rangle\}+ is replaced by \verb+\y\{\langle text \rangle\}+ in the post-processing phase. It should be used for macros without a parameter. You cannot use unprotected macros as replacement text before the post-processing phase, because the post-processing phase is based on expansion whole verbatim text.

The following macros \verb+replfromto_+ and \verb+replthis_+ manipulate with the verbatim text which has been read already and stored in the \verb+_tmpb_+ macro.

The \verb+replfromto \{\langle from \rangle\}{\{\langle to \rangle\} \{\langle what\rangle\}+ finds first \verb+\langle from \rangle+ then the first \verb+\langle to \rangle+ following by \verb+\langle from \rangle+ pattern and the \verb+\langle text \rangle+ between them is packed to \verb+\#1+. Then \verb+\langle from \rangle\{\langle text \rangle\} \{\langle to \rangle\}+ is replaced by \verb+\langle what\rangle\{\langle text \rangle\}+. The \verb+\langle what\rangle+ parameter can use \verb+\#1+ which is replaced by the \verb+\langle text \rangle+.

The \verb+replfromto+ continues by finding next \verb+\langle from \rangle+, then, next \verb+\langle to \rangle+ repeatedly over the whole verbatim text. If the verbatim text is ended by opened \verb+\langle from \rangle+ but not closing by \verb+\langle to \rangle+ then \verb+\langle to \rangle+ is appended to the verbatim text automatically and the last part of verbatim text is replaced too.
First two parameters are expanded before usage of \replfromto. You can use \csstring\% or something else here.

The \replthis \{<pattern>\} \{<what>\} replaces each \(<pattern>\) by \(<what>\). Both parameters of \replthis are expanded first.

The \replfromto\{<from>\} \{<to>\} and \(<pattern>\) are not found when they are hidden in braces {...}. Example:

```latex
\replfromto\{/*\} \{*/\} \{\x C/{/*#1/*}\}
```

replaces all C comments by \x C{...}. The patterns inside {...} are not used by next usage of \replfromto or \replthis macros.

The \_xscan macro does replacing \x by \z in the post-processing phase. The \x \{<letter>\} \{<text>\} expands to \xscan \{<letter>\} \{<text>\}~````. If \#3 is \end then it signals that something wrong happens, the \(<from>\) was not terminated by legal \(<to>\) when \replfromto did work. We must to fix it by the \_xscanR macro.

The \hicolor \{<letter>\} \{<color>\} defines \z \{<letter>\} \{<text>\} as \{<color>\} \{<text>\}. It should be used in the context of \x \{<letter>\} \{<text>\} macros.

The \hisyntax \{<name>\} re-defines default \_prepareverbdata \{<macro>\} \{<verbtext>\} in order to it does more things: It saves \{<verbtext>\} to \_tmpb, appends \_unrelax around spaces and ~```` characters in pre-processing phase, it opens \_hisyntax-<name>.opm file if \_hisyntax <name> is not defined. Then \_the \_isyntax <name> is processed. Finally, the post-processing phase is realized by setting appropriate values to \x and \y macros and doing \edef \_tmph \{\_tmph\}.
Aliases for languages can be declared like this. When `\hisyntax{xml}` is used then this is the same as `\hisyntax{html}`.

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:

```latex
\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 PDF 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}`.

Inkscape is able to save a picture to *.pdf file and labels for the picture to *.pdf_tex file. The second file is in \LaTeX format (unfortunately) and it is intended to read immediately it after *.pdf in included in order to place labels of this picture in the same font as document is printed. We need to read this \LaTeX \file by plain \TeX \macro 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.

```latex
\def\inkinspic{\hbox{\_bgroup\isnextchar\_bgroup\inkinspicB\_inkinspicA}}
\def\inkinspicB #1{\_inkinspic{\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}}
```

Inkscape is able to save a picture to *.pdf file and labels for the picture to *.pdf_tex file. The second file is in \LaTeX\ format (unfortunately) and it is intended to read immediately it after *.pdf in included in order to place labels of this picture in the same font as document is printed. We need to read this \LaTeX\ file by plain \TeX\ macros when `\inkinspic` is used. These macros are stored in the `\_inkdefs` tokens list and it is used locally in the group. The solution is borrowed from OPmac trick 0032.
\def\inkinspicA #1 { \_inkinspicB {#1}}
\def\inkinspicB #1{% 
\ifdim\picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
\_the\_inkdefs
\_opinput {\_the\_picdir #1_tex}% file with labels
\_egroup}

\newtoks\_inkdefs \_inkdefs={% 
\def\makeatletter#1\makeatother{}% 
\def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{#2}\_hss}%
\def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
\def\_beginpicture(#1,#2){\_vbox\_bgroup
\_hbox to\_picwidth{}\_kern#2\_picwidth \_def\end##1{\_egroup}}%
\def\_begintabular[#1]#2#3\end#4{\vtop{\_def\cr{\_cr}\_tabiteml{}\_tabitemr{}\_table{#2}{#3}}}%
\def\_sin{\_gonfunc{sin}}
\def\_cos{\_gonfunc{cos}}
\_pdfscale{⟨x-scale⟩}{⟨y-scale⟩} and \_pdfrotate{⟨degrees⟩} macros are implemented by \pdfsetmatrix primitive. We need to know values of sin, cos function in the \_pdfrotate. We use Lua code for this.
\def\_pdfscale#1#2{\_pdfsetmatrix{#1 0 0 #2}}
\edef\_gonfunc#1#2{\directlua{tex.print(string.format('\_pcent.4f',math.#1(3.14159265*(#2)/180)))}}
\def\_sin{\_gonfunc{sin}}
\def\_cos{\_gonfunc{cos}}
\_pdfrotate{⟨degrees⟩} 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.
\def\_rotatebox#1#2{\_pdfrotate{#1}#2}
\_public \inkinspic ;

\_public \_inkscape ;

\_def\_inkscape #1 \_inkscapeB {#1} % file with labels
\_def\_inkscapeB #1{% 
\_ifdim\picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
\_the\_inkdefs
\_opinput {\_the\_picdir #1_tex}% file with labels
\_egroup}

\_newtoks\_inkdefs \_inkdefs={% 
\def\makeatletter#1\makeatother{}% 
\def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{#2}\_hss}%
\def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
\def\_beginpicture(#1,#2){\_vbox\_bgroup
\_hbox to\_picwidth{}\_kern#2\_picwidth \_def\end##1{\_egroup}}%
\def\_begintabular[#1]#2#3\end#4{\vtop{\_def\cr{\_cr}\_tabiteml{}\_tabitemr{}\_table{#2}{#3}}}%
\_pdfrotate{⟨degrees⟩} 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.
\_public \_inkscape ;

\_def\_inkscape #1 \_inkscapeB {#1} % file with labels
\_def\_inkscapeB #1{% 
\_ifdim\picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
\_the\_inkdefs
\_opinput {\_the\_picdir #1_tex}% file with labels
\_egroup}

\_newtoks\_inkdefs \_inkdefs={% 
\def\makeatletter#1\makeatother{}% 
\def\includegraphics[#1]#2{\_inkscanpage#1,page=,\_end \_inspic{#2}\_hss}%
\def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,\_else\_def\_picparams{page#2}\_fi}%
\def\_beginpicture(#1,#2){\_vbox\_bgroup
\_hbox to\_picwidth{}\_kern#2\_picwidth \_def\end##1{\_egroup}}%
\def\_begintabular[#1]#2#3\end#4{\vtop{\_def\cr{\_cr}\_tabiteml{}\_tabitemr{}\_table{#2}{#3}}}%
\_pdfrotate{⟨degrees⟩} 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.
\_public \_inkscape ;
\_scantwodimens scans two objects with the syntactic rule \langle \text{dimen} \rangle and returns \{\langle \text{number} \rangle\}_{\langle \text{number} \rangle} in sp unit.

\texttt{\verb|\puttext (right) (up) (text)|} puts the \langle text \rangle to desired place: From current point moves \langle down \rangle and \langle right \rangle, puts the \langle text \rangle and returns back. The current point is unchanged after this macro ends.

\texttt{\verb|\putpic (right) (up) (width) (height) (image-file)|} does \texttt{\verb|\puttext|} with the image scaled to desired \langle width \rangle and \langle height \rangle. If \langle width \rangle or \langle height \rangle is zero, natural dimension is used. The \texttt{\verb|\nospec|} is a shortcut to such natural dimension.

\texttt{\verb|\backgroundpic|} \langle image-file \rangle puts the image to the background of each page. It is used in the slides style, for example.

\texttt{\verb|\directlua{\texttt{\verb|tex.print(string.format('{\_pcent d}{\_pcent d}', token.scan_dimen(),token.scan_dimen()))}}} \texttt{\verb|\puttext|} \texttt{\verb|\putpic|} \texttt{\verb|\backgroundpic|};
\_circle\{\langle x \rangle \langle y \rangle \} creates an ellipse with \langle x \rangle axis and \langle y \rangle axis. The origin is in the center.

\_oval\{\langle x \rangle \langle y \rangle \langle \text{roundness} \rangle \} creates an oval with \langle x \rangle, \langle y \rangle size and with given \langle \text{roundness} \rangle. The real size is bigger by 2\langle \text{roundness} \rangle. The origin is at the left bottom corner.

\_mv\{\langle x \rangle \langle y \rangle \langle \text{curve} \rangle \} moves current point to \langle x \rangle, \langle y \rangle, creates the \langle \text{curve} \rangle and returns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.

The \_circle\{\langle x \rangle \langle y \rangle \} is an example of \_oval usage. The \_circle\{\langle x \rangle \langle y \rangle \} is an example of \_circle usage.

The \_ratio, \_width, \_fcolor, \_lcolor, \_shadow and \_overlapmargins are parameters, they can be set by user in optional brackets \[\ldots\]. For example \_fcolor=\Red does \_let\_fcolorvalue=\Red and it means filling color.

The \_setfcolor uses the \_fillstroke macro to separate filling color and drawing color.
A shadow effect is implemented here. The shadow is equal to the silhouette of the given path in gray-transparent color shifted by \_shadowmoveto vector and with blurred boundary. A waistline with the width 2*\_shadowb around the boundary is blurred. The \shadowlevels levels of transparent shapes is used for creating this effect. The \shadowlevels+1/2 level is equal to the shifted given path.

\pdfpageresources primitive is used to define transparency. It does not work when used in a box. So, we use it at the begining of the output routine. The modification of the output routine is done using \_insertshadowresources only once when the shadow effect is used first.
The \texttt{\_doshadow\{\texttt{\textit{curve}}\}} does the shadow effect.

A generic macro \texttt{\_clipinpath\{\texttt{x}\} \{\texttt{y}\} \{\texttt{curve}\} \{\texttt{text}\}} declares a clipping path by the \texttt{\{\texttt{curve}\}} shifted by the \texttt{\{\texttt{x}\}}, \texttt{\{\texttt{y}\}}. The \texttt{\{\texttt{text}\}} is typeset when such clipping path is active. Dimensions are given by bp without the unit here. The macros \texttt{\_clipinoval\{\texttt{x}\} \{\texttt{y}\} \{\texttt{width}\} \{\texttt{height}\} \{\texttt{text}\}} and \texttt{\_clipincircle\{\texttt{x}\} \{\texttt{y}\} \{\texttt{width}\} \{\texttt{height}\} \{\texttt{text}\}} are defined here. These macros read normal \TeX\ dimensions in their parameters.
2.30 The $\texttt{table}$ macro, tables and rules

2.30.1 The boundary declarator:
The $\langle\text{declaration}\rangle$ part of $\texttt{table}\{\langle\text{declaration}\rangle\}\{\langle\text{data}\rangle\}$ includes column declarators (letters) and other material: the $\mid$ or $\langle\text{cmd}\rangle$. If the boundary declarator $\mid$ is not used then the boundaries of columns are just before each column declarator with exception of the first one. For example, the declaration $\{l|c(xx)(yy)c\}$ should be written more exactly using the boundary declarator $\mid$ explicitly, for example $\{lc|l(c(xx)(yy)c\}$. The boundary declarator $\mid$ can be used only once between each two column declarators.

Each table item has its own group. The $\langle\text{cmd}\rangle$ are parts of the given table item (depending on the boundary declarator position). If you want to apply a special setting for given column, you can do this by $\langle\text{setting}\rangle$ followed by column declarator. But if such column is not first, you must use $\langle\text{setting}\rangle$. Example. We have three centered columns, the second one have to be in bold font and the third one have to be in red: $\texttt{table}\{c:(\textbf{bf})c:(\textbf{Red})c\}\{\langle\text{data}\rangle\}$

2.30.2 Usage of the $\texttt{tabskip}$ primitive
The value of $\texttt{tabskip}$ 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, $\texttt{tabskip}$ is 0pt. It means that only $\texttt{tabiteml}$, $\texttt{tabitemr}$ and $\langle\text{cmds}\rangle$ can generate visual spaces between columns. But they are not real spaces between columns because they are in fact the part of the total column width.

The $\texttt{tabskip}$ value declared before the $\texttt{table}$ macro (or in $\texttt{everytable}$ or in $\texttt{thistable}$) is used between all columns in the table. This value is equal for all spaces between columns. But you can set each such space individually if you use $\langle\text{tabskip=\langle value\rangle}\rangle$ in the $\langle\text{declaration}\rangle$ immediately before boundary character. The boundary character represents the column pair for which the $\texttt{tabskip}$ have individual value. For example $c(\texttt{tabskip=5pt}):r$ gives $\texttt{tabskip}$ value between $c$ and $r$ columns. You need not to use boundary character explicitly, so $c(\texttt{tabskip=5pt})r$ gives the same result.

The space before first column is given by the $\texttt{tabskipl}$ and the space after last column is equal to $\texttt{tabskipr}$. Default values are 0pt.

Use nonzero $\texttt{tabskip}$ only in special applications. If $\texttt{tabskip}$ is nonzero then horizontal lines generated by $\texttt{\crli}$, $\texttt{\crlili}$ and $\texttt{\crlp}$ have another behavior than you probably expected: they are interrupted in each $\texttt{tabskip}$ space.

2.30.3 Tables to given width
There are two possibilities how to create tables to given width:

- $\texttt{\table to\langle size\rangle}\{\langle\text{declaration}\rangle\}\{\langle\text{data}\rangle\}$ uses stretchability or shrinkability of all spaces between columns generated by $\texttt{tabskip}$ value and eventually by $\texttt{tabskipl}$, $\texttt{tabskipr}$ values. See example below.
- $\texttt{\table pxto\langle size\rangle}\{\langle\text{declaration}\rangle\}\{\langle\text{data}\rangle\}$ expands the columns declared by $\texttt{p\langle\langle size\rangle\rangle}$, if the $\langle size\rangle$ is given by a virtual $\texttt{tsize}$ unit. See example below.

Example of $\texttt{\table to\langle size\rangle}$:

\begin{verbatim}
\thistable\{\texttt{tabskip=0pt plus1fil minus1fil}\}
\table to\hsize {lr}{\langle\text{data}\rangle}
\end{verbatim}

This table has its width $\texttt{\hsize}$. First column starts at the left boundary of this table and it is justified left (to the boundary). Second column ends at the right boundary of the table and it is justified right (to the boundary). The space between them are stretchable and shrinkable in order to reach given width $\texttt{\hsize}$.

Example of $\texttt{\table pxto\langle size\rangle}$ (means “paragraphs expanded to”):

\begin{verbatim}
\table pxto\hsize {lc|p\langle tsize\rangle}{\texttt{crl}}
\end{verbatim}

134
The first column is variable width (it gets the width of most wide item) and the resting space to given \hspace is filled by the p column.

You can declare more than one p\{\langle coefficient\rangle\times size\} columns in the table when pxto keyword is used. The total sum of \langle coefficients\rangle must be exactly one. For example,

\begin{verbatim}
\table pxto13cm \{r p\{.3\times size\} p\{.5\times size\} p\{.2\times size\} l\}{\langle data\rangle}
\end{verbatim}

This gives the ratio of widths of individual paragraphs in the table.

2.30.4 \eqbox: boxes with equal width across the whole document

The \eqbox \{\langle label\rangle\{\langle text\rangle\} behaves like \hbox\{\langle text\rangle\} in the first run of 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 \{\langle label\rangle\{\langle text\rangle\hspace\{\langle text\rangle\hspace\}\} where \langle dim:label\rangle is the maximum width of all boxes labeled by the same \langle label\rangle. The documentation of the \LaTeX\ package eqparbox includes more information and tips.

The optional parameter r or l can be written before \{\langle label\rangle\} (for example \eqbox r\{\langle label\rangle\{\langle text\rangle\}) 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 second one.

\begin{verbatim}
def\leftitem#1{\par
 \noindent \hangindent=\eqboxsize\{items\}\{2em\}\hangafter=1
 \eqbox r\{items\}\{#1\}\ignorespaces}
def\leftitem #1\par
 \noindent \hangindent=\eqboxsize\{items\}\{2em\}\hangafter=1
 \eqbox r\{items\}\{#1\}\ignorespaces
\end{verbatim}

2.30.5 Implementation of the \table macro and friends

\begin{verbatim}
\_newifi \_ifpxto
 \_def\_table#1#{\_tablebox\_bgroup \_tableW#1\_empty\_end
 \_bgroup \_catcodetable\_optexcatcodes \_tableA}
 \_def\_tableW#1#2\_end{\_pxtofalse
 \_ifx#1\_empty \_def\_tablew{}\_else
 \_ifx#1p \_def\_tablew{}\_tableWx#2\_end \_else \_def\_tablew{#1#2}\_fi\_fi
 \_def\_tableWx xto#1\_end{\_tmpdim=#1\_relax \_pxtotrue}
 \_public \table ;
\end{verbatim}

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.

135
The \tableA\{\langle declaration\rangle\} macro scans the \langle declaration\rangle by \scantabdata\#1\_relax and continues by reading \langle\{data\}\rangle by the \tableB macro.

\tableA
\_def\tableA#1\_egroup
\_the\_thistable \_global\_thistable=()% 
\_ea_ifx\_ea\_the\_tabstrut\_setbox\_tstrutbox=\_null 
\_else \_setbox\_tstrutbox=\_hbox\{\_the\_tabstrut\}
\_setbox\_tstrutbox=\_hbox\{\_vrule width\zo
\_height\_dimexpr\_ht\_tstrutbox+\_tablinespace
\_depth\_dimexpr\_dp\_tstrutbox+\_tablinespace\}
\_offinterlineskip
\_lineskip=-2\_tablinespace
\_fi
\_colnum=0 \_let\_addtabitem=\_addtabitemx
\_def\tmpa{}\_tabdata={\_colnum1\_relax}\_scantabdata\#1\_relax
\_the\_everytable \_tableB
\_fi
\_setbox\_tstrutbox=\_hbox{
\_vrule width\zo
\_height\_dimexpr\_ht\_tstrutbox+\_tablinespace
\_depth\_dimexpr\_dp\_tstrutbox+\_tablinespace}\_offinterlineskip
\_lineskip=-2\_tablinespace
\_fi
\_currentfeature\_colnum=0
\_let\_addtabitem=\_addtabitemx
\_def\tmpa{}\_tabdata={\_colnum1\_relax}\_scantabdata\#1\_relax
\_the\_everytable \_tableB
\_fi}

The \tableB\{\langle data\rangle\} saves \langle data\rangle to \_tmpb and does four \replstrings to prefix each macro \crl (etc.) by \_crcr. The reason is: we want to use macros which scan its parameter to the delimiter written in right part of table item declaration. See \_\fS for example. The \_crcr cannot be hidden in other macro in such case.

The \_tabskip value is saved for places between columns into the \_tabskipmid macro. Then it runs

\_tabskip=\_tabskipi\_halign\{\langle converted declaration\rangle\}\tabskip=\_tabskipr \cr \{\langle data\rangle\}\crcr

This sets the desired boundary values of \_tabskip. The “between-columns” values are set as \_tabskip=\_tabskipmid in the \langle converted declaration\rangle immediately after each column declarator.

If pxto keyword was used, then we set the virtual unit \_tsize to \_hsizestart first. Then the first attempt of the table is created in box 0. Then the \_tsize is re-calculated using \_wd0 and the real table is printed by \_halign in the second pass.

If no pxto keyword was used, then we print the table using \_halign directly. The \tableB macro is nonempty if the to keyword was used.

Because the color selector with \_aftergroup can be used inside the table item, we must to create second real group for each table item. This is reason why we start \_\fS in the \_\tableC macro. Each \_& character is stored as \_&\_\aftergroup\_\_& in \langle converted declaration\rangle. The \_halign\_\tableref\_\tableC really does:

\_halign\_\tableref\_\tableC\_def\_tableC\{\_ea{\_ea\_bgroup\_the\_tabdata\_egroup\_tabskip=\_tabskipr \cr \_tmpb\_crcr} \_egroup\_tables(\_\tablew\_\tableC)

\_newbox\_tstrutbox % strut used in table rows
\_newtoks\_tabdata % the \_halign declaration line

The \scantabdata macro converts \table’s \langle declaration\rangle to \_halign \langle converted declaration\rangle. The result is stored into \_tabdata tokens list. For example, the following result is generated when \langle declaration\rangle=\_\fS:\_fS.

\_halign\_\tableref\_\tableC\_def\_tableC\{\_ea{\_ea\_bgroup\_the\_tabdata\_egroup\_tabskip=\_tabskipr \cr \_tmpb\_crcr}
\_newbox\_tstrutbox % strut used in table rows
\_newtoks\_tabdata % the \_halign declaration line
\_halign\_\tableref\_\tableC

136
The second result in the `\ddlinedata` macro is a template of one row of the table used by `\crlim` macro.

The `\addtabitemx` adds the boundary code (used between columns) to the ⟨converted declaration⟩. This code is `\begin{table}\&\begin{data}⟨declaration⟩\end{data}\end{table}`. 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:

\begin{verbatim}
\def\showcolnum{% \ea\def\ea\totcolnum\ea{\the\colnum}\the\colnum/\totcolnum}%\table{ccc}{\showcolnum & \showcolnum & \showcolnum}
\end{verbatim}

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.

This code converts `\ |` or `\ |` from `\table ⟨declaration⟩` to the ⟨converted declaration⟩.

The default “declaration letters” c, l, r and p are declared by setting `\tabdeclarec`, `\tabdeclarall`, `\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.
User puts optional spaces around the table item typically, i.e. he/she writes & text & instead &text&.

The left space is ignored by internal TeX algorithm but the right space must be removed by macros.

This is a reason why we 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&.

The family of \crl*, \crll*, \crli* and \crlli* macros only does a special parameters settings for paragraph building algorithm. The \FS* prints the paragraph into box 0 first, measures the number of lines by the \prevgraf primitive and use (or don’t use) \hfil (for centering) before the first line.

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.
The \texttt{\vspan{⟨number⟩}{⟨text⟩}} implementaiton is here. We need to lower the box by \(⟨\text{number}⟩-1\)\*(\ht+\dp of \tabstrut) / 2. The #1 parameter must be one-digit number. If you want to set more digits then use braces. 

The parameters of primitive \texttt{\vrule} and \texttt{\hrule} keeps the rule “last wins”. If we re-define \texttt{\hrule} to \texttt{\_orihrule height1pt} then each usage of redefined \texttt{\hrule} uses 1pt height if this parameter isn’t overwritten by another following \texttt{height} parameter. This principle is used for settings another default rule thickness than 0.4pt by the macro \texttt{\rulewidth}. 

The \texttt{\frame{⟨text⟩}} uses “\texttt{vbox in vtop}” trick in order to keep the baseline of the internal text at the same level as outer baseline. User can write \texttt{\frame{abcxyz}} in normal paragraph line, for example and gets the expected result: \texttt{abcxyz}. The internal margins are set by \texttt{\vvkern} and \texttt{\hhkern} parameters. 

\texttt{\eqbox} and \texttt{\eqboxsize} are implemented here. The widths of all \texttt{\eqbox}es are saved to the .ref file in the format \texttt{\_Xeqbox{⟨label⟩}{⟨size⟩}}. The .ref file is read again and maximum box width for each \texttt{⟨label⟩} is saved to \texttt{\_eqb:⟨label⟩}. 

\section{Balanced multi-columns} 

This code is documented in detail in the “\texttt{\TeX}book naruby”, pages 244–246, free available, \url{http://petr.olsak.net/tbn.html}, but in Czech. Roughly speaking, macros complete all material between

\subsection*{2.31 Balanced multi-columns}
\begmulti⟨num-columns⟩ and \endmulti into one \vbox 6. Then the macro measures the amount of free space at the current page using \pagegoal and \pagetotal and does \vsplit of \vbox 6 to columns with height of such free space. This is done only if we have enough amount of material in \vbox 6 to fill full page by columns. This is repeated in loop until we have less amount of material in \vbox 6. Then we run \_balancecolumns which balances the last part of columns. Each part of printed material is distributed to main vertical list as \hbox{⟨columns⟩} and we need not do any change in the output routine.

If you have paragraphs in \begmulti... \endmulti environment then you may say \raggedright inside this environment and you can re-assign \widowpenalty and \clubpenalty (they are set to 10000 in OpTEX).

```
multicolumns.opm
\_def\_multiskip\{\_medskip\} % space above and below \begmulti...\endmulti
\_newcount\_mullines
\_def\_begmulti #1 \{\_par\_bgroup\_wipepar\_multiskip\penalty0 \_def\_Ncols{#1}
\_setbox6=\vbox\_bgroup \_let\_setxhsize=\_relax \_penalty0
\hsize := column width = (\hsize+\colsep) / n - \colsep
\_divide\_hsize by\_Ncols \_advance\_hsize by- \_colsep
\_mullines=0
\_def\par{\_ifhmode\_endgraf\_global\_advance\_mullines by\_prevgraf\_fi}%
\_def\_endmulti{\_vskip-\_prevdepth\_vfil
\_ea\_egroup\_ea\_baselineskip\_the\_baselineskip\relax
\_dimen0=.8\_maxdimen \_tmpnum=\_dimen0 \_divide\_tmpnum by\_baselineskip
\_splittopskip=\baselineskip
\_setbox1=\vsplit6 to\_opt
\_ifdim\_pagegoal=\maxdimen \_dimen1=\_vsize \_corrsize\{\_dimen1\}
\_else \_dimen1=\_pagegoal \_advance\_dimen1 by\_\pagetotal \_fi
\_ifdim\_dimen1<2\_baselineskip
\_vfil\_break \_dimen1=\_vsize \_corrsize\{\_dimen1\} \_fi
\_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=.8\_maxdimen \_fi
\_divide\_dimen0 by\_Ncols \_splitpart
\_else \_balancecolumns \_fi % only balancing
\_multiskip\_egroup
\}
\_def\_endmulti\{\_vskip\_prevdepth\_vfil
\_ea\_egroup\_ea\_baselineskip\_the\_baselineskip\relax
\_dimen0=.8\_maxdimen \_tmpnum=\_dimen0 \_divide\_tmpnum by\_baselineskip
\_splittopskip=\baselineskip
\_setbox1=\vsplit6 to\_dimen1 \_hss
\_loop \_ifnum\_Ncols>\_tmpnum
\_advance\_tmpnum by1
\_setbox1=\hbox\{\_unhbox1 \_vsplit6 to\_dimen1 \_hss\}
\_repeat
\_hbox\{\_nobreak\_vskip\_splittopskip \_nointerlineskip
\_line\{\_unbox1\_unskip\}
\_dimen0=\_dimen1 \_divide\_dimen0 by\_Ncols \_splitpart
\_else \_balancecolumns \_fi % only balancing
\_multiskip\_egroup
\}
\_def\_makecolumns\{\_bgroup % full page, destination height: \dimen1
\_loop \_ifnum\_Ncols>\_tmpnum
\_advance\_tmpnum by1
\_setbox1=\hbox\{\_unhbox1 \_vsplit6 to\_dimen1 \_hss\}
\_repeat
\_hbox\{\_nobreak\_vskip\_splittopskip \_nointerlineskip
\_line\{\_unbox1\_unskip\}
\_dimen0=\_dimen1 \_divide\_dimen0 by\_Ncols \_splitpart
\_else \_balancecolumns \_fi % only balancing
\_multiskip\_egroup
\}
\_def\_splitpart\{%
\_makecolumns % full page
\_vskip \opt plus \_ifil minus \_baselineskip \_break
\_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=\_maxdimen \_fi
\_divide\_dimen0 by\_Ncols \_relax
\_ifx\_balancecolumns\_flushcolumns \_divide\_dimen0 by-.5\_vsize \_fi
\_dimen1=\_vsize \_corrsize\{\_dimen1\} \_dimen2=\_dimen1
\_ifdim\_dimen2<\_baselineskip
\_splitpart\fi
\_else \_balancecolumns \_fi % last balancing
\_fi \_fi
```

Splitting columns...

```
multicolumns.opm
\_def\_makecolumns\{\_bgroup % full page, destination height: \dimen1
\_loop \_ifnum\_Ncols>\_tmpnum
\_advance\_tmpnum by1
\_setbox1=\hbox\{\_unhbox1 \_vsplit6 to\_dimen1 \_hss\}
\_repeat
\_hbox\{\_nobreak\_vskip\_splittopskip \_nointerlineskip
\_line\{\_unbox1\_unskip\}
\_dimen0=\_dimen1 \_divide\_dimen0 by\_Ncols \_splitpart
\_else \_balancecolumns \_fi % only balancing
\_multiskip\_egroup
\}
\_def\_splitpart\{%
\_makecolumns % full page
\_vskip \opt plus \_ifil minus \_baselineskip \_break
\_ifnum\_mullines<\_tmpnum \_dimen0=\_ht6 \_else \_dimen0=\_maxdimen \_fi
\_divide\_dimen0 by\_Ncols \_relax
\_ifx\_balancecolumns\_flushcolumns \_divide\_dimen0 by-.5\_vsize \_fi
\_dimen1=\_vsize \_corrsize\{\_dimen1\} \_dimen2=\_dimen1
\_ifdim\_dimen2<\_baselineskip
\_splitpart\fi
\_else \_balancecolumns \_fi % last balancing
\_fi \_fi
```

140
2.32 Citations, bibliography

2.32.1 Macros for citations and bibliography preloaded in the format

cite-bib.opm

\codedef \cite {Cite, Bibliography <2020-03-09>} % loaded in format

\newcount \bibnum % the bibitem counter
\newtoks \bibmark % the bibmark used if \nonumcitations is set.
\newcount \lastcitenum \lastcitenum=0 % for \shortcitations
\public \cite \noncit \rcite \ecite ;

\cite [[\label\],\label\],...,[\label\]] manages \langle labels \rangle using \_citeA and prints \{\langle bib-marks \rangle\} using \_printsavedcites.
\nocite [[\label\],\label\],...,[\label\]] only manages \langle labels \rangle but prints nothing.
\rcite [[\label\]] behaves like \cite but prints \{\langle bib-marks \rangle\} without brackets.
\ecite [[\label\]]\{\langle text \rangle\} behaves like \rcite [[\label\]] but prints \{\langle text \rangle\} instead \{\langle bib-marks \rangle\}. The \langle text \rangle is hyperlinked like \langle bib-marks \rangle when \cite or \rcite is used. The empty internal macro \_savedcites will include the \langle bib-marks \rangle list to be printed. This list is set by \_citeA inside group and it is used by \_printsavedcites in the same group. Each \cite/\rcite/\ecite macro starts from empty list of \langle bib-marks \rangle because new group is opened.

\_def \citeA \{\_printsavedcites\}
\_def \rcite \{\_printsavedcites\}
\_def \ecite \{\_printsavedcites\}
\_public \cite \noncit \rcite \ecite ;

\langle bib-marks \rangle may be numbers or a special text related to cited bib-entry. It depends on \nonumcitations and on used bib-style. The mapping from \langle label \rangle to \langle bib-mark \rangle is done when \bib or \usebib is processed. These macros store the information to \_Xbib\{\langle label \rangle\}\{\langle number \rangle\}\{\langle nonumber \rangle\} where \langle number \rangle and \langle nonumber \rangle are two variants of \langle bib-mark \rangle (numbered or text-like). This information is read from \.ref file and it is saved to macros \_bib:\{\langle label \rangle\} and \_bibm:\{\langle number \rangle\}. First one includes number and
second one includes (nonumber). The \_lastbibnum macro includes last number of bib-entry used in the document. A designer can use it to set appropriate indentation when printing the list of all bib-entries.

\_citeA \{label\}, processes one label from list of labels given in the parameter of \cite, \nocite, \rcite or \ecite macros. It adds the \{label\} to global list \_citelist which will be used by \usebib (it must to know what \{labels\} are used in the document in order to pick-up only relevant bib-entries from the database. Because we want to save space and not to save the same \{labels\} to \_citelist twice, we distinguish four cases:

- \{label\} was not declared by \_Xbib and it is first such \{label\} in the document: Then \_bib:⟨\{label\}⟩ is undefined and we save label using \_addcitlist, write warning on the terminal and define \_bib:⟨\{label\}⟩ as empty.
- \{label\} was not declared by \_Xbib but it was used previously in the document: Then \_bib:⟨\{label\}⟩ is empty and we do nothing (only data to \_savedcites are saved).
- \{label\} was declared by \_Xbib and it is first such \{label\} in the document: Then \_bib:⟨\{label\}⟩ includes \bibnn{⟨\{number\}⟩} & and we test this case by \if &\_bibnn{⟨\{number\}⟩}. This is true when \_bibnn{⟨\{number\}⟩} expands to empty. The \{label\} is saved by \_addcitlist and \_bib:⟨\{label\}⟩ is re-defined directly as \{number\}.
- \{label\} was declared by \_Xbib and it was used previously in the document. Then we do nothing (only data to \_savedcites are saved).

The \_citeA macro runs repeatedly over whole list of \{labels\}.

\_def \_citeA #1\{#2\}, \{\if #1, \else \fi \}
\_def \_addcitlist\{\#1\#2\}
\_def \_bibnn\{\#1\#2\}
\_def \_addcitlist\{\#1\#2\}
\_def \_sortcitations\{\_edef \_savedcites\{300000,\_ea\}\_ea\_sortcitesB \_savedcites,\_fi\}
\_def \_sortcitations\{\_edef \_savedcites\{300000,\_ea\}\_ea\_sortcitesB \_savedcites,\_fi\}
\_def \_printedsavedcites\{\_sortcitations\}
\_def \_sortcitations\{\_edef \_savedcites\{300000,\_ea\}\_ea\_sortcitesB \_savedcites,\_fi\}

The \{bib-marks\} (in numeric or text form) are saved in \_savedcites macro separated by commas. The \_printedsavedcites prints them by normal order or sorted if \_sortcitations is specified or condensed if \_shortcitations is specified.

The \_sortcitations appends the dummy number 300000 and we suppose that normal numbers of bib-entries are less than this constant. This constant is removed after sorting algorithm. The \_shortcitations sets simply \_lastcitenum=1. The macros for \{bib-marks\} printing follows (sorry, without detail documentation). They are documented in opmac-d.pdf (but only in Czech).
\def\sortcitesB #1, {\if $#1$ \else \mathchardef\tmpa=#1 \edef\savedcites{\ea\sortcitesC \savedcites\end} % fi \fi} \def\sortcitesC #1, {\ifnum\tmpa<#1 \edef\tmpa{\the\tmpa,#1} \ea\sortcitesD \else \edef\savedcites{\savedcites#1,} \ea\sortcitesC \fi} \def\sortcitesD #1 \end {\edef\savedcites{\savedcites\tmpa,#1}} \def\citeB #1, {\if$#1$ \else \if?#1\relax??% \else \ifnum\lastcitenum=0 % only comma separated list \printcite{#1} % \fi \fi \fi \ifx\citesep\empty % first cite item \lastcitenum=1 \relax \printcite{#1} % \else % next cite item \advance\lastcitenum by1 \ifnum\lastcitenum=\relax \lastcitenum=#1 \relax \printcite{#1} % \else % there is a gap between cite items \lastcitenum=\relax \ifnum\tmpb=0 % previous items were printed \printcite{#1} % \else % there is a gap between cite items \lastcitenum=#1 \relax \ifnum\tmpb=0 % previous items were printed \printdashcite{\the\tmpb} \printcite{#1} \chardef\tmpb=0 \fi \fi\fi\fi\fi\fi\fi \ea\citeB \fi \def\shortcitations {\lastcitenum=1 } \def\printcite #1 {\citesep\ilink[cite:#1]{\citelinkA{#1}}} \def\printdashcite #1 {\ifmmode-\else--\fi\ilink[cite:#1]{\citelinkA{#1}}} \def\citesep {} \def\nonumcitations {\lastcitenum=0 \def\sortcitesA{} \def\etalchar##1{$^{##1}$} \def\citelinkA##1 {\isdefined{_bim:##1}\iftrue \csname _bim:##1\endcsname \else ##1 \opwarning{\noexpand\nonumcitations \empty bibmark. Maybe bad bib style}\fi} \def\citelinkA {} \public \nonumcitations \sortcitations \shortcitations ;

The \bib{} \{\label{} \optionalbibmark\} prints one bib-entry without reading any database. The bib-entry follows after this command. This command counts the used \bib items from one \bibnum counter and saves \bibitem{} \{\the\bibnum\} \{\the\bibitem\} into .ref file immediately using \wbib. This is the core of creation of mapping from \{\labels\} to \{\bibmarks\}. cite-bib.om
The \_printbib prints the bib-entry itself. You can re-define it if you want different design. The \_printbib starts in horizontal mode after \noindent and after the eventual hyperlink destination is inserted. By default, the \_printbib sets the indentation by \hangindent and prints numeric ⟨bib-marks⟩ by \llap{⟨the\bibnum⟩} if \nonumcitations then the \_citelinkA is not empty and ⟨bib-marks⟩ ⟨the\bibnum nor \the\bibmark⟩ are not printed. The text of bib-entry follows. User can create this text manually using \bib command or it is generated automatically from a .bib database by \usebib command.

The vertical space between bib-entries is controlled by \bibskip macro. The \usebib command is implemented in usebib.opm file which is loaded when the \usebib command is firstly used. The usebib.opm file loads the librarian.tex for scanning the .bib files. See the section 2.32.2, where the file usebib.opm is documented.

\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.

The macros above works if all \cite (or similar) commands are used before the \usebib command is used because \usebib prints only such bib-entries their ⟨labels⟩ are saved in the \citelist. But if some \cite is used after usebib, then usebib sets \addcitelist to \_writeXcite, so such \cite saves the information to the .ref file in the format ⟨\_Xcite{⟨label⟩}⟩. Such information are copied to \citelistB during reading .ref file and \usebib concatenates two lists of ⟨labels⟩ from \citelist and \citelistB and uses this concatenated list.

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 bibliographics references (only those declared by by \cite[] or \nocite[] in the text). The formatting of such references is defined in the style file.

The principle “first entry wins” is used. Suppose \usebib/s (simple) local,global. If an entry with the same label is declared in local.bib and in global.bib too then the first wins. So, you can set an exceptions in your local.bib file for your document.

The bib-⟨⟨style⟩⟩.opm declares entry types (like @BOOK, @ARTICLE) and declares their mandatory and optionals fields (like author, title). When mandatory field is missing in an entry in the .bib file then warning is printed on the terminal about it. You can suppress such warnings by command \nobibwarning [⟨bib-labels⟩], where ⟨bib-labels⟩ is comma separated list of labels (without spaces) where missing mandatory fields will be no warned.

Old .bib files may use 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 \langle entry-types \rangle and \langle field-names \rangle are case insensitive.

Ancient Bib\TeX{} has read such files and has generated files appropriate for reading by \LaTeX{}. It has worked with a set of \langle entry-types \rangle, see the www page http://en.wikipedia.org/wiki/Bib\TeX{}. The set of entry types listed on this www page is de facto the Bib\TeX{} standard. The Op\TeX{} 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 previous sentence means that bib style writer must define the printing rules for each \langle entry-type \rangle. The printing rules for \langle entry-type \rangle include: which fields will be printed, in what order, by what format they will printed on (italic, caps, etc.), which fields are mandatory, which are optional and which are ingnored 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 \langle entry-type \rangle must be declared by \_\_\_sdef{}\print:(entry-type)\_\_\_ in bib-\langle style\rangle.opm file. The \langle entry-type \rangle have to be lowercase here. Op\TeX{} supports following macros for more comfortable setting of printing rules:

- \_\_\_bprinta{}\[\langle field-name\rangle\] \{\langle if defined \rangle\} \{\langle if not defined \rangle\}. The part \langle if defined \rangle is executed if \langle field-name \rangle is declared in .bib file for the entry which is currently processed. Else the part \langle if not defined \rangle is processed. The part \langle if defined \rangle can include the * parameter which is replaced by the value of \langle field-name \rangle.
- The part \langle if not defined \rangle can include the \_\_\_bibwarning command if the \langle field-name \rangle is mandatory.
- \_\_\_bprintb{}\[\langle field-name\rangle\] \{\langle if defined \rangle\} \{\langle if not defined \rangle\}. The same as \_\_\_bprinta, but the #1 parameter is used instead *. Differences: #1 parameter can be used more than once and can be enclosed in nested braces. The * parameter can be used at most once and cannot be enclosed in braces. Warning: if the \_\_\_bprintb commands are nested (\_\_\_bprintb in \_\_\_bprintb), then you need to write #1 parameter for internal \_\_\_bprintb. But if \_\_\_bprinta commands are nested then the parameter is not duplicated.
- \_\_\_bprintc \_\_\_macro {}\{\langle if non-empty \rangle\}. The \langle if non-empty \rangle part is executed if \_\_\_macro is non-empty. The * parameter can be replaced by the \_\_\_macro.
- \_\_\_bprintv{}\[\langle field1, field2, ..., \rangle\] \{\langle if defined \rangle\} \{\langle if not defined \rangle\}. The part \langle if defined \rangle is executed if \langle field1 \rangle or \langle field2 \rangle or ... is defined, else the second part \langle if not defined \rangle is executed. There is one field name or the list field names separated by commas. The parts cannot include any parameter.

There are two special 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 begin or end of each \langle entry-type \rangle. The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by \_\_\_printbib macro used in Op\TeX{} (and may be redefined by the author or document designer).

The \_\_\_bibmark=\langle something \rangle 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 printing of special fields \_\_\_author and \_\_\_editor. These fields include a list of names, each name is processed individually in a loop. The \_\_\_authorname or \_\_\_editorname is called for each name in 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 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 \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 <identifier> is an identifier of the phrase and the phrase itself is defined by \_sdef{_mt:bib.<identifier>}{<language>}{<phrase>}. See section 2.37.3 for more detail. Phrases for <identifiers>: and, etal, edition, citedate, volume, number, prepages, postpages, editor, editors, available, availablealso, bachelorthesis, masterthesis, phdthesis are defined already, see the end of section 2.37.3.

If you are using non-standard field-names in .bib database and bib style, you have to declare them by \_CreateField{<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 autocitations 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

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.

The \usebib command.

Corrections in librarian macros.

Main action per every entry.
The \_bprinta, \_bprintb, \_bprintc, \_bprintv commands used in the style files:
Various macros + multilinguas. Note that \_nobibwarnlist is used in \_bibwarning and it is set by \nobibwarning macro.

\def\bibwarning{%\ea\isinlist \ea\_nobibwarnlist\ea(\ea,\EntryKey,)\_iffalse\_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:
\begin{verbatim}
\fontfam[LMfonts]
nocite[*]
\usebib/s (iso690) op-biblist
\end
\end{verbatim}

Common rules in .bib files

There are entries of type @FOO {...} in the .bib file. Each entry consists of fields in the form name = "value", or name = {value}. No matter which form is used. If the value is pure numeric then you can say simply name = value. Warning: the comma after each field value is mandatory! If it is missing then the next field is ignored or bad interpreted.

The entry names and field names are case insensitive. If there exist a data field no mentioned here then it is simply ignored. You can use it to store more information (abstract, for example).

There are "standard fields" used in ancient bibTEX (author, title, editor, edition, etc., see http://en.wikipedia.org/wiki/BibTeX). The iso690 style introduces several "non-standard" fields: ednote, numbering, isbn, issn, doi, url, citedate, key, bibmark. They are documented here.

Moreover, there are two optional special fields:

- lang = language of the entry. The hyphenation plus autogenerated phrases and abbreviations will be typeset by this language.
- option = options by which you can control special printing of various fields.

There can be only one option field per each entry with (may be) more options separated by spaces. You can declare the global option(s) in your document applied for each entry by \biboptions={...}.

The author field

All names in the author list have to be separated by " and ". Each author can be written by various formats (the von part is typically missing):

\begin{verbatim}
Firstname(s) von Lastname
or
von Lastname, Firstname(s)
or
von Lastname, After, Firstname(s)
\end{verbatim}

Only the Lastname part is mandatory. Examples:

\begin{verbatim}
Petr Olšák
or
Olšák, Petr

Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
or
da Vinci, painter, Leonardo Piero
\end{verbatim}

The separator " and " between authors will be converted to comma during printing, but between semifinal and final author the word "and" (or something different depending on current language) is printed.

The first author is printed in reverse order: "LASTNAME, Firstname(s) von, After" and the others author are printed in normal order: "Firstname(s) von LASTNAME, After". This feature follows the
ISO 690 norm. The Lastname is capitalized using uppercase letters. But if the \caps font modifier is defined, then it is used and printed \caps\rm LastName.

You can specify the option aumax:\langle number\rangle. The \langle number\rangle denotes the maximum authors to be printed. The rest of authors are ignored and the et-al. is appended to the list of printed authors. This text is printed only if the aumax value is less than the real number of authors. If you have the same number of authors in the .bib file as you need to print but you want to append et-al. then you can use auetal option.

There is an aumin:\langle number\rangle option which denotes the definitive number of printed authors if the author list is not fully printed due to aumax. If aumin is unused then aumax authors is printed in such case.

All authors are printed if aumax:\langle number\rangle option isn’t given. There is no internal limit. But you can set the global options in your document by setting the \biboptions tokens list. For example:

\biboptions={aumax:7 aumin:1}
% if there is 8 or more authors then only first author is printed.
\entdd

Examples:
\begtt
author = "John Green and Bob Brown and Alice Black",
output: GREEN, John, Bob BROWN, and Alice BLACK.

author = "John Green and Bob Brown and Alice Black",
option = "aumax:1",
output: GREEN, John et al.

author = "John Green and Bob Brown and Alice Black",
option = "aumax:2",
output: GREEN, John, Bob BROWN et al.

author = "John Green and Bob Brown and Alice Black",
option = "aumax:3",
output: GREEN, John, Bob BROWN, and Alice BLACK.

author = "John Green and Bob Brown and Alice Black",
option = "auetal",
output: GREEN, John, Bob BROWN, Alice BLACK et al.

If you need to add a text before or after authors list, you can use the auprint:\{\langle value\rangle\} option. The \langle value\rangle will be printed instead of the authors list. The \langle value\rangle 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:\langle number\rangle 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 \langle number\rangle. There is an exception: autrim:0 means that no Firstnames are trimmed. This is default behavior. Another example: autrim:1 means that all Firstnames are trimmed.

author = "John Green and Bob Brown and Alice Black",
option = "auetal autrim:1",
output: GREEN, J., B. BROWN, A. BLACK et al.

If you need to write a team name or institution instead authors, replace all spaces by \_ in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after comma. Example:

author = "Czech\ Technical\ University\ in\ Prague, Faculty\ of\ Electrical\ Engeneering", 

150
The editor field
The editor field is used for list of the authors of the collection. The analogous rules as in author field are used here. It means that the authors are separated by “and”, the Firstnames, Lastnames etc. are interpreted and you can use the options edmax:⟨number⟩, edmin:⟨number⟩, edetal, edtrim:⟨number⟩ and edprint:{⟨value⟩} (with \ED macro). Example:

editor = "Jan Tomek and Petr Karas",
option = "edprint:{\ED, editors.} edtrim:1",

Output: J. TOMEK and P. KARAS, editors.

If edprint option is not set then {\ED, eds.} or {\ED, ed.} is used depending on the entry language and on the singular or plural of the editor(s).

The ednote field
The ednote field is used as the secondary authors and more editional info. The value is read as raw data without any interpretation of Lastname, Firstname etc.

ednote = "Illustrations by Robert \upper{Agarwal}, edited by Tom \upper{Nowak}"

output: Illustrations by Robert AGARWAL, edited by Tom NOWAK.
The \upper command have to be used for Lastnames in ednote field.

The title field
This is the title of the work. It will be printed (in common entry types) by italics. The ISO 690 norm declares, that the title plus optional subtitle are in italics and they are separated by colon. Next, the optional secondary title have to be printed in upright font. This can be added by titlepost:{⟨value⟩}.

Example:

title = "The Simple Title of The Work",
or
title = "Main Title: Subtitle",
or
title = "Main Title: Subtitle",
option = "titlepost:{Secondary title}"

The output of the last example: Main Title: Subtitle. Secondary title.

The edition field
This field is used only for second or more edition of cited work. Write only the number without the word "edition". The shortcut "ed." (or something else depending on current language) is added automatically.

Examples:

edition = "Second",
edition = "2nd",
edition = "2$^{\rm nd}$/\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}";


You can use \EDN macro in editionprint value. This macro is expanded to the edition value. Example:
edition = "Second",
option = "editionprint:{EDN}full revised edition",
or
edition = "Second full revised edition",
option = "editionprint:{EDN}",

The address, publisher, year fields
This is an anachronism from ancient BibTeX (unfortunately no exclusive) that the address field includes only the city of the publisher residence. No more data are here. The publisher field includes the name of the publisher.

address = "Berlin",
publisher = "Springer Verlag",
year = 2012,

Note, that the year needn’t to be inserted into quotes because it is pure numeric.
The letter a, b etc. are appended to the year automatically, if two or more subsequent entries in the bibliography list are not distinct by the first author and year fields. If you needn’t this feature, you can use the noautoletters option.
You can use "yearprint:{⟨value⟩}" option. If it is set then the ⟨value⟩ is used for printing year instead the real field value. The reason: year is sort sensitive, may be you need to print something else than only sorting key. Example:

year = 2000,
option = "yearprint:{© 2000}",

year = "2012a",
option = "yearprint:{2012}";

The address, publisher and year are typically mandatory fields. If they are missing then the warning occurs. But you can set unpublished option. Then this warning is suppressed. There is no difference in the printed output.

The url field
Use it without \url macro, but with http:// prefix. Example:

url = "http://petr.olsak.net/opmac.html",

The ISO 690 norm recommends to add the text “Available from” (or something else if different current language is used) before URL. It means, that the output of previous example is:

If the cs language is the current one than the output is:
Dostupné z: http://petr.olsak.net/opmac.html.
If the urlalso option is used, then the added text has the form “Available also from” or “Dostupné také z:” (if cs language is current).

The citedate field
This is the citation date. The field must be in the form year/month/day. It means, that the two slashes must be written here. The output depends on the current language. Example:

citedate = "2004/05/21",

Output when en is current: [cit. 2004-05-21].
Output when cs is current: [vid. 21. 5. 2004].

The howpublished field
This declares the available medium for cited document if it is not in printed form. Alternatives: online, CD, DVD, etc. Example:

howpublished = "online",

Output: [online].
The volume, number, pages and numbering fields
The volume is the “big mark” of the journal issue and the number is the “small mark” of the journal issue and pages includes the page range of the cited article in the journal. The volume is prefixed by Vol., the number by No. and the pages by pp. But these prefixes depends on the language of the entry.

Example:

volume = 31,
number = 3,
pages = "37--42",

volume = 31,
number = 3,
pages = "37--42",
lang = "cs",
Output: ročník 31, č. 3, s. 37–42.

If you disagree with the default prefixes, you can use the numbering field. When it is set then it is used instead of volume, number, pages fields and instead of any mentioned prefixes. The numbering can include macros \VOL, \NO, \PP, which are expanded to the respective values of fields. Example:

volume = 31,
number = 3,
pages = "37--42",
numbering = "Issue~\VOL/\NO, pages~\PP",
Output: Issue 31/3, pages 37–42

Note: The volume, numbers and pages fields are printed without numbering filed only in the @ARTICLE entry. It means, that if you need to visible them in the @INBOOK, @INPROCEEDINGS etc. entries, then you must to use numbering field.

Common notes about entries
The order of the fields in the entry is irrelevant. We use the printed order in this manual. The exclamation mark (!) denotes the mandatory field. If such field is missing then the warning occurs during processing.

If the unpublished option is set then the fields address, publisher, year, isbn and pages are not mandatory. If the nowarn option is set then no warnings about missing mandatory fields occurs.

If the field is used but not mentioned in the entry documentation below then it is silently ignored.

• The @BOOK entry
  This is used for book-like entries.
  Fields: author(!), title(!), howpublished, edition, ednote, address(!), publisher(!), year(!), citedate, series, isbn(!), doi, url, note.
  The ednote field here means the secondary authors (illustrator, cover design etc.).

• The @ARTICLE entry
  This is used for articles published in a journal.
  Fields: author(!), title(!), journal(!), howpublished, address, publisher, month, year, [numbering or volume, number, pages(!)], citedate, issn, doi, url, note.
  If the numbering is used then it is used instead volume, number, pages.

• The @INBOOK entry
  This is used for the part of a book.
  Fields: author(!), title(!), booktitle(!), howpublished, edition, ednote, address(!), publisher(!), year(!), numbering, citedate, series, isbn or issn, doi, url, note.
  The author field is used for author(s) of the part, the editor field includes author(s) or editor(s) of whole document. The pages field specifies the page range of the part. The series field can include more information about the part (chapter numbers etc.).
  The @INPROCEEDINGS and @CONFERENCE entries are equivalent to @INBOOK entry.

• The @THESIS entry
  This is used for student’s thesis.
  Fields: author(!), title(!), howpublished, address(!), school(!), month, year(!), citedate, type(!), ednote, doi, url, note.
The type field must include the text “Master’s Thesis” or something similar (depending on the language of the outer document).

There are nearly equivalent entries: @BACHELORSTHESIS, @MASTERSTHESIS and @PHDTHESIS. These entries set the type field to an appropriate value automatically. The type field is optional in such case. If it is used then it has a precedence before default setting.

- The @MISC entry
  It is intended for various usage.
  Fields: author, title, howpublished, ednote, citedate, doi, url, note.
  You can use \AU, \ED, \EDN, \VOL, \NO, \PP, \ADDR, \PUBL, \YEAR macros in ednote field. These macros print authors list, editors list, edition, volume, number, pages, address, publisher and year field values respectively.
  The reason of this entry is to give to you the possibility to set the format of entry by your own decision. The most of data are concentrated in ednote field.

- The @BOOKLET, @INCOLLECTION, @MANUAL, @PROCEEDINGS, @TECHREPORT, @UNPUBLISHED entries
  These entries are equivalent to @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 \numcitations is set
When \numcitations is set then \cite prints text orientes bib-marks instead numbers. This style file autogenerates these marks in the form “Lastname of the first author, comma, space, the year” if bibmark field isn’t declared. If you need to set an exception from this common format, then you can use bibmark field.

The OPmac trick http://petr.olsak.net/opmac-tricks-e.html#bibmark describes how to redefine the algorithm for bibmark auto-generating when you need the short form of the type [Au13].

Sorting
If \usebib/c is used then entries are sorted by citation order in the text. If \usebib/s is used then entries are sorted by “Lastname, Firstname(s)” of the first author and if more entries have this value equal, then the year is used (from older to newer). This feature follows the recommendation of the ISO 690 norm.

If you have the same authors and the same year, you can control the sorting by setting years as 2013, 2013a, 2013b, etc. You can print something different to the list using \yearprint{⟨value⟩} option, see the section about address, publisher and year above. The real value of year field (ie. not yearprint value) is also used in the text oriented bib-marks when \numcitations 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 = "Czmecjrkova Svetla",

This entry is now sorted between C and D.

The norm recommends to place the autocitations to the top of the list of references. You can do this by setting key,=\"@\", to each entry with your name because the @ character is sorted before A.

Languages
There is the language of the outer document and the languages of each entry. The ISO 690 norm recommends that the technical notes (the prefix before URL, the media type, the “and” conjunction between semifinal and final author) may be printed in the language of the outer document. The data of the entry have to be printed in the entry language (edition ed./vyd., Vol./ročník, No./č. etc.). Finally there are the phrases independent on the language (for example In:). Unfortunately, the bib\TeX\ supposes that the entry data are not fully included in the fields so the automaton have to add some text during processing (“ed.”, “Vol.”, “see also”, etc.). But what language have to be chosen?

The current value of the language register at the start of the .bib processing is descided 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:

\texttt{howpublished = "\_Mtext{bib.blue-ray}"}

Now, you can set the variants of \texttt{bib.blue-ray} phrase for various languages:

\texttt{\_sdef\_mt:bib.blue-ray:en} \{Blue-ray disc\}
\texttt{\_sdef\_mt:bib.blue-ray:cs} \{Blue-ray disk\}

Summary of non-standard fields
This style uses the following fields unknown by \texttt{bibTeX}:

option ... options separated by spaces
lang ... the language two-letter code of one entry
ednote ... editorial 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

\texttt{aumax:<number>} ... maximum number of printed authors
\texttt{aumin:<number>} ... number of printed authors if aumax exceeds
\texttt{autrim:<number>} ... full Firstnames iff number of authors are less than this
\texttt{auprint:{<value>}} ... text instead authors list (\texttt{\AU} macro may be used)
edmax, edmin, edtrim ... similar as above for editors list
\texttt{edprint:{<value>}} ... text instead editors list (\texttt{\ED} macro may be used)
titlepost:{<value>} ... text after title
yearprint:{<value>} ... text instead real year (\texttt{\YEAR} macro may be used)
editionprint:{<value>} ... text instead real edition (\texttt{\EDN} macro may be used)
urlalso ... the `available also from' is used instead `available from'
unpublished ... the publisher etc. fields are not mandatory
nowarn ... no mandatory fields

Another options in the option field are silently ignored.

2.32.6 Implementation of the \texttt{bib-iso690} style

Option field.
Formating of Author/Editor lists.

Preparation of bib-mark (used when \nonumcitations is set).
Setting phrases.

\_def\_bibconjunctionand{{\_Mtext{bib.and}}}
\_def\_preurl{{\_Mtext{bib.available}}}
\_let\_predoi=\_preurl
\_def\_postedition{{\text{bib.edition}}}
\_def\_Inclause{In:~}
\_def\_prevolume{{\text{bib.volume}}}
\_def\_prenumber{{\text{bib.number}}}
\_def\_prepages{{\text{bib.prepages}}}
\_def\_posteditor{{\ifnum0\_namecountraw>1 \text{bib.editors}\else \text{bib.editor}\fi}}
\_Mtext{{\_identifier}} expands to a phrase by outer document language (no entry language).

\_chardef\_documentlanguage=\_language
\_def\_Mtext#1{{\csname \mt:#1:\csname \lan:\the\_documentlanguage\_endcsname\endcsname}}
\_CreateField {lang}
\_def\_setlang#1{{\ifx#1\empty \else \_ifcsname _mt:bib.and:#1\_endcsname \_language=\csname \#1Patt\_endcsname \relax \else \_opwarning{No phrases for "#1" used by \[\EntryKey\] in .bib}\fi \fi}}

Non-standard fieldnames.

\_CreateField {ednote}
\_CreateField {citedate}
\_CreateField {numbering}
\_CreateField {isbn}
\_CreateField {issn}
\_CreateField {doi}
\_CreateField {url}
\_CreateField {bibmark}

Sorting.

\_SortingOrder{name,year}{lfvj}
\_SpecialSort {key}

Supporting macros.

\_def\_bibwarnings{\_bibwarning}
\_def\_bibwarningsb{\_bibwarning}
\_def\_docitedate #1/#2/#3/#4\relax{\_Mtext{bib.citedate}\_if^#2^#1\else\_if^#3^#1/#2\else\cs{\cs{\lan:\the\_documentlanguage}dateformat}#1/#2/#3\relax\fi\fi}
\_def\_doyear#1{\_biboptionvalue{yearprint}\_yearprint\_ifx\_yearprint\empty\_else\def\_YEAR{#1}\_yearprint\fi}
\_def\_preparenumbering{\_def\_VOL{\_RetrieveField{volume}}\_def\_NO{\_RetrieveField{number}}\_def\_PP{\_RetrieveField{pages}}}
\_def\_prepareednote{\_def\_EDN{\_RetrieveField{edition}}\_def\_ADDR{\_RetrieveField{address}}\_def\_PUBL{\_RetrieveField{publisher}}\_def\_YEAR{\_RetrieveField{year}}\_def\_AU{\_bprintb[!author]{\_doauthor0{####1}}{}}\_def\_AD{\_bprintb[!editor]{\_doeditor0{####1}}{}}\_preparenumbering}
\_def\_doedition#1{\_ifnum0\_namecountraw>1\_Mtext{bib.editors}\else\_Mtext{bib.editor}\fi}
Entry types.

\biboptionvalue{titlepost}\titlepost
\isbiboption{unpublished}\iftrue \let\bibwarninga=\relax \let\bibwarningb=\relax \fi
\isbiboption{nowarn}\iftrue \let\bibwarning=\relax \fi
\isbiboption{urlalso}\iftrue \def\preurl{\_Mtext{bib.availablealso}}\fi
\RetrieveFieldIn{lang}\langentry \setlang\langentry

\def\bookgeneric#1{\bprinta [howpublished] {[*].\ }{}% 
\bprintb [edition] {\_doedition{##1}\.}{}% 
\bprinta [ednote] {\.\}{}% 
\bprinta [address] {\_bprintv[publisher]{:}\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [publisher] {\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}\.}\.}{}% 
\bprinta [numbering] {\_preparenumbering* \_bprintv[citedate]{}{}% 
\bprinta [citedate] {\_docitedate///\relax.\ }{}% 
\_bibwarning}{}% 
\bprinta [series] {\.\ }{}% 
\bprinta [isbn] {\_Mtext{\bibwarningb}\.\}{}% 
\bprintb [doil] {\_predoi DOI \_ulink[http://dx.doi.org/#1]{##1}\.}{}% 
\bprintb [url] {\_preurl \_url{##1}. }{}% 

\def\_bookgeneric#1\{\_biboptionvalue{titlepost}\titlepost
\isbiboption{unpublished}\iftrue \let\bibwarninga=\relax \let\bibwarningb=\relax \fi
\isbiboption{nowarn}\iftrue \let\bibwarning=\relax \fi
\isbiboption{urlalso}\iftrue \def\preurl{\_Mtext{bib.availablealso}}\fi
\RetrieveFieldIn{lang}\langentry \setlang\langentry

\def\bookgeneric#1{\bprinta [howpublished] {[*].\ }{}% 
\bprintb [edition] {\_doedition{##1}\.}{}% 
\bprinta [ednote] {\.\}{}% 
\bprinta [address] {\_bprintv[publisher]{:}\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [publisher] {\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}\.}\.}{}% 
\bprinta [numbering] {\_preparenumbering* \_bprintv[citedate]{}{}% 
\bprinta [citedate] {\_docitedate///\relax.\ }{}% 
\_bibwarning}{}% 
\bprinta [series] {\.\ }{}% 
\bprinta [isbn] {\_Mtext{\bibwarningb}\.\}{}% 
\bprintb [doil] {\_predoi DOI \_ulink[http://dx.doi.org/#1]{##1}\.}{}% 
\bprintb [url] {\_preurl \_url{##1}. }{}% 

\def\bookgeneric#1\{\_biboptionvalue{titlepost}\titlepost
\isbiboption{unpublished}\iftrue \let\bibwarninga=\relax \let\bibwarningb=\relax \fi
\isbiboption{nowarn}\iftrue \let\bibwarning=\relax \fi
\isbiboption{urlalso}\iftrue \def\preurl{\_Mtext{bib.availablealso}}\fi
\RetrieveFieldIn{lang}\langentry \setlang\langentry

\def\bookgeneric#1{\bprinta [howpublished] {[*].\ }{}% 
\bprintb [edition] {\_doedition{##1}\.}{}% 
\bprinta [ednote] {\.\}{}% 
\bprinta [address] {\_bprintv[publisher]{:}\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [publisher] {\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}\.}\.}{}% 
\bprinta [numbering] {\_preparenumbering* \_bprintv[citedate]{}{}% 
\bprinta [citedate] {\_docitedate///\relax.\ }{}% 
\_bibwarning}{}% 
\bprinta [series] {\.\ }{}% 
\bprinta [isbn] {\_Mtext{\bibwarningb}\.\}{}% 
\bprintb [doil] {\_predoi DOI \_ulink[http://dx.doi.org/#1]{##1}\.}{}% 
\bprintb [url] {\_preurl \_url{##1}. }{}% 

\def\bookgeneric#1\{\_biboptionvalue{titlepost}\titlepost
\isbiboption{unpublished}\iftrue \let\bibwarninga=\relax \let\bibwarningb=\relax \fi
\isbiboption{nowarn}\iftrue \let\bibwarning=\relax \fi
\isbiboption{urlalso}\iftrue \def\preurl{\_Mtext{bib.availablealso}}\fi
\RetrieveFieldIn{lang}\langentry \setlang\langentry

\def\bookgeneric#1{\bprinta [howpublished] {[*].\ }{}% 
\bprintb [edition] {\_doedition{##1}\.}{}% 
\bprinta [ednote] {\.\}{}% 
\bprinta [address] {\_bprintv[publisher]{:}\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [publisher] {\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}\.}\.}{}% 
\bprinta [numbering] {\_preparenumbering* \_bprintv[citedate]{}{}% 
\bprinta [citedate] {\_docitedate///\relax.\ }{}% 
\_bibwarning}{}% 
\bprinta [series] {\.\ }{}% 
\bprinta [isbn] {\_Mtext{\bibwarningb}\.\}{}% 
\bprintb [doil] {\_predoi DOI \_ulink[http://dx.doi.org/#1]{##1}\.}{}% 
\bprintb [url] {\_preurl \_url{##1}. }{}% 

\def\bookgeneric#1\{\_biboptionvalue{titlepost}\titlepost
\isbiboption{unpublished}\iftrue \let\bibwarninga=\relax \let\bibwarningb=\relax \fi
\isbiboption{nowarn}\iftrue \let\bibwarning=\relax \fi
\isbiboption{urlalso}\iftrue \def\preurl{\_Mtext{bib.availablealso}}\fi
\RetrieveFieldIn{lang}\langentry \setlang\langentry

\def\bookgeneric#1{\bprinta [howpublished] {[*].\ }{}% 
\bprintb [edition] {\_doedition{##1}\.}{}% 
\bprinta [ednote] {\.\}{}% 
\bprinta [address] {\_bprintv[publisher]{:}\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [publisher] {\_bprintv[year]{,}\.}{}% \{\bibwarninga\} 
\bprintb [year] {\_doyear{##1}\_bprintv[citedate]{\_bprintv[numbering]{.}\.}\.}{}% 
\bprinta [numbering] {\_preparenumbering* \_bprintv[citedate]{}{}% 
\bprinta [citedate] {\_docitedate///\relax.\ }{}% 
\_bibwarning}{}% 
\bprinta [series] {\.\ }{}% 
\bprinta [isbn] {\_Mtext{\bibwarningb}\.\}{}% 
\bprintb [doil] {\_predoi DOI \_ulink[http://dx.doi.org/#1]{##1}\.}{}% 
\bprintb [url] {\_preurl \_url{##1}. }{}%
2.33 Sorting and making Index

\makeindex implements sorting algorithm at \TeX macro-language level. You need not any external program.

There are two passes in sorting algorithm. Primary pass does not distinguish between a group of letters (typically non-accented and accented). If the result of comparing two string is equal in primary pass then secondary pass is started. It distinguishes between variously accented letters. Czech rules, for example says: not accented before diacesis before acute before circumflex before ring. At less priority: lowercase letters must be before uppercase letters.
The \texttt{\_sortingdata(iso-code)} implements these rules for the language (iso-code). The groups between commas are not distinguished in the first pass. The second pass distinguishes all characters mentioned in the \texttt{\_sortingdata(iso-code)} (commas are ignored). The order of letters in the \texttt{\_sortingdata(iso-code)} macro is significant for sorting algorithm. The Czech rules (cs) are implemented here:

\begin{verbatim}
25 \_def \_sortingdatacs {
26   /, ( ),-, @, %
27   aáaąâ, %
28   b B, %
29   cC, %
30   čČ, %
31   dĎdď, %
32   eEéÉěĚ, %
33   fF, %
34   gG, %
35   hH, %
36   iíI, %
37   jJ, %
38   kK, %
39   lLĺĹľĽ, %
40   mM, %
41   nNňŇ, %
42   oOöÖóÓôÔ, %
43   pP, %
44   qQ, %
45   rŘrŘ, %
46   sŚsŚ, %
47   tTťŤ, %
48   uUüÜúÚůŮ, %
49   vV, %
50   wW, %
51   xX, %
52   yYýÝ, %
53   zZ, %
54   žŽ, %
55   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ','
56  }
\end{verbatim}

Characters ignored by sorting algorithm are declared in \texttt{\_ignoredchars(iso-code)}. The compound characters (two or more characters interpreted as one character in sorting algorithm) is mapped to single invisible characters in \texttt{\_compoundchars(iso-code)}. Czech rules declares ch or Ch or CH as a single letter sorted between H and I. See \texttt{\_sortingdatacs} above where these declared characters are used.

The characters declared in \texttt{\_ignoredchars} are ignored in first pass without additional condition. All characters are taken into account in second pass: ASCII characters with code '65 are sorted first if they are not mentioned in the \texttt{\_sortingdata(iso-code)} macro. Others not mentioned characters have undefined behavior during sorting.

\begin{verbatim}
76 \_def \_ignoredcharscs {.,;?!:''(\[]\)}
77 \_def \_compoundcharscs {ch:\"T Ch:\"U CH:\"V} % DZ etc. are sorted normally
\end{verbatim}

Slovak sorting rules are the same as Czech. The macro \texttt{\_sortingdata} includes Slovak letters too. Compound characters are the same. English sorting rules can be defined by \texttt{\_sortingdata} too because English alphabet is subset of Czech and Slovak alphabets. Only difference: \texttt{\_compoundcharsen} is empty in English rules.

You can declare these macros for more languages, if you wish to use \texttt{\makeindex} with sorting rules in respect to your language. Note: if you need to map compound characters to a character, don’t use \texttt{\`I} or \texttt{\`H} because these characters have very specific category code. And use space to separate more mappings, like in \texttt{\_compoundcharscs} above.

\begin{verbatim}
93 \_let \_sortingdataSk = \_sortingdata
94 \_let \_compoundcharsSk = \_compoundchars
95 \_let \_ignoredcharsSk = \_ignoredchars
\end{verbatim}
Preparing to primary pass is implemented by the \_setprimarysorting macro. It is called from \makeindex macro and all processing of sorting is in a group.

Strings to be sorted are prepared in \_\_preparesorting control sequences (in order to save \TeX memory). The \_\_preparesorting \_\langle string \_\rangle converts \_\langle string \_\rangle to \_\_tmpb with respect to the data initialized in \_\_setprimarysorting or \_\_setsecondarysorting. The compound characters are converted to single characters by the \_\_docompound macro.

Macro \_\_isAleB \_\langle string1 \_\rangle \_\_\langle string2 \_\rangle returns the result of comparison of given two strings to \_\_ifAleB control sequence. Usage: \_\_isAleB \_\langle string1 \_\rangle \_\_\langle string2 \_\rangle \_\_\_ifAleB ... \_\_else ... \_\_fi The converted strings (in respect of the data prepared for first pass) must be saved as values of \_\langle string1 \_\rangle and \_\langle string2 \_\rangle macros. The reason is speed: we don’t want to convert them repeatedly in each comparison. The macro \_\_testAleB \_\langle converted string1 \_\_\relax \_\rangle \_\langle converted string2 \_\_\relax \_\rangle \_\_\_testAleB does the real work. It reads first character from both converted strings, compares them and if it is equal then calls itself recursively else gives result.
Merge sort is very effectively implemented by \TeX macros. The following code is created by my son Miroslav. The \texttt{mergesort} macro expects that all items in \texttt{iilist} are separated by comma when it starts. It ends with sorted items in \texttt{iilist} without commas. So \texttt{dosorting} macro must prepare commas between items.

The \texttt{dosorting} \texttt{list} macro redefines \texttt{list} as sorted \texttt{list}. The \texttt{list} have to include control sequences in the form \texttt{⟨c⟩⟨string⟩}. These control sequences will be sorted in respect to \texttt{⟨strings⟩} without change of meanings of these control sequences. Their meanings are irrelevant when sorting. The first character \texttt{⟨c⟩} in \texttt{⟨c⟩⟨string⟩} should be whatever. It does not influence the sorting. Op\TeX uses comma at this place for sorting indexes: \,\texttt{⟨word1⟩},\,\texttt{⟨word2⟩},\,\texttt{⟨word3⟩},\,... .

The actual language (chosen for hyphenation patterns) is used for sorting data. If the \texttt{sortinglang} macro is defined as \texttt{⟨iso-code⟩} (for example \texttt{\def\sortinglang{de}}) then this has precedence and actual language is not used. Moreover, if you specify \texttt{asciisortingtrue} then ASCII sorting will be processed and all language sorting data will be ignored.
The `\makeindex` prints the index. First, it sorts the `\_iilist` second, it prints the sorted `\_iilist`, each item is printed using `\_printindexitem`

The `\makeindex` prints the index. First, it sorts the `\_iilist` second, it prints the sorted `\_iilist`, each item is printed using `\_printindexitem`.

The `\_printindexitem \langle word \rangle` prints one item to the index. If `\langle word \rangle` is defined then this is used instead real `\langle word \rangle` (this exception is declared by `\_iis` macro). Else `\langle word \rangle` is printed by `\_printii`. Finally, `\_printii` prints the value of `\langle word \rangle`, i.e. the list of pages.

`\_printii \langle word \rangle` does more intelligent work because we are working with words in the form `\langle main-word \rangle/\langle sub-word \rangle/\langle sub-sub-word \rangle`. The `\_everyii` tokens register is applied before `\_noindent`. User can declare something special here.

The `\_newiiletter{\langle letter \rangle}` 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}{\uppercasenew\setfontstrategy{at15pt}}}\medskip} makeindex.ompl
```

User can declare something special here.
\printiipages (paglist) & gets (paglist) in the form \langle \text{gp}\rangle \langle \text{pg}\rangle \langle \text{type}\rangle \langle \text{gp}\rangle \langle \text{pg}\rangle \langle \text{type}\rangle \ldots \langle \text{gp}\rangle \langle \text{pg}\rangle \langle \text{type}\rangle \text{ and it converts them to } \langle \text{pg}\rangle, \langle \text{gp}\rangle, \langle \text{from}\rangle \rightarrow \langle \text{to}\rangle, \langle \text{pg}\rangle \text{ 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 \langle \text{type}\rangle. Empty \langle \text{type}\rangle is most common, pages with b \langle \text{type}\rangle must be printed as bold and with i \langle \text{type}\rangle as italics. Moreover, the \langle \text{pg}\rangle mentioned here are \langle \text{gp}\rangle, but we have to print \langle \text{pg}\rangle. The following macros solves these tasks.

\makeindex

You can re-define \_pgprint \langle \text{gp}\rangle \langle \text{iitype}\rangle \langle \text{pg}\rangle \langle \text{type}\rangle \text{ if you need to implement more \langle iitype\rangle.}

\makeindex

The \_Index{\langle word\rangle} puts \langle word\rangle to the index. It writes \_Index{\langle word\rangle}{\langle iitype\rangle} to the .ref file. All others variants of indexing macros expands internally to \_Index.

\makeindex

The \_Index{\langle word\rangle}{\langle iitype\rangle} stores \langle word\rangle to the \_iilist if there is first occurrence of the \langle word\rangle. The list of pages where \langle word\rangle occurs, is the value of the macro \_iilist, so the \langle \text{gp}\rangle \langle \text{pg}\rangle \langle \text{iitype}\rangle is appended to this list. Moreover, we need a mapping from \langle \text{gp}\rangle to \langle \text{pg}\rangle, because we print \langle \text{pg}\rangle in the index, but hyperlinks are implemented by \langle \text{gp}\rangle. So, the macro \_pgi{\langle \text{gp}\rangle} is defined as \langle \text{pg}\rangle.

\makeindex
The implementation of macros \ii, \iid, \iis follows. Note that \ii works in horizontal mode on order to the \write whatsit is not broken from the following word. If you need to keep vertical mode, use \index{word} directly.

The \iiitype{(type)} saves the \langle type \rangle to the \_iitypesaved macro. It is used in the \iindex macro.

\_printfnotemark prints the footnote mark. You can re-define this macro if you want another design of footnotes. For example

\fnotenumpages
\def \_printfnotemark {\ifcase 0\fnotenum\or *\or **\or ***\or $\mathbox{†}$\or $\mathbox{‡}$\or $\mathbox{††}$\fi}
This code gives footnotes* and ** and *** and† etc. and it supposes that there are no more than 6 footnotes at one page.

If you want to distinguish between footnote marks in the text and in the front of footnote itself, then you can define \printfnotemarkA and \printfnotemarkB.

The \fnotelinks\langle colorA\rangle\langle colorB\rangle implements the hyperlinked footnotes (from text to footnote and backward).

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:\langle fnotenum⟩. Each \_Xpage macro sets the \_lfnotenum to zero.

The \fnote {⟨text⟩} macro is simple, \fnotemark and \fnotetext does the real work.

The \fnotetext calls \opfootnote which is equivalent to plain T\TeX \vfootnote. It creates new data to Insert \footins. The only difference is that we are able to propagate a macro parameter into Insert group before the text is printed (see section 2.18). This propagated macro is \fnset which sets smaller fonts.

Note that \vfootnote and \opfootnote doesn’t read the text as a parameter but during normal horizontal mode. This is reason why catcode changes (for example in-line verbatim) can be used here.

By default \mnote{⟨text⟩} are in right margin at odd pages and they are in left margin at even pages. The \mnote macro saves its position to .ref file as \_Xmnote without parameter. We define \_mn:⟨mnotenum⟩ as \_right or \_left when the .ref file is read. The \ifnum 0<0\#2 trick returns true if ⟨pageno⟩ has numeric type and false if it is non-numeric type (Roman numeral, for example). We prefer to use ⟨pageno⟩, but only if it has numeric type. We use ⟨gpageno⟩ in other cases.

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.
The \_mnoteD{⟨text⟩} macro sets the position the marginal note. The outer box of marginal note has zero width and zero depth and it is appended after current line using \vadjust primitive or it is inverted to vertical mode as a box with \vskip\baselineskip followed.

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.

We don’t want to process \fnote, \fnotemark, \mnote in TOC, headlines nor outlines.

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

We define auxiliary macro first (used by the \_address macro)

The \_boxlines{⟨line-1⟩}{⟨col⟩}{⟨line-2⟩}{⟨col⟩}...{⟨line-n⟩}{⟨col⟩} 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 ⟨col⟩ is set active and \everypar starts \hbox{ and acive ⟨col⟩ closes this \hbox by }.
The \report and \letter style initialization macros are defined here. The \letter defines \address and \subject macros.

The \slides macro reads macro file \texttt{slides.opm}, see the section \texttt{2.35.2}.

\section*{2.35.2 \slides style for presentations}

Default margins and design is declared here. The \texttt{ttfont} is scaled by \texttt{mag1.15} in order to balance the ex height of Helvetica (Heros) and LM fonts Typewriter. The \texttt{begtt...\endtt} verbatim is printed by smaller text.
The bottom margin is set to 3mm. If we use 1mm, then baseline of \footline is 2mm from the bottom page. This is depth of the \Grey rectangle used for page numbers. It is r-lapped to \hoffset width because left margin = \hoffset = right margin. It is 14mm for narrow pages or 16mm for wide pages.

The \subtit is defined analogically like \tit.

The \pshow⟨num⟩ prints the text in invisible (transparent) font when \layernum⟨num⟩. The transparency is set by by \pdfrageresourc\primitives.

The main level list of items is activated here. The \item:X and \item:x are used and are re-defined here. If we are in nested level of items and \pg+ is used then \egroups macro expands to the right number of \egroups in order to close page correctly. The level of nested item lists is saved to the \ilevel register and used when we start again the next text after \pg+.

The default values of \pg, i.e. \pg; \pg+ and \pg. are very simple. They are used when \showslides is not specified.

The \endslides is defined as \end primitive, but slide-designer can redefine it. For example, \OpTeX\ trick 0029 shows how to define a clickable navigation to the pages and how to check the data integrity at the end of document using \endslides.

The \bye macro is re-defined here as an alternative to \pg..

We need no numbers and no table of contents when using slides. The \printsec macro is redefined in order the title is centered and typeset in \Blue.
When \texttt{\slideshow} is active then each page is opened by \texttt{\setbox\slidepage\vbox\bgroup} (roughly speaking) and closed by \texttt{\egroup}. The material is \texttt{\unvbox}ed and saved for the usage in the next usage if \texttt{\pg+} is in process. The \texttt{\_slidelay} is incremented instead \texttt{\pageno} if \texttt{\pg+}. This counter is equal to \texttt{\count1}, so it is printed to the terminal and log file next to \texttt{\pageno}.

The code is somewhat more complicated when \texttt{\layers} is used. Then \texttt{\langle\texttt{layered-text}\rangle} is saved to the \texttt{\_layertext} macro, the material before it is in \texttt{\_slidepage} box and the material after it is in \texttt{\_slidepageB} box. The pages are completed in the \texttt{\loop} which increments the \texttt{\_layernum} register.
Default \texttt{\layers{num}} macro (when \texttt{\slideshow} is not activated) is simple. It prints the \texttt{\layered-text} with \texttt{\layernum=(num)+1} because we need the result after last layer is processed.

\texttt{\def\layers{\par\layernum=\numexpr\layernum+1\relax}\
\let\endlayers=\relax
\def\layers{\layers}\
\public\layernum\slideshow\pg
\def\fnotenumpages{\def\fnotenum{\the\lfnotenum}\pgfnotefalse\
\def\lfnotenumreset{\global\lfnotenum=0}\
\let\lfnotenumreset=\relax\
\public\fnotenumpages;}

2.36 Logos

\texttt{\texttt{\protected\def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\ignoreslash}}\
\protected\def\OpTeX{Op\kern-.1em\TeX}\
\protected\def\LuaTeX{Lua\TeX}\
\protected\def\XeTeX{X\kern-.125em\phantom E%\pdfsave\rlap{\pdfscale{-1}{1}\lower.5ex\hbox{E}}\pdfrestore \kern-.1667em \TeX}\
\def\ignoreslash{\futurelet\next\ignoreslashA}\
\def\ignoreslashA{\if\next/\ea\ignoreit\fi}\
\public\TeX\OpTeX\LuaTeX\XeTeX\ignoreslash;}

\texttt{\protected\def\LaTeX{T\kern-.36em\kern\slantcorr % slant correction\
\raise\tmbox{\hbox{\thefontscale[710]\kern-.15em\kern-\slantcorr \TeX}}}\
\protected\def\slantcorr{\ea\ignorept\the\fontdimen1\font\tmpdim}\
\public\LaTeX;}

\OPmac, \CS and \csplain logos.
The expandable versions of logos used in Outlines needs the expandable \ignslash (instead of the \ignoreslash).

2.37 Multilingual support

2.37.1 Lowercase, uppercase codes

All codes in unicode table keep information about pairs lowercase-uppercase letters or single letter. We need to read such information and set appropriate \lccode and \uccode. The \catcode above the code 127 is not set, i.e. the \catcode=12 for all codes above 127.

The file uni-lcuc.opm does this work. It is not much interesting file, only first few lines from 15928 lines in total is shown here.

2.37.2 Hyphenations

The \lan means a shortcut of language name (mostly by ISO 639-1). The following control sequences are used for language switching:

- \lan{number} expands to \iso of the language. The number is internal number of languages used as a value of \language register.
- \ulan{long-lang} expands to \iso too. This is transformation from long name of language (lowercase letters) to \iso.
- \iso\Patt (for example \csPatt) is the language \number declared by \chardef.
- \iso\lang (for example \enlang, \cslang, \sklang, \delang, \pllang) is language selector. It exists in two states.
• Initialization state: when \(\text{iso-code}\lang\) is used first then it must load the patterns into memory using Lua code. If it is done then the \(\text{iso-code}\lang\) re-defines itself to processing state.

• Processing state: it only sets \(\text{language}=\text{iso-code}\)Patt, i.e. it selects the hyphenation patterns. It does a little more language-dependent work, as mentioned below.

• \(\text{langspecific}:=\langle\text{isocode}\rangle\) is processed by \(\text{iso-code}\lang\) and it should include language-specific macros declared by user or macro designer.

The USenglish patterns are preloaded first:

\begin{verbatim}
\chardef\_enPatt=0
\def\_lan:{\langle\text{isocode}\rangle}\{\langle\text{iso-code}\rangle}\{\langle\text{long-lang}\rangle}\% loads patterns using Lua code
\gdef\_enlang{\langle\text{iso-code}\rangle}\{\langle\text{long-lang}\rangle}\% re-defines itself to processing state
\end{verbatim}

\begin{verbatim}
\_preplang \langle\text{iso-code}\rangle \{\langle\text{long-lang}\rangle\} \{\langle\text{hyph-file-spec}\rangle\} \{\langle\text{number}\rangle\} \{\langle\text{pre-hyph}\rangle\} \{\langle\text{post-hyph}\rangle\}

\chardef\_enPatt=0
\def\_lan:{\langle\text{isocode}\rangle}\{\langle\text{iso-code}\rangle}\{\langle\text{long-lang}\rangle}\% \text{language}\=\langle\text{iso-code}\rangle\{\langle\text{number}\rangle\}
\def\_enlang{\langle\text{iso-code}\rangle}\{\langle\text{long-lang}\rangle}\% \text{re-defines itself to processing state}
\end{verbatim}

You can see that \(\text{iso-code}\lang\) runs \_loadpattrs and \_uselang first (in initialization state) and it runs only \_uselang when it is called again (in processing state).

The \langle\text{hyph-file-spec}\rangle is a part of full file name which is read: hyphen-\langle\text{hyph-file-spec}\rangle.tex. The patterns and hyphenation exceptions are saved here in UTF-8 encoding. The \langle\text{hyph-file-spec}\rangle should be a list of individual \langle\text{hyph-file-spec}\rangle’s separated by comma, see the language Serbian below for example.
\_uselang{(iso-code)}\_uselang{Patt (pre-hyph)(post-hyph)} sets \texttt{language}, \texttt{lefthyphenmin}, \texttt{righthyphenmin} and runs \texttt{frenchspacing}. This default language-dependent settings should be re-declared by \texttt{\_langspecific:(iso-code)} which is run finally (it is \texttt{\_relax} by default, only \texttt{\_langspecific:en} runs \texttt{\_nonfrenchspacing}).

\def\_uselang#1#2#3#4{\_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\_relax}
\_frenchspacing % \_nonfrenchspacing can be set in \texttt{\_cs{\_langspecific:lan}}
\_cs{\_langspecific:#1}

The \texttt{\_uselanguage \texttt{(long-lang)}} is defined here (for compatibility with e-plain users).

\def\_uselanguage#1{\_lowercase{\_cs{\_ulan:#1}lang}}
\public \_uselanguage ;

The numbers for languages are declared as fixed constants (no auto-generated). This concept is inspired from CSplain. There are typical numbers of languages in CSplain: 5=Czech in IL2, 15=Czech in T1 and 115=Czech in Unicode. We keep these constants but we load only Unicode patterns (greater than 100), of course.
The \langlist includes names of all languages which are ready to load and use their hyphenation patterns. This list is printed to terminal and to log at init\TeX state here. It can be used when processing document too.

```
\message{Language hyph.patterns ready to load: \langlist.}
```

Maybe, you need to do more language specific actions than just switching hyphenation patterns. For example you need to load a specific font with a specific script used in selected language, you can define a macros for quotation marks depending on the language etc.

The example shows how to declare such language specific things.

```
\def\langset #1 #2{\sdef{_langspecific:#1}{#2}}
\langset fr {... declare French quotation marks} 
\langset de {... declare German quotation marks} 
\langset gr {... switch to Greek fonts family} 
... etc.
```

Note that you need not to set language specific phrases (like \today) by this code. Another concept is used for such tasks. See the section 2.37.3 for more details.

### 2.37.3 Multilingual phrases and quotation marks

```
\public \langlist ;
```

Only four words are generated by Op\TeX macros: “Chapter”, “Table”, “Figure” and “Subject”. These phrases can be generated depending on the current value of \language register, if you use
\_mtext\{(phrase-id)}}, specially \_mtext\{chap\}, \_mtext\{t\}, \_mtext\{f\} or \_mtext\{subj\}. If your macros generate more words then you can define such words by \sdef\{_mt:\{phrase-id\}:\{lang\}} where \{phrase-id\} is a label for declared word and \{lang\} is language shortcut (iso code).

Using \_langw \{lang\} \{chapter\} \{table\} \{figure\} \{subject\} you can declare these words more effectively:

You can add more words as you wish. For example \today macro:

Quotes should be tagged by "\langle text \rangle" and \’\langle text \rangle\’ if \langle iso-code \rangle quotes is declared at beginning of the document (for example \enquotes). If not, then the control sequences \" and \’ are undefined. Remember, that they are used in another meaning when \oldaccents command is used. The macros \" and \’ are not defined as \protected because we need their expansion when \outlines are created. User can declare quotes by \quoteschars\langle clqq \rangle\langle crqq \rangle\langle clq \rangle\langle crq \rangle, where \langle clqq \rangle...\langle crqq \rangle are normal quotes and \langle clq \rangle...\langle crq \rangle are alternative quotes. or use \altquotes to swap between meaning of these two types of quotes.

\enquotes, \csquotes, \dequotes, \frquotes etc. are defined here.
The \quoteschars{⟨llqq⟩⟨rqq⟩⟨lq⟩⟨rq⟩} defines " and ' as \qqA\_qqB in normal mode and as expandable macros in outline mode. We want to well process the common cases: "&" or "‘‘. This is 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 reason why the sub-verbatim mode is used when first character is \{ in the parameter.

Sometimes should be usable to leave the markup "such" or 'such' i.e. without the first backslash. Then you can make the characters " and ' active by the \activequotes macro and leave quotes without first backslash. First, declare \iso-code quotes, then \altquotes (if needed) and finally \activequotes.

Bibliography references generated by \usebib uses more language-dependent phrases. They are declared here. We don’t want to save all these phrases into format, so the trick with \_endinput is used here. When \usebib is processed then following part of the file languages.opm is read again.

Only phrases of few languages are declared here now. If you want to declare phrases of your language, please create an “issue” or a “request” at https://github.com/olsak/OpTeX or send me email with new phrases for your language (or language you know:). I am ready to put them here. Temporarily, you can put your definitions into \bibtexhook token list.
2.38 Other macros

Miscellaneous macros are here.

\useOpTeX and \useoptex are declared as \relax.

The \lastpage and \totalpages get the information from the \currpage. The \Xpage from .ref file sets the \currpage.

We need \uv, \clqq, \crqq, \flqq, \frqq, \uslang, \ehyph, \chyp, \shyph, for backward compatibility with ČSplain. Codes are set according to Unicode, because we are using Czech only in Unicode when LuaTEX is used.

The \letfont was used in ČSplain instead of \fontlet.

Non breaking space in Unicode.

TikZ needs these funny control sequences.

We don’t want to read opmac.tex unless \input opmac is specified.

We allow empty lines in math formulae. It is more comfortable.
Lorem ipsum can be printed by \texttt{\lipsum[(range)]} or \texttt{\lorem[(range)]}, for example \texttt{\lipsum[3]} or \texttt{\lipsum[112-121]}, max=150.

First usage of \texttt{\lipsum} reads the \LaTeX file \texttt{lipsum.ltd.tex} and prints the selected paragraph(s). Next usages of \texttt{\lipsum} prints the selected paragraph(s) from memory. This second and more usages of \texttt{\lipsum} are fully expandable. If you want to have all printings of \texttt{\lipsum} expandable, use dummy \texttt{\lipsum[0]} first.

2.39 Lua code embedded to the format

The file \texttt{optex.lua} is loaded into the format in \texttt{optex.ini} as byte-code and initialized by \texttt{\everyjob}, see section 2.1.

The file implements part of the functionality from \texttt{luatexbase} namespace, nowadays defined by \LaTeX kernel. \texttt{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 luatexload’s use.

GENERAL

Error function used by following functions for critical errors.

For a \texttt{\chardef}, \texttt{\countdef}, etc., csname return corresponding register number. The responsibility of providing a \texttt{\XXdef}’d name is on the caller.

ALLOCATORS

An attribute allocator in Lua that cooperates with normal OpTEX allocator.
CALLBACKS

callback = callback or {}  

Save callback.register function for internal use.

local callback_register = callback.register
function callback.register(name, fn)
  err("direct registering of callbacks is forbidden, use 'callback.add_to_callback'")
end

Table with lists of functions for different callbacks.

local callback_functions = {}

Table that maps callback name to a list of descriptions of its added functions. The order corresponds with callback_functions.

local callback_description = {}

Table used to differentiate user callbacks from standard callbacks. Contains user callbacks as keys.

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).

local default_functions = {}

Table that maps standard (and later user) callback names to their types.

local callback_types = {
  -- file discovery
  find_read_file = "exclusive",
  find_write_file = "exclusive",
  find_font_file = "data",
  find_output_file = "data",
  find_format_file = "data",
  find_vf_file = "data",
  find_map_file = "data",
  find_enc_file = "data",
  find_pk_file = "data",
  find_data_file = "data",
  find_opentype_file = "data",
  find_truetype_file = "data",
  find_type1_file = "data",
  find_image_file = "data",
  open_read_file = "exclusive",
  read_font_file = "exclusive",
  read_vf_file = "exclusive",
  read_map_file = "exclusive",
  read_enc_file = "exclusive",
  read_pk_file = "exclusive",
  read_data_file = "exclusive",
  read_truetype_file = "exclusive",
  read_type1_file = "exclusive",
  read_opentype_file = "exclusive",
  -- data processing
  process_input_buffer = "data",
  process_output_buffer = "data",
  process_jobname = "data",
  -- node list processing
  contribute_filter = "simple",
  buildpage_filter = "simple",}
Return a list containing descriptions of added callback functions for specific callback.

Create a user callback that can only be called manually using call_callback. A default function is only needed by "exclusive" callbacks.
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).

```lua
function callback.add_to_callback(name, fn, description)
    if user_callbacks[name] or callback_functions[name] or default_functions[name] then
        -- either:
        -- a) user callback - no need to register anything
        -- b) standard callback that has already been registered
        -- c) standard callback with default function registered separately
        -- (mlist_to_hlist)
    elseif callback_types[name] then
        -- This is a standard luatex callback with first function being added,
        -- register a proxy function as a real callback. Assert, so we know
        -- when things break, like when callbacks get redefined by future
        -- luatex.
        assert(callback_register(name, function(...)
            return callback.call_callback(name, ...)
        end))
    else
        err("cannot add to callback ".name.." - no such callback exists")
    end
    -- add function to callback list for this callback
    callback_functions[name] = callback_functions[name] or {}
    table.insert(callback_functions[name], fn)
    -- add description to description list
    callback_description[name] = callback_description[name] or {}
    table.insert(callback_description[name], description)
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.

```lua
function callback.remove_from_callback(name, description)
    local descriptions = callback_description[name]
    local index
    for i, desc in ipairs(descriptions) do
        if desc == description then
            index = i
            break
        end
    end
    table.remove(descriptions, index)
    local fn = table.remove(callback_functions[name], index)
    -- add description to description list
    callback_description[name] = callback_description[name] or {}
    table.insert(callback_description[name], description)
end
```

```lua
if #descriptions == 0 then
    -- Delete the list entirely to allow easy checking of "truthiness".
    callback_functions[name] = nil
end
if not user_callbacks[name] and not default_functions[name] then
    -- this is a standard callback with no added functions and no
    -- default function (i.e. not mlist_to_hlist), restore standard
    -- behaviour by unregistering.
    callback_register(name, nil)
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.
Create “virtual” callbacks pre/post\_mlist\_to\_hlist\_filter by setting mlist\_to\_hlist callback. The default behaviour of mlist\_to\_hlist is kept by using a default function, but it can still be overridden by using add\_to\_callback.

```lua
default_functions["mlist\_to\_hlist"] = node.mlist\_to\_hlist
callback.create\_callback("pre\_mlist\_to\_hlist\_filter", "list")
callback.create\_callback("post\_mlist\_to\_hlist\_filter", "reverselist")
callback\_register("mlist\_to\_hlist", function(head, ...)
  -- pre\_mlist\_to\_hlist\_filter
  local new\_head = callback.call\_callback("pre\_mlist\_to\_hlist\_filter", head, ...)
  if new\_head == false then
    node.flush\_list(head)
    return nil
  elseif new\_head ~= true then
    head = new\_head
  end
  -- mlist\_to\_hlist means either added functions or standard luatex behavior
  -- of node.mlist\_to\_hlist (handled by default function)
  head = callback.call\_callback("mlist\_to\_hlist", head, ...)
  -- post\_mlist\_to\_hlist\_filter
  new\_head = callback.call\_callback("post\_mlist\_to\_hlist\_filter", head, ...)
  if new\_head == false then
    node.flush\_list(head)
    return nil
  elseif new\_head ~= true then
    head = new\_head
  end
  return head
end
```

Compatibility with \texttt{\LaTeX} through luatexbase namespace. Needed for luaotfload.

```lua
luatexbase = {
  registernumber = registernumber,
  attributes = attributes,
  new\_attribute = alloc\_new\_attribute,
  callback\_descriptions = callback\_callback\_descriptions,
  create\_callback = callback\_create\_callback,
  add\_to\_callback = callback\_add\_to\_callback,
  remove\_from\_callback = callback\_remove\_from\_callback,
  call\_callback = callback\_call\_callback,
  callback\_types = { }
}
```

## 2.40 Printing documentation

The \texttt{\printdoc} \texttt{\{filename\}\{space\}} and \texttt{\printdoctail} \texttt{\{filename\}\{space\}} commands are defined after the file \texttt{doc.opm} is load by \texttt{\load [doc]}.

The \texttt{\printdoc} starts reading of given \texttt{\{filename\}} from the second line. The file is read in the listing mode. The \texttt{\prindoctail} starts reading given \texttt{\{filename\}} from the first occurrence of the \texttt{\_encode}. The file is read in normal mode (like \texttt{\input \{filename\}}).

The listing mode prints the lines as listing of a code. This mode is finished when first \texttt{\_doc} occurs or first \texttt{\_endcode} occurs. At least two spaces must precede before such \texttt{\_doc}. On the other hand, the \texttt{\_encode} must be at the left edge of the line without spaces. If this rule is not met then the listing mode continues.

If the first line or the last line of the listing mode is empty then such lines are not printed. The maximal number of printed lines in the listing mode is \texttt{\maxlines}. Is set to almost infinity (100000). You can set it to a more sensible value. Such setting is valid only for the first following listing mode.

When the listing mode is finished by \texttt{\_doc} then next lines are read in the normal way, but the material between \texttt{\begtt} ... \texttt{\endtt} pair is shifted by three letters left. The reason is that the three
spaces of indentation is recommended in the \_doc ... \_cod pair and this shifting is a compensation of this indentation.

The \_cod macro ignores the rest of current line and starts the listing mode again.

When the listing mode is finished by the \_endcode then the \endinput is applied, the reading of the file opened by \printdoc is finished.

You cannot reach the end of the file (without \_endcode) in the listing mode.

The listing mode creates all control sequences which are listed in the index as active link to the main documentation point of such control sequence and prints them in blue. Other text is printed in black.

The main documentation point is denoted by "\sequence" in red, for example "\foo". The user documentation point is the first occurrence of "\sequence", for example "\foo". There can be more such markups, all of them are hyperlinks to the main documentation point. And main documentation point is hyperlink to the user documentation point, if such point exists. Finally, the "\sequence" (for example "\foo") are hyperlinks to the user documentation point.

General declarations.

Maybe, somebody needs \seccc or \secccc?

\enddocument can be redefined.

Full page of listing causes underfill \vbox in output routine. We need to add a small tolerance.

The listing mode is implemented here. The \maxlines is maximal lines of code printed in the listing mode.

The scanner of the control sequences in the listing mode.
\def\makecs{\ifcat a \noexpand\next \else \ea\makecsF \fi}
\def\makecsF{\ifx \tmp \empty \let \next=\tmp \fi}
\ifcsname ,\tmp \endcsname \link[cs:\tmp]{\Blue}{\csstring\tmp} \else \let \next=\tmp \fi}
\ifx \next \empty \let \next=\tmp \fi}
\ifcsname ,\next \endcsname \link[cs:\next]{\Blue}{\csstring\tmp}\else \csstring\tmp \fi}fi fi}
\def\processinput{\let \start=\relax
\ea\replstring\ea\tmpb\ea{\aspace^^J}{^^J}
\addtolist\tmpb{\end} \isinlist\tmpb{\start^^J}\iftrue \advance\ttline by1 \fi
\replstring\tmpb{\start^^J}{\start} \replstring\tmpb{\start}{} \replstring\tmpb{^^J\end}{\end} \replstring\tmpb{\end}{\end}
\ea\prepareverbdata\ea\tmpb\ea{\tmpb^^J} \replthis{\csstring\}{\noexpand\makecs} \ea\printverb\tmpb\end
\par
\endgroup \ttskip \isnextchar\par{}{\noindent}fi}
\def\remfirstunderscore#1{\ea\remfirstunderscoreA#1\relax#1}
\def\remfirstunderscoreA#1#2\relax#3{\if _#1\def#3{#2}\fi}
\def\printcodeline#1{\advance\maxlines by-1 \ifnum\maxlines<0 \endverbprinting \fi
\penalty\ttpenalty \kern-4pt
\noindent\rlap{\Yellow\vrule height8pt depth5pt width\hsize} \printfilename \indent printverblinenum #1\par}
\def\printfilename{\hbox to0pt{\hskip\hsize\vbox to0pt{\vss\llap{\Brown\docfile}\kern7.5pt}}\hss}
\let\printfilename=\relax
\everytt={\let\printverblinenum=\relax}
\def\printdoctail #1 {\bgroup \everytt={}
\ttline=-1 \ea\printdoctailA\input #1 \egroup}
{\long\gdef\printdoctailA#1\endcode{}}
\public\printdoc\printdoctail;
\def\docfile{\def\printdoc#1{\par \def\docfile{#1}
\everytt={\ttshift=-15pt \let\printverblinenum=\relax}
\everytt={\let\printverblinenum=\relax}
\def\docfile{}\endgroup}
\def\printdoctail#1 {\bgroup
\everytt={}
\ttline=-1 \ea\printdoctailA\input #1 \egroup}
\long\gdef\printdoctailA#1\endcode{}\def\docfile#1{\par \def\docfile{#1}
\everytt={\ttshift=-15pt \let\printverblinenum=\relax}
\everytt={\let\printverblinenum=\relax}
\def\docfile{}\endgroup}
\def\docfile#1{\par \def\docfile{#1}
\everytt={\ttshift=-15pt \let\printverblinenum=\relax}
\everytt={\let\printverblinenum=\relax}
\def\docfile{}\endgroup}

\docfile is currently documented file.
\printdoc and \printdoctail macros are defined here.
\verbblockuput \vitt{\langle filename \rangle \langle from \rangle-\langle to \rangle \langle filename \rangle} if you need analogical design like in listing mode.
The Index entries are without the trailing backslash. We must to add it when printing Index.

The <something> will be print as ⟨something⟩.

If this macro is loaded by \load then we need to initialize catcodes using the \_afterload macro.

Main documentation point and hyperlinks to/from it. Main documentation point: \`\foo`. User-level documentation point: ^\`\foo`, first occurrence only. Next occurrences are only links to main documentation point. Link to user-level documentation point: ~\`\foo`. If user-level documentation point follows the main documentation point then use \_forwardlink\`\foo`.

The Index entries are without the trailing backslash. We must to add it when printing Index.
Index

| \_aboveliskip | 119 |
| \_activequotes | 177 |
| \_activetttchar | 16–17, 121 |
| \_addcolor | 104 |
| \_additcorr | 96 |
| \_address | 25, 167–168 |
| \_addtabitemx | 137 |
| \_addto | 36, 51, 97 |
| \_addtomodlist | 72 |
| \_adef | 17, 35 |
| \_adots | 81 |
| \_addpageno | 97, 99 |
| \_afterfi | 38 |
| \_afteritcorr | 96 |
| \_afterload | 48 |
| \_allocator | 37 |
| \allowbreak | 53 |
| \_altquotes | 176–177 |
| \_asciisortingtrue | 162 |
| \_athe | 120 |
| \_authorname | 145 |
| \_b | 54 |
| \_backgroundbox | 98 |
| \_backgroundpic | 130 |
| \_bbchar | 75, 89 |
| \_begitems | 13–14, 119–120 |
| \_begtt | 16–18, 98, 121 |
| \_belowliskip | 119 |
| \_belovetitle | 115, 117 |
| \_bf | 8–9, 60, 62, 75, 89 |
| \_bggroup | 35 |
| \_bi | 8–9, 60, 62, 75, 89 |
| \_bib | 20, 143 |
| \_bibmark | 141, 145 |
| \_bibnum | 109, 141 |
| \_bipeskip | 144 |
| \_bibtexhook | 46, 144 |
| \_bibwarning | 145, 149 |
| \_big | 80 |
| \_Big | 80 |
| \_bigbreak | 53 |
| \_bigg | 80 |
| \_Bigg | 80 |
| \_biggl | 80 |
| \_Biggl | 80 |
| \_bigm | 80 |
| \_Bigm | 80 |
| \_bigr | 80 |
| \_Bigr | 80 |
| \_bigskip | 52 |
| \_Black | 102 |
| \_Blue | 21, 102 |
| \_box | 83 |
| \_boldify | 64, 97 |
| \_boldmath | 9, 75–77, 87 |
| \_boldunimath | 87 |
| \_boxlines | 167 |
| \_bp | 50 |
| \_bk | 50 |
| \_bprinta | 145, 148 |
| \_bprintb | 145, 148 |
| \_bprintc | 145, 148 |
| \_bprintv | 145, 148 |
| \_bracedparam | 49 |
| \_break | 53 |
| \_Brown | 102 |
| \_bslash | 35 |
| \_buildrel | 83 |
| \_byehook | 36 |
| \_c | 54 |
| \_calsmallmath | 75, 89 |
| \_caption | 10–12, 118 |
| \_cases | 83 |
| \_catalogexclude | 74 |
| \_catalogmathsample | 74 |
| \_catalogsample | 74 |
| \_catalogonly | 74 |
| \_catalogsample | 74 |
| \_catcode | 50 |
| \_cdots | 81 |
| \_centerline | 53 |
| \_chap | 10, 12, 17–18, 49, 114, 116 |
| \_chapfont | 64, 114 |
| \_chapsmallmath | 155 |
| \_chapsmallmath | 155 |
| \_chapsmallmath | 155 |
| \_chapsmallmath | 155 |
| \_chapseparator | 47 |
| \_circle | 131 |
| \_cliptools | 47 |
| \_cite | 12, 20, 141, 144 |
| \_citet | 142 |
| \_citeborder | 12, 109 |
| \_cliptools | 23, 133 |
| \_cliptools | 23, 133 |
| \_cliptools | 23, 133 |
| \_cliptools | 23, 133 |
| \_cliptools | 31, 178 |
| \_cliptools | 104 |
| \_cliptools | 103 |
| \_cliptools | 103 |
| \_cliptools | 31, 32 |
| \_cliptools | 16–17, 121 |
| \_cliptools | 31–32 |
| \_cliptools | 137 |
| \_cliptools | 103 |
| \_colordef | 21, 101–103, 105 |
| \_colordefFin | 103 |
| \_colorstackpop | 103 |
| \_colorstackpush | 103 |
| \_colorstackset | 103 |
| \_commentchars | 18, 122, 124 |
| \_commoncolordef | 104 |
| \_completepage | 97–98 |
| \_compoundchars | 160 |
| \_compoundchars | 160 |
| \_compoundchars | 160 |
| \_compoundchars | 160 |
| \_compoundchars | 83 |
| \_corrmsizes | 76 |
| \_crl | 15, 134, 138 |
| \_crl | 15, 134, 138 |
| \_crl | 15, 134, 138 |
| \_crl | 15, 134, 138 |
| \_crl | 178 |
| \_cs | 36 |
| \_CS | 171 |
| \_cskip | 10, 118 |
| \_cs | 23–24, 172 |
| \_cs | 23–24, 172 |
| \_cs | 23–24, 172 |
| \_cs | 23–24, 172 |
| \_cs | 35 |
| \_cslang | 35 |
| \_csplain | 171 |
| \_csquotes | 24, 176 |
| \_ctablelist | 48 |
| \_currentheadings | 70 |
| \_currpage | 107, 110, 178 |
| \_currstyle | 85 |
| \_currV | 66, 71 |
| \_currv | 8–9, 59–62, 64, 72 |
| \_cyan | 21, 102 |
| \_d | 54 |
| \_ddlinedata | 137 |
| \_ddots | 81 |
| \_decdigit | 50 |
| \_defaultfontfeatures | 74 |
| \_defaultitem | 14, 45, 120 |
| \_delang | 23, 172 |
| \_dequotes | 24, 176 |
| \_dest | 13, 109 |
| \_destactive | 109 |
| \_destheight | 109 |
| \_displaylines | 84 |
| \_do | 39 |
| \_do | 39 |
| \_dobystyle | 85 |
| \_do | 31–32, 50 |
| \_docompound | 161 |
| \_doldo | 86 |
| \_doresetfont | 58, 70 |
| \_doresetfont | 58, 70, 74 |
| \_dos | 133 |
| \_dosorting | 162 |
| \_dospecials | 52 |