OpTEX

Format Based on Plain \TeX{} and OPmac\textsuperscript{1}

Petr Olšík, 2020

http://petr.olsak.net/optex

OpTEX is \LaTeX{} format with Plain \TeX{} and OPmac. Only \LaTeX{} engine is supported.

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

The main goal of OpTEX is:

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

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

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

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

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

\textsuperscript{1} OPmac package is a set of simple additional macros to Plain \TeX{}. It enables users to take advantage of \LaTeX{} functionality but keeps Plain \TeX{} simplicity. See http://petr.olsak.net/opmac-e.html for more information about it.

\textsuperscript{2} All these features are implemented by \TeX{} macros, no external program is needed.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>Basic macros</td>
<td>35</td>
</tr>
<tr>
<td>2.5</td>
<td>Allocators for \TeX registers</td>
<td>36</td>
</tr>
<tr>
<td>2.6</td>
<td>If-macros, loops, is-macros</td>
<td>38</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Classical \texttt{newif}</td>
<td>38</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Loops</td>
<td>38</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Is-macros</td>
<td>40</td>
</tr>
<tr>
<td>2.7</td>
<td>Setting parameters</td>
<td>41</td>
</tr>
<tr>
<td>2.7.1</td>
<td>Primitive registers</td>
<td>41</td>
</tr>
<tr>
<td>2.7.2</td>
<td>Plain \TeX registers</td>
<td>42</td>
</tr>
<tr>
<td>2.7.3</td>
<td>Different settings than in plain \TeX</td>
<td>42</td>
</tr>
<tr>
<td>2.7.4</td>
<td>Op\TeX parameters</td>
<td>43</td>
</tr>
<tr>
<td>2.8</td>
<td>More Op\TeX macros</td>
<td>47</td>
</tr>
<tr>
<td>2.9</td>
<td>Plain \TeX macros</td>
<td>50</td>
</tr>
<tr>
<td>2.10</td>
<td>Preloaded fonts for text mode</td>
<td>54</td>
</tr>
<tr>
<td>2.11</td>
<td>Scaling fonts in text mode (low-level macros)</td>
<td>55</td>
</tr>
<tr>
<td>2.11.1</td>
<td>The \texttt{fontdef} declarator</td>
<td>56</td>
</tr>
<tr>
<td>2.11.2</td>
<td>The \texttt{fontlet} declarator</td>
<td>56</td>
</tr>
<tr>
<td>2.11.3</td>
<td>Optical sizes</td>
<td>56</td>
</tr>
<tr>
<td>2.11.4</td>
<td>Implementation notes</td>
<td>56</td>
</tr>
<tr>
<td>2.12</td>
<td>The Font Selection System</td>
<td>59</td>
</tr>
<tr>
<td>2.12.1</td>
<td>Terminology</td>
<td>59</td>
</tr>
<tr>
<td>2.12.2</td>
<td>Font families, selecting fonts</td>
<td>59</td>
</tr>
<tr>
<td>2.12.3</td>
<td>Math Fonts</td>
<td>60</td>
</tr>
<tr>
<td>2.12.4</td>
<td>Declaring font commands</td>
<td>60</td>
</tr>
<tr>
<td>2.12.5</td>
<td>Modifying font features</td>
<td>61</td>
</tr>
<tr>
<td>2.12.6</td>
<td>Special font modifiers</td>
<td>61</td>
</tr>
<tr>
<td>2.12.7</td>
<td>How to create the font family file</td>
<td>62</td>
</tr>
<tr>
<td>2.12.8</td>
<td>How to write the font family file with optical sizes</td>
<td>64</td>
</tr>
<tr>
<td>2.12.9</td>
<td>How to register the font family in the Font Selection System</td>
<td>65</td>
</tr>
<tr>
<td>2.12.10</td>
<td>Notices about extension of \texttt{font} primitive</td>
<td>66</td>
</tr>
<tr>
<td>2.12.11</td>
<td>Implementation of the Font Selection System</td>
<td>66</td>
</tr>
<tr>
<td>2.13</td>
<td>Preloaded fonts for math mode</td>
<td>71</td>
</tr>
<tr>
<td>2.14</td>
<td>Math macros</td>
<td>73</td>
</tr>
<tr>
<td>2.15</td>
<td>Unicode-math fonts</td>
<td>82</td>
</tr>
<tr>
<td>2.15.1</td>
<td>Unicode-math macros preloaded in the format</td>
<td>82</td>
</tr>
<tr>
<td>2.15.2</td>
<td>Macros and codes set when \texttt{loadmathfont} is processed</td>
<td>84</td>
</tr>
<tr>
<td>2.15.3</td>
<td>A few observations</td>
<td>90</td>
</tr>
<tr>
<td>2.15.4</td>
<td>Printing all Unicode math slots in used math font</td>
<td>90</td>
</tr>
<tr>
<td>2.16</td>
<td>Scaling fonts in document (high-level macros)</td>
<td>91</td>
</tr>
<tr>
<td>2.17</td>
<td>Output routine</td>
<td>93</td>
</tr>
<tr>
<td>2.18</td>
<td>Margins</td>
<td>96</td>
</tr>
<tr>
<td>2.19</td>
<td>Colors</td>
<td>97</td>
</tr>
<tr>
<td>2.20</td>
<td>The \texttt{.ref} file</td>
<td>102</td>
</tr>
<tr>
<td>2.21</td>
<td>References</td>
<td>103</td>
</tr>
<tr>
<td>2.22</td>
<td>Hyperlinks</td>
<td>104</td>
</tr>
<tr>
<td>2.23</td>
<td>Making table of contents</td>
<td>106</td>
</tr>
<tr>
<td>2.24</td>
<td>PDF outlines</td>
<td>108</td>
</tr>
<tr>
<td>2.24.1</td>
<td>Nesting PDF outlines</td>
<td>108</td>
</tr>
<tr>
<td>2.24.2</td>
<td>Strings in PDF outlines</td>
<td>109</td>
</tr>
<tr>
<td>2.25</td>
<td>Chapters, sections, subsections</td>
<td>110</td>
</tr>
<tr>
<td>2.26</td>
<td>Lists, items</td>
<td>115</td>
</tr>
</tbody>
</table>
2.27 Verbatim, listings ......................................................... 116
  2.27.1 Inline and “display” verbatim ................................. 116
  2.27.2 Listings with syntax highlighting .............................. 121
2.28 Graphics ................................................................. 124
2.29 The \table macro, tables and rules .............................. 130
  2.29.1 The boundary declarator : ..................................... 130
  2.29.2 Usage of the \tabskip primitive ............................... 130
  2.29.3 Tables to given width ........................................... 130
  2.29.4 \eqbox: boxes with equal width across the whole document .... 131
  2.29.5 Implementation of the \table macro and friends ............ 131
2.30 Balanced multi-columns ............................................. 135
2.31 Citations, bibliography ............................................. 137
  2.31.1 Macros for citations and bibliography preloaded in the format .... 137
  2.31.2 The \usebib command ........................................... 140
  2.31.3 The usebib.opm macro file loaded when \usebib is used ......... 141
  2.31.4 Usage of the bib-iso690 style ................................ 144
  2.31.5 Implementation of the bib-iso690 style ....................... 151
2.32 Sorting and making Index ............................................ 155
2.33 Footnotes and marginal notes ...................................... 161
2.34 Styles ................................................................. 163
  2.34.1 \report and \letter styles .................................... 163
  2.34.2 \slides style for presentations ............................... 164
2.35 Logos ................................................................. 167
2.36 Multilingual support ................................................ 168
  2.36.1 Lowercase, uppercase codes ................................... 168
  2.36.2 Hyphenations .................................................... 168
  2.36.3 Multilingual phrases and quotation marks .................... 171
2.37 Other macros ......................................................... 173
2.38 Lua code embedded to the format ................................ 174
2.39 Printing documentation ............................................. 180

Index 184
Chapter 1
User documentation

1.1 Starting with OpTEX

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

    \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

    $\texttt{\fontfam[LMfonts]}$
    \texttt{Hello World! $\textbackslash$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 světě!}) 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)
    $\texttt{\typosize[11/13]}$ % setting default font size and baselineskip (sec. 1.3.2)
    $\texttt{\margins/1 a4 (1,1,1,1)in}$ % setting A4 paper, 1 in margins (section 1.2.1)
    $\texttt{\cslang}$ % Czech hyphenation patterns (section 1.7.1)

    \texttt{Tady je zkušební textík v českém jazyce. $\textbackslash$bye}$

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:

    $\margins/\langle pg \rangle \langle fmt \rangle (\langle left \rangle, \langle right \rangle, \langle top \rangle, \langle bot \rangle) \langle unit \rangle$

    example:
    $\margins/1 a4 (2.5,2.5,2,2)cm$

Parameters are:

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

\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 \langle \left \rangle, \langle \right \rangle, \langle \top \rangle, \langle \bot \rangle can be empty. If both \langle \left \rangle and \langle \right \rangle are nonempty then \hsize is set. Else \hsize is unchanged. If both \langle \left \rangle and \langle \right \rangle are empty then typesetting area is centered in the paper format. The analogical rule works when \langle \top \rangle or \langle \bot \rangle parameter is empty (\vsize instead \hsize is used).

Examples:
\begin{verbatim}
\margins/1 a4 (,,,)mm \% \hsize, \vsize untouched, 
\hspace{9pt} \% typesetting area centered
\margins/1 a4 (,2,,)cm \% right margin set to 2cm
\hspace{9pt} \% \hsize, \vsize untouched, vertically centered
\end{verbatim}

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

Op\TeX uses following paper formats: a4, a4l (landscape a4), a5, a5l, b5, letter and user can declare another own format by \sdef:
\begin{verbatim}
\sdef{_pgs:b5l}{(250,176)mm}
\sdef{_pgs:letterl}{(11,8.5)in}
\end{verbatim}

The \langle fmt \rangle can be also in the form \langle \langle width \rangle, \langle height \rangle \rangle \langle unit \rangle where \langle unit \rangle is optional. If it is missing then \langle unit \rangle after margins specification is used. For example:
\begin{verbatim}
\margins/1 (100,200) (7,7,7,7)mm
\magscale[1414] \margins/1 a4 (,,,)mm
\end{verbatim}


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:
\begin{verbatim}
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
\end{verbatim}

1.2.2 Concept of default page

Op\TeX 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:
\begin{verbatim}
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
\end{verbatim}
The page body should be divided to top insertions (floating tables and figures) followed by a real text and followed by footnotes. Typically, only real text is here.

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

More about the page layout is documented in sections 2.7.4 and 2.17.

1.2.3 Footnotes and marginal notes

The Plain \TeX\’s macro \texttt{\_footnote} can be used as usual. But a new macro \texttt{\_fnote{\langle text\rangle}} is defined. The footnote mark is added automatically and it is numbered on each chapter from one\(^2\). The \langle text\rangle is scaled to 80 \%. User can redefine footnote mark or scaling, as shown in the section 2.33.

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

\begin{verbatim}
Text in a paragraph\fnotemark{1}\fnote{First notice}... % a "normal" footnote
\table{...}{...}\fnotemark{2}\fnotemark{1}\fnote{Second notice}
\fnotetext{Third notice}
...
\table{...}{...}\fnotemark{1}\fnote{Fourth notice}
\fnotetext{Second notice}
\fnotetext{Third notice}
...
\table{...}{...}\fnotemark{1}\fnote{First notice}
\fnotetext{Second notice}
\fnotetext{Third notice}
\end{verbatim}

The marginal note can be printed by the \_mnote{\langle text\rangle} macro. The \langle text\rangle is placed to the right margin on the odd pages and it is placed to the left margin on the even pages. This is done after second \TeX run because the relevant information is stored in an external file and read from it again. If you need to place the notes only to the fixed margin write \_fixmnotes\_right or \_fixmnotes\_left.

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

1.3 Fonts

1.3.1 Font families

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

If you write \texttt{\_fontfam[?]} then all font families registered in \TeX are listed on the terminal and in the log file. If you write \texttt{\_fontfam[catalog]} then a catalog of all fonts registered in

\(^2\) You can declare \texttt{\_fnotenumglobal} if you want footnotes numbered in whole document from one or \texttt{\_fnotenumpages} if you want footnotes numbered at each page from one. Default setting is \texttt{\_fnotenumchapters}
OpTeX and available in your TeX system is printed. The instructions how to register your own font family is appended in such catalog.

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

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

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

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

\begin{verbatim}
This is \{\texttt{em important}\} text.  \% = This is \{\textit{important}\}/\} text.
\textit{This is \{em important\} text.}  \% = This is/\} \{\texttt{rm important}\} text.
\textbf{This is \{em important\} text.}  \% = This is \{\texttt{bi important}\}/\} text.
\textbf{This is \{em important\} text.}  \% = This is/\} \{\texttt{bf important}\} text.
\end{verbatim}

More about the OpTeX Font Selection System is written in the technical documentation in the section 2.12. You can mix more font families in your document, you can declare your own variant selectors or modifiers etc.

### 1.3.2 Font sizes

The command \(\texttt{\textbackslash typosize[\langle fontsize\rangle/\langle baselineskip\rangle]}\) 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 \(\texttt{\textbackslash ptunit}\) which is 1pt by default. You can change the unit by the command \(\texttt{\textbackslash ptunit=\langle something-else\rangle}\), for instance \(\texttt{\textbackslash ptunit=1mm}\) enlarges all font sizes declared by \(\texttt{\textbackslash typosize}\). Examples:

\begin{verbatim}
\texttt{\textbackslash typosize[10/12]} \% default of Plain TeX
\texttt{\textbackslash typosize[11/12.5]} \% font 11pt, baseline 12.5pt
\texttt{\textbackslash 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 \(\texttt{\textbackslash typosize}\) for example) then you must first finalize the paragraph before closing the group: \(\texttt{\textbackslash typosize[12/14] \ldots\langle text of paragraph\rangle \ldots \texttt{\textbackslash par}}\).

The command \(\texttt{\textbackslash typoscale[\langle font-factor\rangle/\langle baselineskip-factor\rangle]}\) 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}
\texttt{\textbackslash typoscale[800/800]} \% fonts and baselineskip re-size to 80 \%
\texttt{\textbackslash typoscale[magstep2/]} \% fonts bigger 1,44times (\texttt{\textbackslash magstep2 expands to 1440})
\end{verbatim}

First usage of \(\texttt{\textbackslash typosize}\) or \(\texttt{\textbackslash typoscale}\) macro in your document sets so called *main values*, i.e. main font size and main baselineskip. They are internally saved in registers \(\texttt{\textbackslash mainfsize}\) and \(\texttt{\textbackslash mainbaselineskip}\).

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.
The size of the current font can be changed by the command \texttt{\the\fontsize[⟨font-size⟩]} or can be rescaled by \texttt{\the\fontscale[⟨factor⟩]}. These macros don’t change math fonts sizes nor baselineskip.

There is “low level” \texttt{\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{\setfontsize{at15pt}\currvar} sets current variant to 15pt.

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

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

1.3.3 Typesetting math

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

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

Hundreds math symbols and operators like in AMSTEX are accessible. For example \texttt{\alpha}, \texttt{\geq}, \texttt{\sum}, \texttt{\sphericalangle}, \texttt{\bumpeq}, \texttt{\bumpequal}. See AMSTEX manual or \texttt{Typesetting Math with PostScript} 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−X Y Z
\frak % fracture abc–xyz, ABC–XYZ
\bbchar % double stroked letters ABC–XYZ
\bf % sans serif bold abc–xyz, ABC–XYZ
\bi % sans serif bold slanted abc×xyz, ABC−XYZ
\end{verbatim}

The last two selectors \texttt{\bf} and \texttt{\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 2.15.2 where definitions of Unicode math variants of \texttt{\bf} and \texttt{\bi} selectors are documented.

The math fonts can be scaled by \texttt{\typosize} and \texttt{\typoscale} macros. Two math fonts collections are prepared: \texttt{\normalmath} for normal weight and \texttt{\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{\mathbox{⟨text⟩}} inside math mode. It behaves as \texttt{\hbox{⟨text⟩}} (i.e. the ⟨text⟩ is printed in horizontal non-math mode) but the size of the ⟨text⟩ is adapted to the context of math size (text or script or scriptscript). Moreover, there is the macro \texttt{\mathstyles{⟨math list⟩}} which depends on the current math style. It is documented at the end of the section 2.14.
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 (it means newline) macro 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.

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 (it means newline) macro to the breakpoints.

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

1.4.2 Another numbered objects

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

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

\begin{alignno}a^2+b^2 &= c^2 \cr c &= \sqrt{a^2+b^2} & \eqmark \cr\end{alignno}

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

\begin{caption}The dependency of the computer-dependency on the age.\cskip\noindent\hfil\table{rl}\end{caption}
This example produces:

**Table 1.4.1** 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>

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

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

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

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

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

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

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

Three independent numbers are used in this example. One for Theorems and Corollaries second for Definitions and third for Examples. The user can write `\theorem Let $M$ be...` and the new paragraph is started with the text: Theorem 1.4.1. Let $M$ be... You can add an optional parameter in brackets. For example, `\theorem [(L'Hôpital's rule)] Let $f$, $g$ be...` is printed like Theorem 1.4.2 (L'Hôpital’s rule). Let $f$, $g$ be...

### 1.4.3 References

Each automatically numbered object documented in sections 1.4.1 and 1.4.2 can be referenced if optional parameter `[⟨label⟩]` is appended to `\chap, \sec, \secc, \caption/t, \caption/f`

\(^3\) This feature can be changed, see the section 2.25 in the technical documentation.
or \texttt{\emph{eqmark}}. The alternative syntax is to use \texttt{\label{\langle label\rangle}} before mentioned commands (not necessarily directly before). The reference is realized by \texttt{\ref{\langle label\rangle}} or \texttt{pgref{\langle label\rangle}}. Example:

\begin{verbatim}
\sec[beatle] About Beatles
\end{verbatim}
\begin{verbatim}
\noindent\hfil\table{rl}{...} \% the table
\cskip
\caption{\[\texttt{comp-depend}\]} The dependency of the \texttt{comp-dependency} on the age.
\end{verbatim}
\begin{verbatim}
\label[pythagoras]
\$$ a^2 + b^2 = c^2 \texttt{\emph{eqmark}} $$
\end{verbatim}

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

If there are forward referenced objects then user have to run \TeX\ twice. During each pass, the working \texttt{.\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 \texttt{\label{\langle label\rangle}} before the \texttt{\theorem, \definition} etc. (macros defined with \texttt{\numberedpar}) if you want to reference these numbered objects. You can’t use \texttt{\theorem[\langle label\rangle]} because the optional parameter is reserved to another purpose here.

You can create a reference to whatever else by commands \texttt{\label{\langle label\rangle}} \texttt{\wlabel{\langle text\rangle}}. The connection between \texttt{\langle label\rangle} and \texttt{\langle text\rangle} is established. The \texttt{\ref[\langle label\rangle]} will print \texttt{\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 \texttt{\showlabels}. The labels are printed at their destination places after such declaration.

### 1.4.4 Hyperlinks, outlines

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

The last object is an external link and it is colored by \texttt{\langle color-out\rangle}. Others links are internal and they are colored by \texttt{\langle color-in\rangle}. Example:

\begin{verbatim}
\hyperlinks \Blue \Green \% internal links blue, URLs green.
\end{verbatim}

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 \texttt{\_pgborder, \_tocborder, \_citeborder, \_refborder} and \texttt{\_urlborder} as the triple of RGB components of the used color. Example:

\begin{verbatim}
\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
\end{verbatim}

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

The hyperlinked footnotes can be activated by \texttt{\fnotelinks \langle color-fnt\rangle \langle color-fnf\rangle} where footnote marks in text have \texttt{\langle color-fnt\rangle} and the same footnote marks in footnotes have
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)\}:\{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

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:

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

For example:

\begitems
* First idea
* Second idea in subitems:
  \begitems \style i

\enditems

---

\footnote{More exactly, there are the same rules as for \code command, see section 1.4.7.}
* 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 \texttt{\sdef{\_item:\{\text\}}{\{\text\}}}{}. Default item can be set by \texttt{\defaultitem={\{\text\}}}{}. The list environments can be nested. Each new level of items is indented by next multiple of \texttt{\indent} value which is set to \texttt{\parindent} by default. The \texttt{\ilevel} register says what level of items is currently processed. Each \texttt{\begitems} starts \texttt{\everylist} tokens register. You can set, for example:

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

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

1.4.6 Tables

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

\begin{verbatim}
\table{||lc|r||}{
  Month & commodity & price \\
  January & notebook & \$ 700 \\
  February & skateboard & \$ 100 \\
  July & yacht & k\$ 170 
}
\end{verbatim}

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 \texttt{l}, \texttt{r}, \texttt{c} declarators, you can use the \texttt{p{\{\text\}}} 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:

- \texttt{p{\{\text\}\texttt{\fL}}} fit left, i.e. left justified, ragged right,
- \texttt{p{\{\text\}\texttt{\fR}}} fit right, i.e. right justified, ragged left,
- \texttt{p{\{\text\}\texttt{\fC}}} fit center, i.e. ragged left plus right,
- \texttt{p{\{\text\}\texttt{\fS}}} fit special, short one-line pararaph centered, long paragraph normal,
- \texttt{p{\{\text\}\texttt{\fX}}} fit extra, left-right justified but last line centered.
You can use \((text)\) in the \(declaration\). Then this text is applied in each line of the table. For example \(r(\kern10\text{pt}) l\) adds more 10 pt space between \(r\) and \(l\) rows.

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

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

- \(\crl\) ... the end of the row with a horizontal line after it.
- \(\crl1\) ... the end of the row with a double horizontal line after it.
- \(\crl{}\) like \(\crl\) but the horizontal line doesn’t intersect the vertical double lines.
- \(\crl1{}\) like \(\crl\) but horizontal line is doubled.
- \(\crl{\langle\text{list}\rangle}\) ... like \(\crl\) but the lines are drawn only in the columns mentioned in comma separated \(\langle\text{list}\rangle\) of their numbers. The \(\langle\text{list}\rangle\) can include \(\langle\text{from}\rangle\)–\(\langle\text{to}\rangle\) declarators, for example \(\crl{1\text{--}3,5}\) is equal to \(\crl{1,2,3,5}\).

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

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

- \(\everytable={}\) % code used in \textbackslash vbox before table processing
- \(\thistable={}\) % code used in \textbackslash 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
- \(\tablinesepace=2\text{pt}\) % additional vert. space before/after horizontal lines
- \(\vkkern=1\text{pt}\) % space between lines in double vertical line
- \(\hhkern=1\text{pt}\) % space between lines in double horizontal line
- \(\tabskip=0\text{pt}\) % space between columns
- \(\tabskipl=0\text{pt}\ \tabskipr=0\text{pt}\) % space before first and after last column

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

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

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

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

creates the following result:

<table>
<thead>
<tr>
<th></th>
<th>first</th>
<th>second</th>
<th>third</th>
</tr>
</thead>
<tbody>
<tr>
<td>seven</td>
<td>eight</td>
<td>nine</td>
<td></td>
</tr>
</tbody>
</table>
The \texttt{\vspan\{number\}\{text\}} shifts the \texttt{\{text\}} down in order it looks like to be in the center of the \texttt{\{number\}} lines (current line is first). You can use this for creating tables like in the following example:

\thisstable{\vrule height 20pt depth10pt width0pt}
\table{\baselineskip=20pt \tablinespace=0pt \rulewidth=.8pt}
\table{|8{c|}}{\crlp{3-8}
\mspan2[c]{} & \mspan3[c]{\text{Singular}} & \mspan3[c]{\text{Plural}} & \crlp{3-8}
\mspan2[c]{} & \mspan3[c]{\text{Neuter}} & \mspan3[c]{\text{Masculine}} & \mspan3[c]{\text{Feminine}} & \mspan3[c]{\text{Masculine}} & \mspan3[c]{\text{Feminine}} & \mspan3[c]{\text{Neuter}} & \crlp{3-8}
\vspan2{I} & \text{Inclusive} & \mspan3[c]{} & \mspan3[c]{} & \mspan3[c]{\text{O}} & \mspan3[c]{\text{X}} & \mspan3[c]{\text{X}} & \crlp{2-8}
\vspan2{II} & \text{Exclusive} & \mspan3[c]{} & \mspan3[c]{} & \mspan3[c]{\text{O}} & \mspan3[c]{\text{X}} & \mspan3[c]{\text{X}} & \crlp{2-8}
\vspan2{III} & \text{Informal} & \mspan3[c]{} & \mspan3[c]{} & \mspan3[c]{\text{O}} & \mspan3[c]{\text{X}} & \mspan3[c]{\text{X}} & \crlp{2-8}
}\endtable

Singular | Plural | Neuter | Masculine | Feminine | Masculine | Feminine | Neuter
--------|--------|--------|-----------|----------|-----------|----------|--------
I       |        | O      | X         |          |           |          |        
II      |        | X      | X         |          |           |          |        
III     |        | O      | X         |          |           | X        | O      

The \texttt{\vspan\{number\}\{text\}} parameter of \texttt{\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{\vspan\{2.1\}\{text\}}.

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

The \texttt{\c}, \texttt{\l}, \texttt{\r} and \texttt{\p} are default “declaration letters” but you can define more such letters by \texttt{\def\_tabdeclare⟨letter⟩\{⟨left⟩##⟨right⟩\}}. More about it is in technical documentation in section 2.29.5. See the definition of the \texttt{\tabdeclarec} macro, for example.

The \texttt{\}: columns boundary declarator is described in section 2.29.1. The tables with given width can be declared by \texttt{\to⟨size⟩} or \texttt{\ppto⟨size⟩}. More about it is in section 2.29.3 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\{char\}}. For example \texttt{\activettchar\{\texttt{\#\}}\texttt{\#\}} declares the character \texttt{\#} for in-line verbatim markup. And you can use \texttt{\relax} for verbatim \texttt{\relax} (for example). Another alternative of printing in-line verbatim text is \texttt{\code\{⟨text⟩\}} (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\ldots\endtt} environment will follow the line numbering. \TeX{} 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\ldots\endtt} 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 \texttt{\everytt}, use \texttt{\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 `\def` 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 \def!{?}\def?{!}
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⟩`:

\`
\-=\+\%
\`

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

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

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

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

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

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

If the code is read by `\verbinput` and it uses two characters at the front of the lines as a comment lines, you can set them by `\commentchars{first}{second}`. Such comments are fully interpreted by TeX (i.e. not verbatim). Section 2.27.1 (page 120) 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 2.23.

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

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

1.5.2 Making the index

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

The `\ii` command (insert to index) declares the word separated by the space as the index item. This declaration is represented as invisible 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 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 \TeX{}book 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 by the \iis command. Example:

The \iis 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 Bib\TeX{}ing

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

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

The \rcite[⟨labels⟩] creates the same list as \cite[⟨labels⟩] but without the outer brackets. Example: \rcite[tnb], pg.-13 creates [4, pg. 13].
The `\cite{<label>}`{<text>} prints the `<text>` only, but the entry labeled `<label>` is decided as to be cited. If `\hyperlinks` is used then `<text>` is linked to the references list.

You can define alternative formatting of `\cite` command. Example:

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

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

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

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

References created manually using `\bib[(<label>)]` command.

\[
\bib [tst] P. Olšák. \textit{Typografický systém \TeX{}}.
\textit{269-s. Praha: CSTUG, 1995.}
\]

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

\[
\bib [tbn] = \{Olšák, 2001\}
\]


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

\[
\usebib/c (<style>) <bib-base> \% sorted by \cite-order (c=cite),
\usebib/s (<style>) <bib-base> \% sorted by style (s=style).
\]

\% example:

\[
\usebib/s (simple) op-example
\]

The `<bib-base>` is one or more *.bib database source files (separated by spaces and without extension) and the `<style>` is the part of the filename bib-<style>.opm where the formatting of the references list is defined. Op\TeX\ supports \texttt{simple} or \texttt{iso690} styles. The features of the \texttt{iso690} style is documented in the section 2.31.4 in detail.

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

1.6 Graphics

1.6.1 Colors

Op\TeX\ 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

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

20
The command \morecolors reads more definitions of color selectors. There is about 300 color names like \DeepPink, \Chocolate etc. If there are numbered variants of the same name, then the letters B, C, etc. are appended to the name in OpTEX. For example \Chocolate is Chocolate1, \ChocolateB is Chocolate2 etc.

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

The basic colors \Blue, \Red, \Cyan, \Yellow etc. are defined with CMYK components using \setcmykcolor. On the other hand, you can define a color with three RGB components and \morecolors defines such RGB colors. By default, the color model isn’t converted but only stored to PDF output for each used color. 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\footnote{Printed output is more equal to the monitor preview specially if you are using ICC profile for your printer.} for printing.

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

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

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.

Usage: \coloron\langle background\rangle\langle foreground\rangle\langle text\rangle

The \coloron can be defined as follows:

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

1.6.2 Images

The \inspic {\langle filename\rangle.\langle extension\rangle} or \inspic {\langle filename\rangle.\langle extension\rangle\langle space\rangle} inserts the picture stored in the graphics file with the name \langle filename\rangle.\langle extension\rangle to the document. You 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={img/} supposes that image files are in img subdirectory. Note: the directory name must end by / in the \picdir declaration.
Inkscape\(^6\) is able to save a picture to PDF and labels of the picture to another file\(^7\). This second file should be read by T\(\LaTeX\) in order to print labels in the same font as document font. Op\(\LaTeX\) supports this feature by \texttt{\textbackslash inkspic \{filename\}.pdf}\ command. It reads and displays both: PDF image and labels generated by Inkscape.

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

### 1.6.3 PDF transformations

All typesetting elements are transformed by linear transformation given by the current transformation matrix. The \texttt{\textbackslash pdfsetmatrix \{⟨a⟩ ⟨b⟩ ⟨c⟩ ⟨d⟩\}} command makes the internal multiplication with the current matrix so linear transformations can be composed. One linear transformation given by the \texttt{pdfsetmatrix} above transforms the vector \([0,1]\) to \([⟨a⟩,⟨b⟩]\) and \([1,0]\) to \([⟨c⟩,⟨d⟩]\). The stack-oriented commands \texttt{\textbackslash pdfsave} and \texttt{\textbackslash 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 T\(\LaTeX\)'s point of view as from transformation point of view when \texttt{\textbackslash pdfrestore} is processed. Due to this fact the \texttt{\textbackslash pdfsave\rlap{⟨transformed text⟩}\textbackslash pdfrestore} or something similar is recommended.

Op\(\LaTeX\) provides two special transformation macros \texttt{\textbackslash pdfscale} and \texttt{\textbackslash pdfrotate}:

\begin{verbatim}
\textbackslash pdfscale\{⟨horizontal-factor⟩\}\{⟨vertical-factor⟩\}
\textbackslash pdfrotate\{⟨angle-in-degrees⟩\}
\end{verbatim}

These macros simply calls the properly \texttt{\textbackslash 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: \texttt{\textbackslash pdfsave \textbackslash pdfrotate\{30\}\textbackslash pdfscale\{-2\}\{2\}\textbackslash rlap\{text1\}\textbackslash pdfrestore}

\begin{verbatim}
% text1 is scaled two times and it is reflected about vertical axis
% and next it is rotated by 30 degrees left.
\end{verbatim}

\begin{verbatim}
\textit{second:}\ \text\{\textbackslash pdfsave \textbackslash pdfscale\{-2\}\{2\}\text\{\textbackslash pdfrotate\{30\}\text\{\textbackslash rlap\{text2\}\text\{\textbackslash pdfrestore\}

% text2 is rotated by 30 degrees left then it is scaled two times
% and reflected about vertical axis.
\end{verbatim}

\begin{verbatim}
\textit{third:}\ \text\{\textbackslash pdfsave \text\{\textbackslash pdfrotate\{-15.3\}\text\{\textbackslash pdfsetmatrix\{2 0 1.5 2\}\text\{\textbackslash rlap\{text3\}\text\{\textbackslash pdfrestore\}

% first slanted, then rotated by 15.3 degrees right
\end{verbatim}

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

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

Example: \texttt{\textbackslash frame\{\text\{\textbackslash transformbox\{\text\{\textbackslash pdfscale\{1\}\{1.5\}\text\\{\textbackslash pdfrotate\{-10\}\}\text\{\textbackslash transformbox\{\text\{\text\\{moj\}\}\}}\}

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

---

\(^6\) A powerfull and free wysiwyg editor for creating vector graphics.

\(^7\) Chose “Omit text in PDF and create LaTeX file” option.
1.6.4 Ovals, circles

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

\ovalparams=\{\roundness=2pt % diameter of circles in the corners
\fcolor=\Yellow % color used for filling oval
\lcolor=\Red % line color used in the border
\lwidth=0.5bp % line width in the border
\shadow=N % use a shadow effect
\overlapmargins=N % ignore margins by surrounding text
\hhkern=0pt \vvkern=0pt\} % left-righ margin, top-bottom margin

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

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

The \incircle\{\ratio=1.8\}\{\text\} creates a box like this \text. The \ratio parameter means width/height. The usage is analogical like for oval. The default parameters are

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

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

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

1.6.5 Putting images and texts wherever

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

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

1.7 Others

1.7.1 Using more languages

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

en(USenglish) enus(USenglishmax) it(italian) ia(interlingua) id(indonesian) cs(czech) sk(slovak)
de(nGerman) fr(french) pl(polish) cy(welsh) da(danish) es(spanish) sl(slovenian) hu(hungarian) tr(turkish)
et(estonian) eu(basque) ga(irishe) nb(bokmal) nn(nynorsk) nl(dutch) pt(portuguese) ro(romanian) hr(croatian)
zh(pinyin) is(icelandic) sbh(upperSorbian) af(afrikaans) gl(galician) kmr(kurmanji) tk(turkmen) la(latin) lac(latin)
lat(liturgicallatin) elm(mongolian)
For compatibility with e-plain macros, there is the command \uselanguage{⟨language⟩}. The parameter \(⟨language⟩\) is long form of language name, i.e. \uselanguage{Czech} works the same as \cslang. The \uselanguage parameter is case insensitive.

For compatibility with \csplain, there are macros \ehyph, \chyph, \shyph which are equivalent to \enlang, \cslang and \sklang.

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

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

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

If you declare \⟨iso-code⟩quotes, then the control sequences \" and \' should be used like this: \"⟨quoted text⟩" or \'⟨quoted text⟩' (note that the terminating character is the same but it isn’t escaped). This prints language dependent normal or alternative quotes around ⟨quoted text⟩. The language is specified by ⟨iso-code⟩. Op\TeX\ declares quotes only for Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English (\csquotes, \dequotes, ..., \enquotes). You can simply define your own quotes as shown in languages.omp file. The \" is used for quotes visually 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 \⟨iso-code⟩quotes\altquotes in such case.

1.7.2 Pre-defined styles

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

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

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

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

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

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

### 1.7.3 Loading other macro packages

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

```
\load [qrcode, tikz]
```

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

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

### 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 `\lorem[⟨number⟩]` or `\lorem[⟨from⟩-⟨to⟩]`. It prints a paragraph (or paragraphs) with neutral text. The numbers ⟨number⟩ or ⟨from⟩, ⟨to⟩ must be in the range 1 to 150 because there are 150 paragraphs with neutral text prepared for you. The `\lipsum` macro is equivalent to `\lorem`. Example: `\lipsum[1-150]` prints all prepared paragraphs.

### 1.7.5 Logos

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

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

### 1.7.6 The last page

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

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

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

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

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

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

1.8 Summary

\tit Title (terminated by end of line)
\chap Chapter Title (terminated by end of line)
\sec Section Title (terminated by end of line)
\secc Subsection Title (terminated by end of line)
\maketoc % table of contents generation
\ii item1,item2 % insertion the items to the index
\makeindex % the index is generated
\label [labname] % link target location
\ref [labname] % link to the chapter, section, subsection, equation
\pgref [labname] % link to the page of the chapter, section, ...
\caption/t % a numbered table caption
\caption/f % a numbered caption for the picture
\eqmark % a numbered equation
\begitems % start list of the items
\enditems % end of list of the items
\begtt % start verbatim text
\endtt % end verbatim text
\activettchar X % initialization character X for in-text verbatim
\code % another alternative for in-text verbatim
\verbinput % verbatim extract from the external file
\begmulti num % start multicolumn text (num columns)
\endmulti % end multicolumn text
\cite [labnames] % refers to the item in the lits of references
\rcite [labnames] % similar to \cite but [] are not printed.
\sortcitations \shortcitations \nonumcitations % cite format
\bib [labname] % an item in the list of references
\usebib/? (style) bib-base % direct using of .bib file, ? in {s,c}
\load [<filenames>] % loading macro files
\fontfam [FamilyName] % selection of font family
\typosize [font-size/baselineskip] % size setting of typesetting
\typoscale [factor-font/factor-baselineskip] % size scaling
\thefontsize [size] \thefontscale [factor] % current font size
\inspic file.ext % insert a picture, extensions: jpg, png, pdf
\table {rule}{data} % macro for the tables like in LaTeX
\fnote {text} % footnote (local numbering on each page)
\mnote {text} % note in the margin (left or right by page number)
\hyperlinks {color-in}{color-out} % PDF links activate as clickable
\outlines {level} % PDF will have a table of contents in the left tab
\magscale [factor] % resize typesetting, line/page breaking unchanged
\margins/pg format (left, right, top, bottom)unit % margins setting
\report \letter \slides % style declaration macros

26
1.9 Compatibility with Plain TeX

All macros of Plain TeX are re-written in OpTeX. Common macros should work in the same sense as in original Plain TeX. Internal control sequences like \p@ or \f@t are removed and mostly replaced by control sequences prefixed by _ (like _this). If you need to use basic set of such Plain TeX control sequences (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 \", \', \v, \u, \=, \^, \., \H, \~, \` 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 \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.

\begin{verbatim}
\%% 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 `\^=7 %
\catcode `\^^K=7 % circumflex and uparrow are for superscripts
\catcode `\^^A=8 % downarrow is for subscripts
\catcode `\^^I=10 % ascii tab is a blank space
\catcode `\_=11 % underline can be used in control sequences
\catcode `\~=13 % tilde is active
\catcode `\^^a0=13 % non breaking space in Unicode
\catcode 127=12 % normal character
\def\optexversion{Beta 0.17 Nov.2020}
\def\fmtname{OpTeX}
\newlinechar=`\^^J
\ifx\directlua\undefined
\fi
\end{verbatim}

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.

\begin{verbatim}
\%% OpTeX version
\def\optexversion{Beta 0.17 Nov.2020}
\def\fmtname{OpTeX}
\newlinechar=`\^^J
\ifx\directlua\undefined
\fi
\end{verbatim}

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

\begin{verbatim}
\%% Engine testing:
\ifx\directlua\undefined
\newlinechar=`\``\J
\fi
\end{verbatim}
The basic macros for macro file syntax is defined, i.e. \_endcode, \_doc and \_cod. The \_codedecl will be re-defined later.

Individual *.opm macro files are read.

The file optex.lua is embedded into the format as byte-code. It is documented in section 2.38.

The \_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 \TeX{} are used and defined with _ prefix. The user can be sure that when he/she does \def\foo then internal macros of \TeX{} nor \TeX{} primitives will be not damaged. For example \def\if{...} will not damage macros because \TeX{}’s macros are using \_if instead of \if. All \TeX{} primitives are initialized with two representative control sequences: \word and \_word, for example \hbox and \_hbox. The first alternative is reserved for users or such control sequences can be re-defined by user.

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

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

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

\public <list of control sequences> ;

For example \public \foo \bar ; does \let\foo=\_foo, \let\bar=\_bar.
At the end of each code segment in \TeX{}, the \_public macro is used. You can see, what macros are defined for public usage in such code segment.

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

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

2.2.2 Name space of control sequences for users

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

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

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

\int_a^b \quad \text{is interpreted as \int_a^b}
\max_M \quad \text{is interpreted as \max_M}
\alpha_{ij} \quad \text{is interpreted as \alpha_{ij}}

This feature is implemented using lua code at input processor level, see the section 2.14 for more details. You can deactivate this feature by \mathsboff. After this, you can still write $\int_a^b$ (Unicode) or $\int_a^b$ without problems but \int_a^b yields to undefined control sequence
\int_a. You can activate this feature again by \mathsbon. The effect will take shape from next line read from input file.

- Control sequences in the form $\langle pkg\rangle \langle word \rangle$ is intended for package writers as internal macros for a package with $\langle pkg \rangle$ identifier, see section 2.2.4.

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

### 2.2.3 Macro files syntax

Each segment of OpT\(\varepsilon\)X 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{codedecl} and ends by \_\texttt{endcode}. The \_\texttt{endcode} is equivalent for \texttt{\endinput}, so documentation can follow. The \_\texttt{codedecl} has syntax:

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

If the mentioned \texttt{sequence} is defined, then \_\texttt{codedecl} does the same as \texttt{\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 \_doc ... \_cod pair between the macro lines. The documentation text should be here. It is ignored when macros are read but it can be printed using \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{qr_utfstring} from \texttt{qrcode.opm} package.

The package writer needs not write repeatedly \_\texttt{pkg\_foo \_pkg\_bar} etc. again and again in the macro file.\footnote{We have not adapted 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\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{endnamespace} 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{newifi\_if⟨pkg⟩\_bar}, for example \_\texttt{newifi\_ifqr\_incorner}. Then the control sequences \_\texttt{qr\_incornertrue} and \_\texttt{qr\_incorneralse} can be used (or the sequences \_\texttt{\.incornertrue} and \_\texttt{\.incorneralse} when \_\texttt{namespace\{qr\}} is used).

### 2.2.5 Summary about rules for external macro files published for OpT\(\varepsilon\)X

If you are writting a macro file which is intended to be published for OpT\(\varepsilon\)X, 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 OpT\(\varepsilon\)X and primitive name space in read only mode if there is not explicit and documented reason to redefine them.
- Use $\langle pkg \rangle \langle name \rangle$ for your internal macros or $\langle . \rangle$ for the $\langle \_namespace \{⟨pkg⟩\} \rangle$ 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 ⟨filename⟩.opm where ⟨filename⟩ 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

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

\ea is useful shortcut for \expandafter. We recommend to use always the private form of \_ea because there is high probability that \ea will be redefined by the user.

\public (sequence) (sequence) ... ; does \let \{sequence\} = \_{sequence} for all sequences.
\private (sequence) (sequence) ...; does \let \{sequence\} = \_\{sequence\} for all sequences.
\xargs \{what\} (sequence) (sequence) ... ; does \{what\}(sequence) for each sequences.

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

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

The \optexversion and \fmtname are defined in the optex.ini file. Maybe, somebody will need a private version of these macros.
The \_mathsbon and \_mathsboff are defined in math-macros.opm file. Now, we define the macros \_namespace \{\textit{pkg label}\}, \_resetnamespace \{\textit{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 \ea#1\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 \pdfcopyfont \copyfontinfont
\let \pdfxform \saveboxresource
\let \pdflastxform \lastsavedboxresourceindex
\let \pdffontxform \useboxresource
\let \pdflastimage \lastsavedimageresourceindex
\let \pdffontlastimage \lastsavedimageresourcepages
\let \pdffontximage \useimageresource
\let \pdflastximage \lastsavedimageresourcepages
\let \pdffontximagepages \useimageresource
\let \pdfsavepos \savepos
\let \pdflastxpos \lastxpos
\let \pdflastxpos \lastxpos
\let \pdflastypos \lastypos
\let \pdfoutput \outputmode
\let \pdfpxdimen \pxdimen
\let \pdfinsertht \insertht
\let \pdfnormaldeviate \normaldeviate
\let \pdfuniformdeviate \uniformdeviate
\let \pdfsetrandomseed \setrandomseed
\let \pdfrandomseed \randomseed
\let \pdfprimitive \primitive
\let \ifpdfprimitive \ifprimitive
\let \ifpdfabsnum \ifabsnum
\let \ifpdfabsdim \ifabsdim
```

33
\_directlua {tex.enableprimitives('pdf', {'tracingfonts'})}

\protecteddef \_pdffontname {\_pdffeedback fontname}
\protecteddef \_pdffontobjnum {\_pdffeedback fontobjnum}
\protecteddef \_pdffontsize {\_pdffeedback fontsize}
\protecteddef \_pdfpageref {\_pdffeedback pageref}
\protecteddef \_pdffontattr {\_pdffeedback fontattr}
\protecteddef \_pdfpkresolution {\_pdffeedback pkresolution}
\protecteddef \_pdfinclusioncopyfonts {\_pdffeedback inclusionscopyfonts}
\protecteddef \_pdfinclusioncopyfonts {\_pdffeedback inclusionscopyfonts}
\protecteddef \_pdfgentounicode {\_pdffeedback gentounicode}
\protecteddef \_pdfglyphbtouindices {\_pdffeedback glyphbtouindices}
\protecteddef \_pdfcompresslevel {\_pdffeedback compresslevel}
\protecteddef \_pdfobjcompresslevel {\_pdffeedback objcompresslevel}
\protecteddef \_pdfdecimaldigits {\_pdffeedback decimaldigits}
\protecteddef \_pdfgamma {\_pdffeedback gamma}
\protecteddef \_pdfimageresolution {\_pdffeedback imageresolution}
\protecteddef \_pdfimageapplygamma {\_pdffeedback imageapplygamma}
\protecteddef \_pdfimagegamma {\_pdffeedback imagegamma}
\protecteddef \_pdfimagebicolor {\_pdffeedback imagebicolor}
\protecteddef \_pdfimageaddfilename {\_pdffeedback imageaddfilename}
\protecteddef \_pdfimageaddfilename {\_pdffeedback imageaddfilename}
\protecteddef \_pdfminorversion {\_pdffeedback minorversion}
\protecteddef \_pdfuniqueresname {\_pdffeedback uniqueresname}
\protecteddef \_pdfhorigin {\_pdffeedback horigin}
\protecteddef \_pdfvorigin {\_pdffeedback vorigin}
\protecteddef \_pdflinkmargin {\_pdffeedback linkmargin}
2.4 Basic macros

We define first bundle of basic macros.

\begin{verbatim}
\_protected\_edef\_pdfdestmargin \pdfvariable destmargin
\_protected\_edef\_pdfthreadmargin \pdfvariable threadmargin
\_protected\_edef\_pdfpagesattr \pdfvariable pagesattr
\_protected\_edef\_pdfpageattr \pdfvariable pageattr
\_protected\_edef\_pdfpageresources \pdfvariable pageresources
\_protected\_edef\_pdfxformattr \pdfvariable xformattr
\_protected\_edef\_pdfxformresources \pdfvariable xformresources
\_protected\_edef\_pdfpkmode \pdfvariable pkmode
\_pdfminorversion = 5
\_pdfobjcompresslevel = 2
\_pdfcompresslevel = 9
\_pdfdecimaldigits = 3
\_pdfpkresolution = 600
\end{verbatim}

\section{Basic macros}

We define first bundle of basic macros.

\begin{verbatim}
\_public\_group\_egroup\_empty\_space\_null\_wlog\_bslash\_nbb\_pcent ;
\end{verbatim}

\_public\_group\_egroup\_empty\_space\_null\_wlog ;

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

\begin{verbatim}
\_edef \_bslash \_csstring\\
\_edef \_nbb \_bslash\_bslash
\_edef \_pcent \_csstring\%
\end{verbatim}

\_public\_bslash\_nbb\_pcent ;

\_def \{\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 .

\_xdef \{\text\} is equivalent to \xdef{\text\}.

\_slet \{\textA\} \{\textB\} is equivalent to \let \{\textA\} = \{\textB\}.

\_def \{\char\} \{\body\} puts the \{\char\} as active character and defines it as \{\body\}. You can declare a macro with parameters too. For example \_adef @#1{...$1...}.
\def \_def \_adef #1{\_catcode`#1=13 \_begingroup \_lccode`\~=`#1\_lowercase{\_endgroup\_def~}}

\_public \adef ;

\cs \{\langle text \rangle\} is only a shortcut to \csname \langle text \rangle\endcsname, but you need one more \_ea if you need to get the real control sequence \langle text \rangle.

\_def \_cs #1{\csname#1\endcsname}

\_def \_trycs#1#2{\_ifcsname #1\_endcsname \_csname #1\_endcsname \_else #2\_fi}

\_public \cs \trycs ;

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

\_long\_def \_addto #1#2{\_ea\_def\_ea#1\_ea{#1#2}}

\_public \addto ;

\_def \_opwarning #1{\_wterm{WARNING: #1.}}

\_public \opwarning ;

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

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

\_def \_tracingall{\_tracingonline=1 \_loggingall}

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

\_byehook{\_ifx \_initunifonts \relax \_relax \_opwarning{Unicode font was not loaded}\_fi \_ifnum \_unresolvedrefs>0 \_opwarning{Rerun to get references right}\_fi}

2.5 Allocators for TEX registers

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

Unlike in Plain\TeX, the mentioned allocators are not \outer. User can use \dimen0 to \dimen200 and similarly for \skip, \muskip, \box and \toks directly.

User can use \count20 to \count200 directly too. This is the same philosophy like in old plain\TeX, but the range of directly used registers is wider.

Inserts are allocated from 254 to 201 using \newinsert.

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

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

\def\newcountarray #1[\#2]{% \newcountarray \foo[100] \_global\_advance\_maicount by -\#2\_relax \_ifnum \_countalloc > \_maicount \errmessage{No room for a new array of \string\count}\% \else \_global\chardef#1=\_maicount \_fi }

\def\usecount #1[\#2]{% \usecount \foo[2] \_count\numexpr\#1+\#2\_relax }
The limits are set first.

Each allocation macro needs its own counter.

The common allocation macro \_allocator \langle sequence \rangle \langle primitive declarator \rangle is defined.

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

The \_newinsert macro is defined differently than others.
Other allocation macros \newattribute and \newcatcodetable have their counter allocated by the \newcount macro.

```
\newcount \attributealloc \attributealloc=0
\chardef \mais\attribute=\maicount
\def \newattribute #1{\allocator #1{attribute}\attributedef}
\newcount \catcodetablealloc \catcodetablealloc=10
\chardef \maicatcodetable=32767
\def \newcatcodetable #1{\allocator #1{catcodetable}\chardef}
\public \newattribute \newcatcodetable ;
```

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

```
\newcount \tmpnum \newcount \tmpnum
\newdimen \tmpdim \newdimen \tmpdim
```

A few registers are initialized like in plainTEX. Note that \z@skip from plainTEX is \zoskip here because we absolutely don’t support the \@ category dance. The \z@ and \p@ is not defined because we can write 0pt or 1pt which is more legible in source code. You can see plain-at.opm file.

```
\newdimen \maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
\newskip \hideskip \hideskip=-1000pt plus 1fill % negative but can grow
\newskip \centering \centering=0pt plus 1000pt minus 1000pt
\newbox \voidbox % permanently void box register
\public \maxdimen \hideskip \centering \zoskip \voidbox ;
```

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

### 2.6.1 Classical \newif

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

The macro \newifi enables to declare \_ifxxx and to use \_xxxtrue and \_xxxfalse. This means that it is usable for internal name space (_-prefixed macros).

```
\newif \newifi \newifiA \_string #1\relax#1
\def \newifi #1{\ea\newifiA \_string\if #1\relax#2[\%\sdef{#1true}{\let#2=\iftrue}\sdef{#1false}{\let#2=\iffalse}\let#2=\iffalse\]}
\public \newifi ;
```

### 2.6.2 Loops

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

There are public version \loop...\repeat and private version \_loop...\_repeat. You cannot mix both versions in one loop.
The \loop macro keeps its original plain TeX meaning. It is not expandable and nested \loops are possible only in a TeX group.

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

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

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

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

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

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

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

\begin{verbatim}
\issomething \langle params \rangle \iftrue \langle codeA \rangle \else \langle codeB \rangle \fi
\end{verbatim}

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

\texttt{\isempty \langle text \rangle} \texttt{\iftrue} is true if the \texttt{\langle text \rangle} is empty. This macro is expandable.

\texttt{\istoksempty \langle tokens variable \rangle} \texttt{\iftrue} is true if the \texttt{\langle tokens variable \rangle} is empty. It is expandable.

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

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

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

\texttt{\isinlist \langle text \rangle} \texttt{\iftrue} is true if the \texttt{\langle text \rangle} is included the macro body of the \texttt{\langle list \rangle}. The category code are relevant here. The macro is not expandable.

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

\begin{verbatim}
\isempty \{ \langle text \rangle \} \iftrue \else \langle codeB \rangle \fi
\end{verbatim}

\begin{verbatim}
\isequal \{ \langle textA \rangle \} \{ \langle textB \rangle \} \iftrue
\end{verbatim}

\begin{verbatim}
\ismacro \{ \langle text \rangle \} \iftrue
\end{verbatim}

\begin{verbatim}
\isdefined \{ \langle csname \rangle \} \iftrue
\end{verbatim}

\begin{verbatim}
\isinlist \{ \langle text \rangle \} \iftrue
\end{verbatim}

\begin{verbatim}
\isfile \{ \langle filename \rangle \} \iftrue
\end{verbatim}


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

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

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

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

\_public \_isfont ;

2.7 Setting parameters

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

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

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

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

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

2.7.1 Primitive registers

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

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

\_topskip=10pt % top edge of page-box to first baseline distance
\_splittopskip=10pt
\_normalbaselineskip=12pt
\_normallineskip=1pt
\_normallineskiplimit=0pt
\_interdisplaylinepenalty=100
\_interfootnotelinepenalty=100
\_normallineskip=1pt
\_jot=3pt
\_frenchspacing\{\_sfcode\=1000 \_sfcode\?=1000 \_sfcode\!=1000
\_sfcode\:=1000 \_sfcode\;=1000 \_sfcode\,=1000 \_sfcode\=1500 \_sfcode\:=1250 \_sfcode\rm
\_nonfrenchspacing\{\_sfcode\=3000 \_sfcode\?=3000 \_sfcode\!=3000
\_sfcode\:=1500 \_sfcode\;=1250 \_sfcode\,=1000 \_sfcode\=1000 \_sfcode\rm

2.7.2 Plain \TeX registers

Declared registers used in plain \TeX

\_frenchspacing\{\_sfcode\=1000 \_sfcode\?=1000 \_sfcode\!=1000
\_sfcode\:=1000 \_sfcode\;=1000 \_sfcode\,=1000 \_sfcode\=1500 \_sfcode\:=1250 \_sfcode\rm
\_frenchspacing\{\_sfcode\=3000 \_sfcode\?=3000 \_sfcode\!=3000
\_sfcode\:=1500 \_sfcode\;=1250 \_sfcode\,=1000 \_sfcode\=1000 \_sfcode\rm

2.7.3 Different settings than in plain \TeX

Default “baseline setting” is for 10 pt fonts (like in plain \TeX). But \texttt{\_topskip} and \texttt{\_splittopskip} are changed when first \texttt{\_topskip} sets the main values (default font size and default \texttt{\_baselineskip}).

\_topskip=10pt % top edge of page-box to first baseline distance
\_splittopskip=10pt
\_jot=3pt
\_normalbaselineskip=12pt
\_normallineskip=1pt
\_normallineskiplimit=0pt
\_interdisplaylinepenalty=100
\_interfootnotelinepenalty=100
\_frenchspacing\{\_sfcode\=1000 \_sfcode\?=1000 \_sfcode\!=1000
\_sfcode\:=1000 \_sfcode\;=1000 \_sfcode\,=1000 \_sfcode\=1500 \_sfcode\:=1250 \_sfcode\rm
\_nonfrenchspacing\{\_sfcode\=3000 \_sfcode\?=3000 \_sfcode\!=3000
\_sfcode\:=1500 \_sfcode\;=1250 \_sfcode\,=1000 \_sfcode\=1000 \_sfcode\rm

\_topskip=10pt % top edge of page-box to first baseline distance
\_splittopskip=10pt
\_jot=3pt
\_normalbaselineskip=12pt
\_normallineskip=1pt
\_normallineskiplimit=0pt
\_interdisplaylinepenalty=100
\_interfootnotelinepenalty=100
\_frenchspacing\{\_sfcode\=1000 \_sfcode\?=1000 \_sfcode\!=1000
\_sfcode\:=1000 \_sfcode\;=1000 \_sfcode\,=1000 \_sfcode\=1500 \_sfcode\:=1250 \_sfcode\rm
\_nonfrenchspacing\{\_sfcode\=3000 \_sfcode\?=3000 \_sfcode\!=3000
\_sfcode\:=1500 \_sfcode\;=1250 \_sfcode\,=1000 \_sfcode\=1000 \_sfcode\rm

The \texttt{\_nonfrenchspacing} is not set by default because the author of Op\TeX is living in the Europe. If you set \texttt{\_enlang} hyphenation patterns then \texttt{\_nonfrenchspacing} is set.
Different values than in plain \TeX{} have following primitive registers. We prohibit orphans, set more information for tracing boxes, set page origin to upper left corner of the paper (no at 1 in, 1 in coordinates) and set default page dimensions as A4, no letter.

\begin{verbatim}
\_normalbaselines \% baseline setting, 10 pt font size
\_emergencystretch=20pt \% we want to use third pass of aparagraph building algorithm
\_clubpenalty=10000 \% after first line of paragraph
\_widowpenalty=10000 \% before last line of paragraph
\_showboxbreadth=150 \% for tracing boxes
\_showboxdepth=7
\_errorcontextlines=15
\_tracinglostchars=2 \% missing character warnings on terminal too
\_outputmode=1 \% PDF output
\_pdfvorigin=0pt \% origin is exactly at left upper corner
\_pdfhorigin=0pt
\_hoffset=25mm \% margins are 2.5cm, no 1in
\_voffset=25mm
\_hsize=160mm \% 210mm (from A4 size) - 2*25mm (default margins)
\_vsize=244mm \% 297mm (from A4 size) - 2*25mm (default margins) - 3mm baseline correction
\_pagewidth=210 true mm
\_pageheight=297 true mm
\end{verbatim}

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

\begin{verbatim}
\_def\plaintexsetting{%
\_emergencystretch=0pt \% origin is exactly at left upper corner
\_clubpenalty=150
\_widowpenalty=150
\_pdfvorigin=1in
\_pdfhorigin=1in
\_hoffset=0pt
\_voffset=0pt
\_hsize=6.5in
\_vsize=8.9in
\_pagewidth=8.5 true in
\_pageheight=11 true in
\_nonfrenchspacing
}
\_public \plaintexsetting ;
\end{verbatim}

### 2.7.4 Op\TeX{} parameters

The main principle how to configure Op\TeX{} is not to use only parameters. A designer can copy macros from Op\TeX{} and re-define them as required. This is a reason why we don’t implement dozens of parameters, but we keep Op\TeX{} macros relatively simple. Example: do you want another design of section titles? Copy macros \texttt{\_printsec} and \texttt{\_printsecc} from \texttt{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 Op\TeX{} (OPmac uses macros for them). The reason of this difference: if user sets parameter by unprotected control sequence, an Op\TeX{} macro can read \texttt{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 \texttt{\picdir} tokens list can include a directory where image files (loaded by \texttt{\inspic}) are saved. Empty \texttt{\picdir} (default value) means that image files are in the current directory (or somewhere in the \TeX{} system where Lua\TeX{} is able to find them). If you set non-empty value to the \texttt{\picdir}, then it must end by / character, for example \texttt{\picdir=img/} means that there exists a directory img in your current directory and the image files are stored here.

\begin{verbatim}
\_newtoks\_picdir
\_public \picdir ;
\end{verbatim}

You can control the dimensions of included images by the parameters \texttt{\picwidth} (which is equivalent to \texttt{\picw}) and \texttt{\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.

\begin{verbatim}
\_newdimen\_picwidth \_picwidth=0pt \_let\picw=\_picwidth
\_newdimen\_picheight \_picheight=0pt
\_public \picwidth \picheight ;
\end{verbatim}

The \everytt is token list used in \begtt...\endtt environment and in the verbatim group opened by \verbatim 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 \everystint 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 \everyinttt=\Red for example if you want in-line verbatim in red color.

\begin{verbatim}
\_newtoks\_everytt \_everytt=
\_newtoks\_everyinttt \_everyinttt=
\_public \everytt \everyinttt ;
\end{verbatim}

The \ttline is used in \begtt...\endtt environment or in the code printed by \verbatim. 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 \verbatim lines shows the line numbers of inputted file. If \ttline<-1 then no line numbers are printed.

\begin{verbatim}
\_newcount\_ttline \_ttline=-1 % last line number in \begtt...\endtt
\_public \ttline ;
\end{verbatim}

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

\begin{verbatim}
\_newdimen\_ttindent \_ttindent=\_parindent % indentation in verbatim
\_newdimen\_ttshift
\_newdimen\_iindent \_iindent=\_parindent
\_public \ttindent \ttshift \iindent ;
\end{verbatim}

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

\begin{verbatim}
\_newdimen\_ttindent \_ttindent=\_parindent % indentation in verbatim
\_newdimen\_ttshift
\_newdimen\_iindent \_iindent=\_parindent
\_public \ttindent \ttshift \iindent ;
\_newcount \tabspaces \_tabspaces=3
\_public \tabspaces ;
\end{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 \begin{itemize}.
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.

\begin{Verbatim}
\newtoks\defaultitem \defaultitem={$\bullet$}\enspace
\newtoks\everyitem
\newtoks\everylist
\newskip\listskipamount \listskipamount=\medskipamount
\newcount\ilevel
\public\defaultitem \everyitem \everylist \listskipamount \ilevel;
\end{Verbatim}

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

\begin{Verbatim}
\newskip\titskip \titskip=40pt \relax \% \vglue above title printed by \tit
\public\titskip;
\end{Verbatim}

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.

\begin{Verbatim}
\newdimen\colsep \colsep=20pt \% space between columns
\public\colsep;
\end{Verbatim}

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

\begin{Verbatim}
\newtoks\everytocline
\public\everytocline;
\end{Verbatim}

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.

\begin{Verbatim}
\newtoks\bibtexhook
\public\bibtexhook;
\end{Verbatim}

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

\begin{Verbatim}
\newtoks\everyii
\public\everyii;
\end{Verbatim}

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

\begin{Verbatim}
\newdimen\mnotesize \mnotesize=20mm \% the width of the mnote paragraph
\newdimen\mnoteindent \mnoteindent=10pt \% distance between mnote and text
\public\mnotesize \mnoteindent;
\end{Verbatim}

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.
\texttt{\tabiteml} is applied before each item, \texttt{\tabitemr} is applied after each item of the table. \texttt{\tablinespace} is additional vertical space between horizontal rules and the lines of the table. \texttt{\hhkern} gives the space between horizontal lines if they are doubled and \texttt{\vvkern} gives the space between such vertical lines. \texttt{\tabskipl} is \texttt{\tabskip} used before first column, \texttt{\tabskipr} is \texttt{\tabskip} used after the last column. \texttt{\tsize} is virtual unit of the width of paragraph-like table items when \texttt{\table\size{⟨size⟩}} is used.

```
\newtoks\everytable \newtoks\thisstable
\newdimen\tablinespace \newdimen\vvkern \newdimen\hhkern \newdimen\tsize
\newskip\tabskipl \newskip\tabskipr
\_newdimen\_tablinespace=2pt \_newdimen\_vvkern=1pt \_newdimen\_hhkern=1pt
\_newskip\_tabskipl=0pt\_relax \_newskip\_tabskipr=0pt\_relax
\_public\_everytable \_thistable \_tabiteml \_tabitemr \_tabstrut \_tablinespace \_vvkern \_hhkern \_tsize \_tabskipl \_tabskipr;
```

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

```
\_newtoks\_eqlines \_newtoks\_eqstyle
\_newdimen\_eqspace=20pt
\_public\_eqlines \_eqstyle \_eqspace;
```

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

```
\_newtoks\_lmfil
\_public\_lmfil;
```

The output routine uses token list \texttt{\headline} and \texttt{\footline} in the same sense as in plain TreX. If they are non-empty then \texttt{\hfil} or \texttt{\hss} must be here because they are used inside \texttt{\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 \texttt{\fontdef\rmfixed{\rm}}, \texttt{\fontdef\itfixed{\it}}. Then use them in headline and footline:

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

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

```
\newdimen\_headlinedist \_headlinedist=14pt
\newdimen\_footlinedist \_footlinedist=24pt
\_public\_headlinedist \_footlinedist;
```

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

```
\_newdimen\_headlinedist\_headlinedist=14pt
\_newdimen\_footlinedist\_footlinedist=24pt
\_public\_headlinedist\_footlinedist;
The \texttt{\nextpages} tokens list can include settings which will be used at next pages. It is processed at the end of output routine with \texttt{globaldefs=1} prefix. The \texttt{\nextpages} is reset to empty after processing. Example of usage:

\begin{verbatim}
\headline={} \nextpages={\headline={\fixedrm \firstmark \hfil}}
\end{verbatim}

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

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

\begin{verbatim}
\pgbackground={\Yellow \hrule height 0pt depth\pdfpageheight width\pdfpagewidth}
\end{verbatim}

The parameters used in \texttt{\inovals} and \texttt{\incircles} macros. The default values (documented in user manual) are set in the macros. The user can re-set these values using tokens \texttt{\ovalparams}, \texttt{\circleparams}.

\begin{verbatim}
\ovalparams\circleparams\roundness
\end{verbatim}

\subsection{More Op\TeX{} macros}

The second bundle of Op\TeX{} macros is here.

We define \texttt{\opinput \{file name\}} macro which does \texttt{\input \{file name\}} but the catcodes are set to normal catcodes (like Op\TeX{} initializes them) and the catcodes setting are returned back to the current values when the file is read. You can use \texttt{\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 \texttt{\optexcatcodes} catcode table and \texttt{\plaintexcatcodes}. They save the commonly used catcode tables. Note that \texttt{\catcodetable} is a part of Lua\TeX{} extension. The catcodetable stack is implemented by Op\TeX{} macros. The \texttt{\setctable \{catcode table\}} pushes current catcode table to the stack and activates catcodes from the \texttt{\{catcode table\}}. The \texttt{\restorectable} returns to the saved catcodes from the catcode table stack. So, the \texttt{\opinput} macro can be implemented simply:

\begin{verbatim}
\def\opinput #1\{\setctable \optexcatcodes \\input \{#1\}\relax \restorectable}
\end{verbatim}

The implementation of the catcodetable stack follows.

The current catcodes are managed in the \texttt{\catcodetable0}. If the \texttt{\setctable} is used first (or at the outer level of the stack), then the \texttt{\catcodetable0} is pushed to the stack and the current table is
re-set to the given \(\langle\text{catcode table}\rangle\). 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 \optecatodes table and sets it to the main catcode table 0.

The \load \(\langle\text{filename-list}\rangle\) loads files specified in comma separated \(\langle\text{filename-list}\rangle\). 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: \(\langle\text{filename}\rangle\) 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 \(\langle\text{opt default}\rangle\) \(\langle\text{params}\rangle\)\(\langle\text{replacement text}\rangle\) defines the \macro with the optional parameter followed by normal parameters declared in \(\langle\text{params}\rangle\). The optional parameter must be used as the first parameter in brackets [..]. If it isn’t used then \(\langle\text{opt default}\rangle\) is taken into account. The \(\langle\text{replacement text}\rangle\) can use \the\opt because optional parameter is saved to the \opt tokens register. Note the difference from \TeX concept where the optional parameter is in #1. \OpTeX uses #1 as the first normal parameter (if declared).

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

The declarator \olddef\macro #1\{\langle\text{replacement text}\rangle\} 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 \(\langle\text{replacement text}\rangle\). 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 \skiptoeol macro reads the text to the end of the current line and ignores it.

\macro{\skiptoeol}langle\text{to end of line}\rangle scans the \langle\text{to end of line}\rangle in verbatim mode and runs the \macro{\langle\text{to end of line}\rangle}. The \macro{\langle\text{to end of line}\rangle} can be defined \def\macro#1{...\skiptextokens{#1}...}.

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

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

The \replstring macro\{<textA>\}\{<textB>\} 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\langle textA \rangle \ldots\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\text{something}\rangle or \catcode`\langle\text{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 textwithoutspaces \rangle. The \_ea\ignorept\the\langle dimen \rangle expands to a decimal number \the\langle dimen \rangle but without pt unit.

The \_ea\ignoreit\langle\token\rangle just ignores the \langle\token\rangle.
You can use expandable \texttt{\bp{⟨dimen⟩}} converor from \TeX \texttt{⟨dimen⟩} (or from an expression accepted by \texttt{\dimexpr} primitive) to a decimal value in big points (used as natural unit in the PDF format). So, you can write, for example:

\texttt{\pdfliteral{q \_bp{.3\hsize-2mm} \_bp{2mm} m 0 \_bp{-4mm} l S Q}}

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

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

The pair \texttt{\doc ... \cod} is used for documenting macros and to printing the technical documentation of the Op\TeX. The syntax is:

\texttt{\doc <ignored text> <documentation> \cod <ignored text>}

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

Plain \TeX macros

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

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

\texttt{\addto \dospecials{\do\_}}

The shortcuts \texttt{\chardef\@one} is not defined in Op\TeX. Use normal numbers instead of such obscurities.

Plain \TeX basic macros and control sequences. \texttt{\endgraf}, \texttt{\endline}. The \texttt{\char\^L} is not defined in Op\TeX because it is obsolete.
Plain TeX classical \texttt{\obeylines} and \texttt{\obeyspaces}.

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

Penalties macros: \texttt{\break}, \texttt{\nobreak}, \texttt{\allowbreak}, \texttt{\filbreak}, \texttt{\goodbreak}, \texttt{\eject}, \texttt{\supereject}, \texttt{\dosupereject}, \texttt{\removelastskip}, \texttt{\smallskip}, \texttt{\medskip}, \texttt{\bigskip}.
The `\strutbox` is declared as 10pt size dependent (like in plain TeX), but the macro \_setbaselineskip (from `fonts-opmac.opm`) redefines it.

Alignment. `\hidewidth` `\ialign` `\multispan`.

Tabbing macros are omitted because they are obsolete.

Indentation and others. `\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.
\def\_ifmmode \kern.06em \vbox{\hrule width 3em}\else \fi \% obsolete
\def\hbox{(_}
\def\dag{\_errmessage{\_usedirectly †}}
\def\ddag{\_errmessage{\_usedirectly ‡}}
% \def\copyright{\_errmessage{\_usedirectly ©}}
\def\copyright{©} % << example, what to do
% \def\Orb{\_mathhexbox20D} % obsolete (part of Copyright)
% \def\P{\_mathhexbox27B} % obsolete
\def \_usedirectly #1{Load Unicoded font by \string\fontfam\space and use directly #1}
\def \_mathhexbox #1#2#3{\_leavevmode \_hbox{$\_math \_mathchar"#1#2#3$}}
\public \mathhexbox ;

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

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

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

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

\def \_oldaccents {\_def \_tgrave={\_accent\_tgrave #1}\_def \_tacute={\_accent\_tacute #1}\_def \_circumflex={\_accent\_circumflex #1}\_def \_ttilde={\_accent\_ttilde #1}\_def \_dieresis={\_accent\_dieresis #1}\_def \_ring={\_accent\_ring #1}}
\public \oldaccents ;

% ec-lmr encoding (will be changed after \fontfam macro):
\chardef\_tgrave=0
\chardef\_circumflex=2
\chardef\_ttilde=3
\chardef\_dieresis=4
\chardef\_hungarumlaut=5
\chardef\_ring=6
\chardef\_macron=9
\chardef\_dotaccent=10
\chardef\_cedilla=11
\chardef\_uniaccents (%) accents with Unicode
\chardef\_tgrave=0060
\chardef\_tacute=00B4
\chardef\_circumflex=00CF
\chardef\_ttilde=02DC
\chardef\_dieresis=00A8

\public \uniaccents ;
The plain \TeX\ macros \texttt{\hrulefill, \dotfill, \rightarrowfill, \leftarrowfill, \downbracefill, \upbracefill}. The last four are used in non-Unicode variants of \texttt{\overrightarrow, \overleftarrow, \overbrace and \underbrace} macros, see section 2.14.

\begin{verbatim}
305 \def \hrulefill {\leaders \hrule \hfill}
306 \def \dotfill {\cleaders \hbox{$\math \mkern1.5mu.$} \hfill}
307 \def \rightarrowfill {$\math \smash-\mkern-7mu%\cleaders \hbox{$\mkern-2mu \smash-\mkern-2mu$} \hfill \mkern-7mu \mathord \rightarrow$}
308 \def \leftarrowfill {$\math \mathord \leftarrow \mkern-7mu%\cleaders \hbox{$\mkern-2mu \smash-\mkern-2mu$} \hfill \mkern-7mu \smash-$}
309 \mathchardef \braceld="37A \mathchardef \bracerd="37B
310 \mathchardef \bracelu="37C \mathchardef \braceru="37D
311 \def \downbracefill {$\math \setbox0=\hbox{$\braceld$} \braceld \leaders \vrule height \ht0 depth0pt \hfill \braceru \bracelu \leaders \vrule height \ht0 depth0pt \hfill \bracerd$}
312 \def \upbracefill {$\math \setbox0=\hbox{$\braceld$} \bracelu \leaders \vrule height \ht0 depth0pt \hfill \bracerd \braceld \leaders \vrule height \ht0 depth0pt \hfill \braceru$}
313
314 \public \hrulefill \dotfill \rightarrowfill \leftarrowfill \downbracefill \upbracefill ;
\end{verbatim}

The last part of plain \TeX\ macros: \texttt{\magnification, \bye}. Note that math macros are defined in the \texttt{math-macros.opm} file (section 2.14).

\begin{verbatim}
320 \def \magnification {\afterassignment \magA \count255 }
321 \def \magA {\mag=\count255 \truedimen \ht\vsize \truedimen \vsizerem}
322 \% only for backward compatibility, but \texttt{\margins} macro is preferred.
323 \public \magnification ;
324 \def \showhyphens #1{\setbox0=\vbox{\parfillskip=0pt \hspace{\maxdimen} \tenrm \tenem \tenbf \tenit \tenbi \tentt}}
\end{verbatim}

\texttt{2.10 Preloaded fonts for text mode}

Format in lu\TeX\ can download only non-Unicode fonts. Latin Modern EC is loaded here. These fonts are totally unusable in Lua\TeX\ 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 \texttt{\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 \texttt{\everyjob}. But why to load a set of fonts at 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 \texttt{\tenrm, \tenbf, \tenit, \tenbi, \tentt} are declared as \texttt{\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.
2.11 Scaling fonts in text mode (low-level macros)

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

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

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

The \texttt{\resizethefont} resizes the current font to the size given by previous \texttt{\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 \texttt{\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 \texttt{\resizethefont} is used.

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

\begin{verbatim}
\def\smaller{\setfontsize{mag.9}\resizethefont}
Text \smaller text \smaller text \smaller text.
\end{verbatim}

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

\begin{verbatim}
\font\tencomfortaa=Comfortaa-Regular-T1 at10pt
\def\comfortaa{\tencomfortaa\resizethefont}
\comfortaa Here is 10 pt text
\setfontsize{at12pt}
\comfortaa Here is 12 pt text
\end{verbatim}

The example above uses the 8bit tfm font. You can use Unicode font too, of course. The \texttt{\fontfam} macro initializes the extended \texttt{\font} primitive features for Lua\TeX\ (see section 2.12.10). If you didn't use this command, you must to initialize these features by the \texttt{\initunifonts} command explicitly, for example:

\begin{verbatim}
\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
\end{verbatim}
2.11.1 The \fontdef declarator

You can declare \(\langle\text{newfont}\rangle\) by the \fontdef command.

\[
\text{\fontdef \langle\text{newfont}\rangle \{(font\ modifiers) \langle\text{variant-selector}\rangle}\}
\]

e.g.: \fontdef \bigfont \{\setfontsize{at15pt}\bf}\}

This command runs \(\langle\text{font modifiers}\rangle \langle\text{variant-selector}\rangle\) in a group and sets the resulting current font as \(\langle\text{newfont}\rangle\).

The resulting \(\langle\text{newfont}\rangle\) 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 2.12.

2.11.2 The \fontlet declarator

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

\[
\text{\fontlet \langle\text{newfont}\rangle = \langle\text{fontswitch}\rangle \langle\text{sizespec}\rangle}\}
\]

e.g.: \fontlet \bigfont \= \_tenbf at15pt

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

Summary: you can declare font switches:

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

2.11.3 Optical sizes

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

There is \defaultoptsize=10pt by default.

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

2.11.4 Implementation notes

The \setfontsize \(\langle\text{sizespec}\rangle\) saves the \(\langle\text{sizespec}\rangle\) to the \_sizespec macro. The \optsize value is calculated from the \(\langle\text{sizespec}\rangle\). If the \(\langle\text{sizespec}\rangle\) is in the mag\(\langle\text{number}\rangle\) format then the contents of the \_sizespec macro is re-calculated to the at\(\langle\dimen\rangle\) format using previous \optsize value.

```latex
\text{\_newdimen \optsize \optsize=10pt}
\text{\_newdimen \defaultoptsize \defaultoptsize=10pt}
\text{\_newdimen \lastmagsize}
\text{\_def \_setfontsize \#1{\_edef \_sizespec{\_isnextchar a{\_setoptsizeA}{\_isnextchar m{\_setoptsizeC}{\_setoptsizeB}}}}}
```
\_resizefont \{(variant-name)\}\{\font switch\}, for example \resizefont{bf}\_tenbf resizes the font given by the variant. The variant XX have its font switch \_tenXX. The \_doresizefont\fontswitch is used. It works in TFM mode (\_doresizetfmfont) or OTF mode (\_doresizeunifont). In both modes, it does
\_font \_tenXX = \{\fontname\}\sizespec

The \{\fontname\} is generated by the \_fontname TP\_X primitive where \_rfskipat removes the \texttt{at}\{\texttt{dimen}\} part of the \_fontname output. The \{\fontname\} is generated differently in OTF mode, see \_doresizeunifont macro.

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

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

\_fontlet \{\font switch A\}\{\font switch B\}\{\size spec\} does
\_font \{\font switch A\} = \{\fontname\}\{\size spec\}

The \{\fontname\} is extracted using the primitive command \_fontname. The \_size spec is set by \_newcurrfontsize \{\size spec\} set 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 TP\_X when \_font primitive is started. \_resizethedef is implemented by \_newcurrfontsize using data from the \_sizespec macro.
The variant selector is defined by \protected\def\XX{\_tryloadXX \_tenXX} The \_tryloadXX can be in \_relax state if no font modifiers were declared. But normally it does \_resizefont{XX}\tenXX. This meaning is activated by the \_reloading macro.

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

The implementation detail: The \_⟨font id⟩reg is defined as the ⟨optical size data⟩ and all control sequences \_⟨fontname⟩reg from this data line has the same meaning because of the \_reversetfm macro. The \_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:

<table>
<thead>
<tr>
<th>Font</th>
<th>Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmr</td>
<td>5.5</td>
</tr>
<tr>
<td>lmr8</td>
<td>8.5</td>
</tr>
<tr>
<td>lmr9</td>
<td>9.5</td>
</tr>
<tr>
<td>lmr10</td>
<td>11.1</td>
</tr>
<tr>
<td>lmr12</td>
<td>*</td>
</tr>
<tr>
<td>lmbx</td>
<td>5.5</td>
</tr>
<tr>
<td>lmbx6</td>
<td>6.5</td>
</tr>
<tr>
<td>lmbx7</td>
<td>7.5</td>
</tr>
<tr>
<td>lmbx8</td>
<td>8.5</td>
</tr>
<tr>
<td>lmbx9</td>
<td>9.5</td>
</tr>
<tr>
<td>lmbx10</td>
<td>11.1</td>
</tr>
<tr>
<td>lmbx12</td>
<td>*</td>
</tr>
<tr>
<td>lmri</td>
<td>7.5</td>
</tr>
<tr>
<td>lmri8</td>
<td>8.5</td>
</tr>
<tr>
<td>lmri9</td>
<td>9.5</td>
</tr>
<tr>
<td>lmri10</td>
<td>11.1</td>
</tr>
<tr>
<td>lmri12</td>
<td>*</td>
</tr>
<tr>
<td>lmtt</td>
<td>8.5</td>
</tr>
<tr>
<td>lmtt9</td>
<td>9.5</td>
</tr>
<tr>
<td>lmtt10</td>
<td>11.1</td>
</tr>
<tr>
<td>lmtt12</td>
<td>*</td>
</tr>
</tbody>
</table>

2.12 The Font Selection System

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

2.12.1 Terminology

We distinguish between

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

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

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

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

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

2.12.2 Font families, selecting fonts

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

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

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

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

\{(family selector) \{font modifiers\} \{variant selector\}\}
For example:

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

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

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

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

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

2.12.3 Math Fonts

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

If you want to use your specially selected Unicode-math font then use \loadmath \{(\{font_file\}\)} or \loadmath \{(\{font_name\}\)} before first \fontfam is used.

2.12.4 Declaring font commands

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

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

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

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

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

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

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

The \{(\new \{\{font modifiers\}\} \{\{variant selector\}\} should be used in the same sense as \rm, \bf etc. It can be used as the final command in the \fontdef or \famvardef declarators too. When the \{(\new \{\{font modifiers\}\} \{\{variant selector\}\} is used in normal text then it does following steps: pushes current font context to a stack, modifies font context by declared \{\{font modifiers\}\}, runs following \{\{variant selector\}\}. It selects a font. Then pops the stack. The font context have its original values but new font is selected.
The \famvardef creates the \(\textit{new variant selector}\) family dependent. When the selector is used in another family than it is defined then warning is printed on the terminal “\textit{⟨var selector⟩} is undeclared in current family” and nothing happens. But you can declare the same variant selector by \famvardef macro in the context of new family. Then the same command will be do different work depending on the current font family.

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

\begin{verbatim}
\famvardef \mr \{\medium\rm \}
\famvardef \mi \{\medium\it\}
\end{verbatim}

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

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

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

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

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

### 2.12.5 Modifying font features

Each OTF font provides “font features”. You can list these font features by \texttt{otfinfo -f font.otf}. For example LinLibertine fonts provide \textit{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 \texttt{\setff \{⟨feature⟩\}} acts like family independent font modifier and prepares a new \(\langle\textit{feature}\rangle\). You must use a variant selector in order to reinitialize the font with the new font feature. For example \texttt{\setff{+frac}\rm} or \texttt{\setff{+frac}\currvar}.

You can declare a new variant selector too:

\begin{verbatim}
\fontfam[\textit{LinLibertine}]
\famvardef \fraclig \{\setff{+frac}\currvar\}
\end{verbatim}

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

### 2.12.6 Special font modifiers

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

\begin{verbatim}
\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
\end{verbatim}

The \setfontsize command is described in the section 2.11. The \setff command was described in previous subsection.
\setfontcolor \{\{color\}\} specifies the color and the opacity of the text. The \{color\} parameter should be in hexadecimal format of four bytes \{red\}\{green\}\{blue\}\{opacity\}, for example FF0080FF means full red, zero green, half blue and full opacity. You can use names red, green, blue, yellow, cyan, magenta, white, grey, lgrey (without backslash) instead of the hexadecimal specification. The empty parameter \{color\} means default black color.

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

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

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

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

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

2.12.7 How to create the font family file

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

\_famdecl [Heros] \Heros {TeX Gyre Heros fonts based on Helvetica}
{\caps \cond} {\rm \bf \it \bi} {FiraMath}
{[texgyreheros-regular]}
{\def\fontnamegen{[texgyreheros-\condV-\_currV]\:\_capsV\_fontfeatures}}
\_wlog{\_detokenize{\
Modifiers:^^J
\caps ...... caps & small caps^^J
\cond ...... condensed variants^^J}}
\_moddef \resetmod {\fsetV caps=\{} \_fvars regular bold italic bolditalic }
\_moddef \caps {\fsetV caps=s\mcp;onum; }\n\_moddef \nocaps {\fsetV caps=\} }
\_moddef \cond {\fsetV cond=cn }
\_moddef \nocond {\fsetV cond=\} }
\_initfontfamily % new font family must be initialized
\_loadmath {[FiraMath-Regular]}

If you want to write such font file, you need to keep following rules.
• Use the \_famdecl command first. It has the following syntax:
\_famdecl \{(Name of family)\} \{(Familyselector) \{(comments)\} \{(modifiers)\} \{(variant selectors)\} \{(comments about math fonts)\} \{(font-for-testing)\} \{\def\fontnamegen{\{font name or font file name generated\}}\}}

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

- You can use \_wlog\detokenize{} to write 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.ofm file for more information about this special feature.
- Declare font modifiers using \_moddef if they are present. The \resetmod must be declared in each font family.
- Check if all your declared modifiers does not produce any space in horizontal mode. For example check: \X\caps Y, the letters XY must be printed without any space.
- Optionally, declare new variants by the \famvardef macro.
- Run \initfontfamily in order to start the family.
- If math font should be loaded, use \loadmath{\math font}.

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

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

Note that the extended \font syntax \font\{\selector\} \{\fontname\:\{\fontfeatures\}\} \{\sizespec\} or \font\{\selector\} \{\fontname\:\{\fontfeatures\}\} \{\sizespec\} is expected here.

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

\_famdecl
\{\_fontnamegen\{xx-\_currV\}\}

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

\_famdecl ...
\{\_def \_fontnamegen\{xx-\_currV\}\}

The following auxiliary macros are used here:

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

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

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

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

is processed.

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

\fsetV \{word\}=(\value-a),\{word\}=(\value-b) ...etc. terminated by a space

with obvious result \_\{word\}\V(\{value-a\}) \_\{word\}\V(\{value-b\}) etc.

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

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

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

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

The \moddef macro does more things than simple \_def:
• The modifier macros are defined as \_\texttt{protected}.
• The modifier macros are defined as family-dependent.

The \texttt{\famvardef} macro has the same features.

The \texttt{\langle Familyselector \rangle} is defined by the \texttt{\famdecl} macro as:
\begin{verbatim}
\protected\def\langle Familyselector \rangle {%
  \def\_currfamily {\langle Familyselector \rangle}%
  \def\_fontnamegen {\langle font name or font file name generated \rangle}%
  \resetmod
\end{verbatim}

The font context consists from
• Family context, i.e. \_\texttt{currfamily} and \_\texttt{fontnamegen} values saved by the \texttt{\langle Familyselector \rangle},
• \_\texttt{ sizespec} value saved by the \texttt{\setfontsize} macro,
• whatever what influences the expansion of the \_\texttt{fontnamegen} macro, they are typically macros \_\texttt{(key)V} saved by the font modifers.

The \texttt{\initfontfamily} must be run after modifers decaration. It sets \_\texttt{let resetmod=\resetmod} and runs the \texttt{\langle Familyselector \rangle}. Finally, it runs \_\texttt{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 \texttt{\langle Familyselector \rangle} only as public, i.e. without \_ prefix. We assume that if user re-defines them then he/she needs not them, so we have no problems.

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

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

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

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

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

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

You can use \_\texttt{optname} macro when \_\texttt{fontnamegen} in expanded. This macro is fully expandable and its input is \texttt{\langle internal-template \rangle} and its output is a part of the font file name \texttt{\langle size-dependent-template \rangle} with respect to given optical size.

You can declare a collection of \texttt{\langle size-dependent-template \rangle}s for one given \texttt{\langle internal-template \rangle} by the \_\texttt{regoptsizes} macro. The syntax is shown for one real case:
\begin{verbatim}
\_regoptsizes lmr.r lmroman?-regular
  5 <5.5 6 <6.5 7 <7.5 8 <8.5 9 <9.5 10 <11.1 12 <15 17 <*
\end{verbatim}

In general:
\begin{verbatim}
\_regoptsizes \langle internal-template \rangle \langle general-output-template \rangle \langle resizing-data \rangle
\end{verbatim}

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

If \_\texttt{optname} gets an argument which is not registered \texttt{\langle internal-template \rangle} then it expands to \_\texttt{failedoptname} which typically ends to error message about missing font. You can redefine \_\texttt{failedoptname} macro to some existing font if you find it useful.
We are using a special macro \_LMregfont in f-lmfonts.opm. It sets the file names to lowercase and enables to use a shortcasts instead real \resizing-data. There are shortcats \_regoptFS, \_regoptT, etc. here. The collection of \internal-templates are declared, each of them covers a collection of real file names.

The \optfontalias \{new-template\} \{\internal-template\} declares \new-template with the same meaning as previously declared \internal-template.

The \optname macro can be used even if no optical sizes are provided by a font family. Suppose that font file names are much more chaotic (because artists are very creative people), so you need to declare more systematic \internal-templates and do an alias from each \internal-template to \real-font-name.

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}

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

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

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

The information from the user’s fams-local.opm file has precedence. For example fams-ini.opm declares aliases Times\rightarrow 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 \_famifo, \_famalias and \_famtext. See the example from fams-ini.tex:
The \_faminfo command has the syntax:
\_faminfo \[⟨Family Name⟩\] {⟨comments⟩} {⟨file-name⟩}
{⟨mod-plus-vars⟩}

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

2.12.10 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 built-in font renderer from Luaotfload. The harfbuzz library gives more features for rendering indic and arabic scripts. But it is not used as default, you need to specify mode=harf in 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.

2.12.11 Implementation of the Font Selection System

\_def \_initunifonts {%
\_directlua{%
require('luaotfload-main')
print_bak = print
print = function () end
luaotfload.main()
print = print_bak % "print hack" until luaotfload will be corrected
}%
\_gdef \_rfskipatX ##1" ##2\_relax{##1}%
\_global \_let \_doresizefont=\_doresizeunifont
\_global \_let \_tryloadtt {\_fontdef \_tentt{\_def \_fontnamegen{[lmmono10-regular]}\_rm}}%
\_global \_let \_initunifonts=\_relax \% we need not to do this work twice
}%
\_gdef \_initunifonts ()
\_directlua{%
require('luaotfload-main')
print_bak = print
print = function () end
luaotfload.main()
print = print_bak % "print hack" until luaotfload will be corrected
}%
\_gdef \_rfskipatX ##1" ##2\_relax{##1}%
\_global \_let \_doresizefont=\_doresizeunifont
\_global \_let \_tryloadtt {\_fontdef \_tentt{\_def \_fontnamegen{[lmmono10-regular]}\_rm}}%
\_global \_let \_initunifonts=\_relax \% we need not to do this work twice
\famdecl \[\{\text{Family Name}\}\] \{\{\text{command}\}\} \{\{\text{modifiers}\}\} \{\{\text{variants}\}\} \{\{\text{math}\}\} \{\text{font for testing}\}\} \{\\text{def} \{_\text{fontnamegen}\}{\{\text{data}\}}\} \text{runs} \{\_\text{initfontfamily}\}, \text{then checks if} \{\_\text{Famselector}\} \text{is defined. If it is true, then closes the file by} \{\text{endinput}\}. \text{Else it defines} \{\_\text{Famselector}\} \text{and saves it to the} \{\_\text{mainfamcommand}\} \text{macro because the} \{\_\text{initfontfamily}\} \text{needs it. The} \{\_\text{currentfamily}\} \text{is set to the} \{\_\text{Famselector}\} \text{because the following} \{\text{moddef}\} \text{commands need to be in the right font family context. The} \{\_\text{currentfamily}\} \text{is set to the} \{\_\text{Famselector}\} \text{by the} \{\_\text{Famselector}\} \text{too, because} \{\_\text{Famselector}\} \text{must set the right family context. The font family context is given by the current} \{\_\text{currfamily}\} \text{value and by the actual meaning of the} \{\_\text{fontnamegen}\} \text{macro.}

\famdecl \{\_\text{fvars}\} \{\_\text{rm-template}\} \{\text{bf-template}\} \{\text{it-template}\} \{\text{bi-template}\} \text{saves data for usage by the} \{\_\text{currv}\} \text{macro. If a template is only dot then previous template is used (it can be used if the font family doesn't dispose with all standard variants).} \_\text{fvars} \text{ expands to a template declared by} \_\text{fvars} \text{depending on the} \{\_\text{variant name}\}. \text{Usable only of standard four variants. Next variants can be declared by the} \_\text{famvardef}\} \text{macro.} \_\text{fset}\{\_\text{key}\}={\text{value}} \ldots,\{\_\text{key}\}={\text{value}} \text{expands to} \_\text{def}\{\_\text{key}\}={\text{value}} \text{in the loop.} \_\text{onlyif}\{\_\text{key}\}={\text{value-a}}, \{\_\text{value-b}\} \ldots,\{\_\text{value-z}\}: \{\_\text{what}\} \text{runs} \{\_\text{what}\} \text{only if the} \{\_\text{key}\}={\text{value}} \text{is defined as} \{\_\text{what}\} \text{or} \{\_\text{what}\} \text{or} ... \text{or} \{\_\text{what}\}.\text{...}
The \moddef {\langle modifier\rangle} {\langle data\rangle} simply speaking does \def{\langle modifier\rangle} {\langle data\rangle}, but we need to respect the family context. In fact, \protected\def{\langle f\rangle}{\langle current family\rangle} {\langle modifier\rangle} {\langle data\rangle} is performed and the \langle modifier\rangle is defined as \famdepend\langle modifier\rangle {\langle f\rangle} {\langle current family\rangle} {\langle modifier\rangle}. It expands to \langle f\rangle {\langle current family\rangle} {\langle modifier\rangle} value if it is defined or it prints warning. When the \langle current family\rangle value is changed then we can declare the same \langle modifier\rangle with different meaning.

When user declare a prefixed variant of the \langle modifier\rangle then unprefixed modifier name is used in internal macros, this is reason why we are using the \remfirststunderscore\langle temp\rangle (where \langle temp\rangle expands to \langle something\rangle) or to \langle something\rangle. The \remfirststunderscore redefine \langle temp\rangle in the way that it expands only to \langle something\rangle without the first _. 

The \famdef {\langle XX\rangle} {\langle data\rangle} uses analogical trick like \moddef with the \famdepend macro. The auxiliary \famdefA {\langle XX\rangle} {\langle ten\rangle} {\langle tryload\rangle} is used. It does:

- \protected\def{\langle XX\rangle} {\langle f\rangle} {\langle current family\rangle} {\langle modifier\rangle} {\langle data\rangle},
- \def{\langle f\rangle} {\langle current family\rangle} {\langle XX\rangle} {\langle ten\rangle} {\langle tryload\rangle} keeps family dependent definition,
- \def{\langle tryload\rangle} {\langle fontdef\rangle} {\langle ten\rangle} {\langle data\rangle} loads actually the font \langle ten\rangle,
- \def{\langle current var\rangle} {\langle ten\rangle} {\langle XX\rangle} in ordef to the \currentvar macro work correctly.

You cannot re-declare private standard variant selector \string\langle #1\rangle.

You cannot re-declare private standard variant selector \string\langle #1\rangle.

\protected\def{\langle XX\rangle} {\langle f\rangle} {\langle current family\rangle} {\langle modifier\rangle} {\langle data\rangle},
\def{\langle f\rangle} {\langle current family\rangle} {\langle XX\rangle} {\langle ten\rangle} {\langle tryload\rangle} keeps family dependent definition,
The \fontfam \{(Font Family)\} does:

- Convert its parameter to lower case and without spaces, e.g. \fontfam\.
- If the file f-\fontfam.opm exists read it and finish.
- Try to load user defined \fams-local.opm.
- If the \fontfam 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 {} }}%
\else
  \_tryloadfamslocal
  \_edef \_famfile{\_trycs{\_famf: \_famname}{}{}}%
  \_ifx \_famfile \_empty \_listfamnames
  \_else \_opinput {\_famfile.opm}%
  \fi \fi%
}
\_def \_tryloadfamslocal{%
  \_isfile {fams-local.opm} \_iftrue \_opinput {fams-local.opm} \_fi
  \_let \_tryloadfamslocal=\_relax % need not to load fams-local.opm twice
}
\_def \_listfamnames {%
  \wterm{===== List of font families ======}
  \begingroup
  \_let \_famtext=\_wterm
  \_def \_faminfo [##1]##2##3##4{%
    \_wterm{ \_space \_noexpand\fontfam [##1] -- ##2}%
    \_let \_famalias=\_famaliasA}
  \_opinput {fams-ini.opm}
  \_isfile {fams-local.opm} \_iftrue \_opinput {fams-local.opm} \_fi
  \_message{^^J}
  \_endgroup
}
\_def \_famaliasA\_message{ \_space \_space \_space \_space -- alias:}
\_def \_famalias[#1]\_message{[#1]}\_famalias
}
\_public \fontfam ;
```

When the \fams-ini.opm or \fams-local.opm files are read then we need to save only a mapping from family names or alias names to the font family file names. All other information is ignored in this case. But if these files are read by the \_listfamnames macro or when printing a catalog then more information is used and printed.

\_famtext does nothing or prints the text on the terminal.

\_faminfo \{(Family Name) \{(comments) \{(file-name) \{(mod-plus-vars)\} does
\_def \_famf:\{familyname\} \{\file-name\} or prints information on the terminal.
\_famalias \{(Family Alias)\ does \_def \_famf:\{familyalias\} \{\file-name\} where \file-name is stored from the previous \_faminfo command. Or prints information on the terminal.
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 `\catalogexclulde` of the catalog are declared here.

```
\newtoks \catalogsample
\newtoks \catalogmathsample
\newtoks \catalogonly
\newtoks \catalogexclulde
\catalogsample={(ABCĎabcd Qťy fi f1 Ćějčů Ćějčů Ľň 0123456789)}
\public \catalogonly \catalogexclulde \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.

```
\def \_defaultfontfeatures {+tlig;}
\def \_defaultfontfeatures {+tlig;}
\edef \_fontfeatures{\_defaultfontfeatures \_setff {} % default font features: +tlig; \_setff {} % default font features: +tlig;}
```

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

```
\def \_savedfontcolor{}
\def \_savedletterspace{}
\def \_savedws{}
\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\_removefeature{+Ws}\_if^#1^\_def \_setwsp##1{}\_removefeature{+Ws}\_else\_def \_setwsp\_setwspA{#1}\_setff{+Ws}\_fi
\reloading
\def \_setwsp #1{}
```

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

```
\def \_savedfontcolor{}
\def \_savedletterspace{}
\def \_savedws{}
\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\_removefeature{+Ws}\_if^#1^\_def \_setwsp##1{}\_removefeature{+Ws}\_else\_def \_setwsp\_setwspA{#1}\_setff{+Ws}\_fi
\reloading
\def \_setwsp #1{}
```

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.

```
\def \_defaultfontfeatures {+tlig;}
\def \_defaultfontfeatures {+tlig;}
\edef \_fontfeatures{\_defaultfontfeatures \_setff {} % default font features: +tlig; \_setff {} % default font features: +tlig;}
```

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

```
\def \_savedfontcolor{}
\def \_savedletterspace{}
\def \_savedws{}
\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\_removefeature{+Ws}\_if^#1^\_def \_setwsp##1{}\_removefeature{+Ws}\_else\_def \_setwsp\_setwspA{#1}\_setff{+Ws}\_fi
\reloading
\def \_setwsp #1{}
```

The `\_settterspace` and `\_setletterspace` are macros based on the special font features provided by LuaTeX (and by XeTeX too but it is not our business). The `\setwordspace` recalculates the `\fontdimen2,3,4` of the font using the `\setwsp` macro which is used by the `\_doresizeunifont` macro. It activates a dummy font feature +Ws too in order the font is reloded by the `\font` primitive (with independent `\fontdimen` registers).
2.13 Preloaded fonts for math mode

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

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

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

\setmathsizes[12/8.4/6]\normalmath ... math typesetting at 12 pt is ready.

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.

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 2.15.2.
The optical sizes of Computer Modern fonts, AMS and other fonts are declared here.

\texttt{math-preload.opm}

\% CM math fonts, optical sizes:
\begin{verbatim}
  \_regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
  \_regtfm cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
  \_regtfm cmtex 0 cstex8 8.5 cstex9 9.5 cstex10 *
  \_regtfm cmsy 0 cmsy5 5.5 cmsy6 6.5 cmsy7 7.5 cmsy8 8.5 cmsy9 9.5 cmsy10 *
  \_regtfm cmbsy 0 cmbsy5 5.5 cmbsy6 6.5 cmbsy7 7.5 cmbsy8 8.5 cmbsy9 9.5 cmbsy10 *
  \_regtfm cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
  \_regtfm cmexb 0 cmexb10 *

  \_regtfm cmr 0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
  \_regtfm cmbx 0 cmbx5 5.5 cmbx6 6.5 cmbx7 7.5 cmbx8 8.5 cmbx9 9.5
  \_regtfm cmti 0 cmti7 7.5 cmti8 8.5 cmti9 9.5 cmti10 11.1 cmti12 *
  \_regtfm cmtt 0 cmtt8 8.5 cmtt9 9.5 cmtt10 11.1 cmtt12 *

  \_regtfm msam 0 msam5 5.5 msam6 6.5 msam7 7.5 msam8 8.5 msam9 9.5 msam10 *
  \_regtfm msbm 0 msbm5 5.5 msbm6 6.5 msbm7 7.5 msbm8 8.5 msbm9 9.5 msbm10 *

  \_regtfm eufm 0 eufm5 6 eufm7 8.5 eufm10 *
  \_regtfm eufb 0 eufb5 6 eufb7 8.5 eufb10 *
  \_regtfm rsfs 0 rsfs5 6 rsfs7 8.5 rsfs10 *

  \_regtfm bfsans 0 ecss0x500 5.5 ecss0x600 6.5 ecss0x700 7.5 ecss0x800
  \_regtfm bbsans 0 ecss0x000 5.5 ecss0x000 6.5 ecss0x700 7.5 ecss0x800
  \_regtfm bbfbsans 0 ecss0x000 5.5 ecss0x000 6.5 ecss0x700 7.5 ecss0x800
  \_regtfm bbbsans 0 ecss0x000 5.5 ecss0x000 6.5 ecss0x700 7.5 ecss0x800

\end{verbatim}

% AMS math fonts, optical sizes:
\begin{verbatim}
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15
  \_regtfm mrm * 15

\end{verbatim}

% \_loadmathfamily \texttt{(number)} \texttt{(font)} loads one math family, i.e. the triple of fonts in the text size, script size and script-script size. The \texttt{(font)} is \texttt{(font-id)} used in the \_regtfm parameter or the real TFM name. The family is saved as \_fam\texttt{(number)}.

\_loadmathfamily \texttt{(number)} \texttt{(font)} loads one math family like \_loadmathfamily does it. But the second parameter is a \texttt{(font-switch)} declared previously by the \texttt{font} primitive. The font family is loaded at \_sizemtext, \_sizemscript and \_sizemsscript sizes. These sizes are set by the \texttt{setmathsizes} \texttt{[text-size]/[script-size]/[scriptscript-size]} macro. These parameters are given in the \texttt{ptmunit} unit, it is set to \texttt{1ptunit} and it is set to 1pt by default.

\_corrmsizes should be used in the \texttt{\normalmath} and \texttt{\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.

\texttt{math-preload.opm}

\def\corrmsizes{\ptmunit=\ptunit\_relax} % for corrections of sizes in different fonts
\edef\loadmathfamily#1#2{\_chardef\_tmp\#1\_corrmsizes
  \edef\optizesave{\_the\_optsize\X}
  \_optsize\_sizemtext \_font\_mF=\_ubichtfm\(#2\) \_optsize \_textfont\#1=\_mF

72
The \_setmathdimens macro is used in \_normalmath or \_boldmath macros. It makes math dimensions dependent on the font size (plain \TeX{} sets them only for 10 pt typesetting). The \_skewchar of some math families are set here too.

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.

2.14 Math macros

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

There is a problem: The \_n is tokenized as \_ and \_n and it works without problem. But \_\int_a^b is tokenized as \_\int_a, \_\^, \_b. The control sequence \_\int_a isn't defined. We must write \_\int_a^b. The Lua code presented here solves this problem. But you cannot set our own control sequence in the form \_\langle \langle word \rangle \_\text{one-letter} \rangle because such control sequences are inaccessible: preprocessor rewrites it.

The \_mathsbon macro activates the rewriting rule \_\langle \langle word \rangle \_\langle nonletter \rangle \rangle to \_\langle \langle word \rangle \_\langle nonletter \rangle \rangle and \_\langle \langle word \rangle \_\langle letter \rangle \langle nonletter \rangle \rangle to \_\langle \langle word \rangle \_\langle letter \rangle \langle nonletter \rangle \rangle at input processor level. The \_mathsboff deactivates it. You can ask by \_ifmathsb if this feature is activated or deactivated. By default, it is activated in the everyjob, see section 2.1. Note, that the everyjob is processed after the first line of the document is read, so the \_mathsbon is activated from second line of the document.
All mathcodes are set to equal values as in plain\TeX. But all encoding-dependant declarations (like these) will be set to different values when Unicode-math font is used.
All control sequences declared by \mathcode\ 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.

%\protected\def\log {\_mathop{\_rm log}\_nolimits}
%\protected\def\lg {\_mathop{\_rm lg}\_nolimits}
%\protected\def\ln {\_mathop{\_rm ln}\_nolimits}
%\protected\def\lim {\_mathop{\_rm lim}}
%\protected\def\limsup {\_mathop{\_rm lim\_thinsk sup}}
%\protected\def\liminf {\_mathop{\_rm lim\_thinsk inf}}
%\protected\def\sin {\_mathop{\_rm sin}\_nolimits}
%\protected\def\arcsin {\_mathop{\_rm arcsin}\_nolimits}
%\protected\def\sinh {\_mathop{\_rm sinh}\_nolimits}
%\protected\def\cos {\_mathop{\_rm cos}\_nolimits}
%\protected\def\arccos {\_mathop{\_rm arccos}\_nolimits}
%\protected\def\cosh {\_mathop{\_rm cosh}\_nolimits}
%\protected\def\tan {\_mathop{\_rm tan}\_nolimits}
%\protected\def\arctan {\_mathop{\_rm arctan}\_nolimits}
%\protected\def\tanh {\_mathop{\_rm tanh}\_nolimits}
%\protected\def\cot {\_mathop{\_rm cot}\_nolimits}
%\protected\def\coth {\_mathop{\_rm coth}\_nolimits}
%\protected\def\sec {\_mathop{\_rm sec}\_nolimits}
%\protected\def\csc {\_mathop{\_rm csc}\_nolimits}
%\protected\def\max {\_mathop{\_rm max}}
%\protected\def\min {\_mathop{\_rm min}}
%\protected\def\sup {\_mathop{\_rm sup}}
%\protected\def\inf {\_mathop{\_rm inf}}
%\protected\def\arg {\_mathop{\_rm arg}\_nolimits}

...etc. (see math-macros.omp)

math-macros.omp

\protected\def\log {\_mathop{\_rm log}\_nolimits}
\protected\def\lg {\_mathop{\_rm lg}\_nolimits}
\protected\def\ln {\_mathop{\_rm ln}\_nolimits}
\protected\def\lim {\_mathop{\_rm lim}}
\protected\def\limsup {\_mathop{\_rm lim\_thinsk sup}}
\protected\def\liminf {\_mathop{\_rm lim\_thinsk inf}}
\protected\def\sin {\_mathop{\_rm sin}\_nolimits}
\protected\def\arcsin {\_mathop{\_rm arcsin}\_nolimits}
\protected\def\sinh {\_mathop{\_rm sinh}\_nolimits}
\protected\def\cos {\_mathop{\_rm cos}\_nolimits}
\protected\def\arccos {\_mathop{\_rm arccos}\_nolimits}
\protected\def\cosh {\_mathop{\_rm cosh}\_nolimits}
\protected\def\tan {\_mathop{\_rm tan}\_nolimits}
\protected\def\arctan {\_mathop{\_rm arctan}\_nolimits}
\protected\def\tanh {\_mathop{\_rm tanh}\_nolimits}
\protected\def\cot {\_mathop{\_rm cot}\_nolimits}
\protected\def\coth {\_mathop{\_rm coth}\_nolimits}
\protected\def\sec {\_mathop{\_rm sec}\_nolimits}
\protected\def\csc {\_mathop{\_rm csc}\_nolimits}
\protected\def\max {\_mathop{\_rm max}}
\protected\def\min {\_mathop{\_rm min}}
\protected\def\sup {\_mathop{\_rm sup}}
\protected\def\inf {\_mathop{\_rm inf}}
\protected\def\arg {\_mathop{\_rm arg}\_nolimits}
These macros are defined similarly as in plainTeX. Only internal macro names from plainTeX with @ character are re-written in more readable form.

\sp is alternative for ^. The \sb alternative for _ was defined at the line 27 of the file math-macros.opm.

Active \prime character is defined here.

\big, \Big, \bigg, \Bigg, \bigl, \bigm, \bigr, \Bigl, \Bigm, \Bigr, \biggl, \biggm, \biggr, \Biggl, \Biggm, \Biggr are based on the \_scalebig macro because we need the dependency on the various sizes of the fonts.

Math relations defined by the \jointrel plain \TeX macro:
\ldots, \cdots, \vdots, \ddots from plain \TeX

\ldots \cdots \vdots \ddots from \LaTeX

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

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

\texttt{math-macros.opm}

\begin{verbatim}
\protected\def\cong\{\mathrel{\mathpalette\overeq\sim}\} % congruence sign
\def\overeq#1#2{\lower.05em\vbox{
\lineskiplimit\maxdimen\lineskip=-.05em
\ialign{$\math#1\hfil##\hfil$\crcr #2\crcr=\crcr}}}
\protected\def\notin\{\mathrel{\mathpalette\cancel\in}\}
\def\cancel#1#2{\math\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
\protected\def\rightleftharpoons\{\mathrel{\mathpalette\rlhp{}}\}
\def\rlhp#1{\vcenter{\math\hbox{
\ooalign{$\raise.2em\hbox{$#1\rightharpoonup$}\crcr
$#1\leftharpoondown$}}}}
\protected\def\buildrel#1\over#2{\mathrel{\mathop{\kern0pt #2}\limits^{#1}}}
\protected\def\doteq\{\buildrel\textstyle.\over=\}
\protected\def\bmod\{\nonscript\mskip-\medmuskip\mkern5mu
\mathbin{\rm mod}\penalty900\mkern5mu\nonscript\mskip-\medmuskip\}
\protected\def\pmod#1\{\allowbreak\mkern18mu({\rm mod}\thinspace\thinspace\thinspace\thinspace\thinspace#1)}
\end{verbatim}

\texttt{math-macros.opm}

\begin{verbatim}
\protected\def\matrix#1{\null_thinsk
\edef\stylenum{\the\numexpr\mathstyle/2\relax}\
\vcenter{\matrixbaselines\math
\ialign{\the_lmf\hfil##\hfil&\quad{##\unsskip}\hfil\hfil\crcr
\mathstrut\crcr
#1\crcr
\mathstrut\crcr
\noalign{\kern-\baselineskip}}}
\null_thicksk\vbox{\kern\ht1\box2}}\endgroup}
\def\matrixscriptbaselines{\baselineskip=.7\baselineskip
\def\quad {\hskip.7em\relax}
\let\matrixstyle=\scriptstyle}
\protected\def\pmatrix#1{\left(\matrix{#1}\right)}
\end{verbatim}

The \texttt{\textbackslash cases} and \texttt{\textbackslash bordermatrix} macros are identical from plain \TeX.

\texttt{math-macros.opm}

\begin{verbatim}
\protected\long\def\cases#1{\left\{\math
\ialign{##\hfil&\quad##\unsskip\hfil##\hfil\crcr
#1\crcr}}\right.}
\newdimen\ptrenwd
\ptrenwd=8.75pt % width of the big left (\
\protected\def\bordermatrix#1{\begingroup \math
\matrixbaselines=\null\math
\ialign{##\hfil&\quad##\unsskip\hfil##\hfil&&\quad##\unsskip\hfil##\hfil\crcr
\omit\strut\hfil\crcr
#1\crcr\noalign{\kern.2em}#2\crcr\omit\strut\cr}}
\endgroup}
\end{verbatim}

79
The \eqalign macro behaves like in Plain TeX by default. It creates the vcenter in the math mode. The contents is two column \halign with right aligned left column and left aligned right column. The table items are in \displaystyle and the \vcenter is advanced by \jot (3pt in plain TeX). It follows from the default settings of \eqlines and \eqstyle parameters.

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

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

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

OpTeX offers \displaylines to⟨dimen⟩{⟨formula⟩\cr⟨formula⟩\cr...⟨formula⟩} as an alternative case of usage \displaylines. See section 4.3 in the Typesetting Math with OpTeX. The centered formulas are in vcenter in this case, so lines cannot be broken to more pages, but this case enables to use \eqno or \leqno or \eqmark.

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

These macros are inspired from 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{\hbox{$\currstyle#1$}}}}

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.
2.15 Unicode-math fonts

The \loadmath \{Unicode-math font\} macro loads math fonts and redefines all default math-codes using \input unimath-codes.opm. If Unicode-math font is loaded then \_mathloadingfalse is set, so new UnicodeMath font isn’t loaded until \doaloadmath 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

2.15.1 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.
The Unicode version of the \normalmath and \boldmath macros are defined here as \normalunimath and \boldunimath macros. They are using \setunimathdimens in similar sense as \setmathdimens.

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

Unicode math font includes all typical math alphabets together, user needs not to load more \TeX math families. These math alphabets are encoded by different parts of Unicode table. We need auxiliary macros for setting mathcodes by selected math alphabet.

\umathrange \langle from \rangle -- \langle to \rangle \langle class \rangle \langle family \rangle \langle first \rangle sets \Mathcodes of the characters in the interval...
\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 from\rangle-\langle to \rangle clause includes normal letters like A-Z.

\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 from\rangle-\langle to \rangle clause includes normal letters like A-Z.

\umathrange \langle first \rangle is the same as \umathrange {\langle alpha \rangle-\langle omega \rangle} \langle first \rangle. \umathrangeGREEK \langle first \rangle is the same as \umathrange {\langle Alpha \rangle-\langle Omega \rangle} \langle first \rangle. \greekdef \langle control sequences \rangle \_relax defines each control sequence as a normal character with codes \umathnumB, \umathnumB+1, \umathnumB+2 etc. It is used for redefining the contol sequences for math Greek \\textbackslash alpha, \\beta, \\gamma etc.

2.15.2 Macros and codes set when \loadmatfont is processed

The file unimath-codes.opm is loaded when the \loadmath is used. The macros here redefines globally all encoding dependent settings declared in the section 2.14.

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

The math alphabets are declared here using the \umathrange{\langle range \rangle}{\langle class \rangle}{\langle family \rangle}{\langle starting-code \rangle} macro.
The \cal, \bbchar, \frak, \script and the \rm, \bf, \it, \bi, \tt are defined here. Their “8bit definitions” from the file \math-preload.opm (section 2.13) are removed. You can redefine them again if you need different behavior (for example you don’t want to use sans serif bold in math). What to do:
\_protected\_def\_bf
\{_tryloadbf\_tenbf \_inmath\{_bfvariables\_bfGreek\_bfdigits\}
\_protected\_def\_bi
\{_tryloadbi\_tenbi \_inmath\{_bivariables\_bigreek\_bfGreek\_bfdigits\}
\_public \bf \bi ;
\_inmath \{\_cmds\} applies \{\_cmds\} only in math mode.

% You can redefine these macros to follow your wishes.
% For example you need upright lowercase greek letters, you don't need
% \bf and \bi behaves as sans serif in math, ...
\_protected\_def\_rm
\{_tryloadrm\_tenrm \_inmath\{_rmvariables \_rmdigits\}
\_protected\_def\_it
\{_tryloadit\_tenit \_inmath\{_itvariables\}
\_protected\_def\_bf
\{_tryloadbf\_tenbf \_inmath\{_bsansvariables \_bsansgreek \_bsansGreek \_bsansdigits\}
\_protected\_def\_bi
\{_tryloadbi\_tenbi \_inmath\{_bisansvariables \_bisansgreek \_bsansGreek \_bsansdigits\}
\_protected\_def\_tt
\{_tryloadtt\_tentt \_inmath\{_ttvariables \_ttdigits\}
\_protected\_def\_bbchar
\{_bbvariables \_bbdigits\}
\_protected\_def\_cal
\{_calvariables\}
\_protected\_def\_frak
\{_frakvariables\}
\_protected\_def\_misans
\{_isansvariables \_sansdigits\}
\_protected\_def\_mbisans
\{_bisansvariables \_bisansgreek \_bsansGreek \_bsansdigits\}
\_protected\_def\_script
\{_rmvariables \_fam4 \}
\_protected\_def\_mit
\{_itvariables \_rmdigits \_itgreek \_rmGreek \}
\_public \rm \it \bf \bi \tt \bbchar \cal \frak \misans \mbisans \script \mit ;

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

```
002E;P
002F;B
0030..0039;N
003A;P
003B;P
003C;R
003D;R
003E;R
003F;P
0040;N
0041..005A;A
005B;O
005C;B
005D;C
005E;N
005F;N
```

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

```
\begingroup % \input mathclass.opm (which is a copy of MathClass.txt):
\_def\_p#1;#2\{_edef\_tmp\{_pB#2\}_ifx\_tmp\_empty \_else\_pA#1\_end\_fi\}
\_def\_pA#1..#2..#3\_end#4{%
\_ifx\_relax#2\_relax \_pset\{"#1\}{#4}\_else
\umathnumA=\_tmp
\_loop
\_pset\{"\umathnumA\}{#4}%
\_ifnum\umathnumA=\_advance\umathnumA by1
\_repeat
\_fi
%
\_def\_pB#1;\{_edef\_tmp\{_pB#1\}_ifx\_tmp\_empty \_else\_pA#1\_end\_fi\}
\_def\_pA#1\_end#2\_if#3\_end#4{%
\_if#3\_if\_pset\{"#1\}{#4}\_else
\umathnumA=\_tmp
\_loop
\_pset\{"\umathnumA\}{#4}%
\_ifnum\umathnumA=\_advance\umathnumA by1
\_repeat
\_fi
%
\_if#3\_if#4\_if#5\_if#6\_if#7\_if#8\_if#9\_if#10\_if#11\_if#12\_if#13\_if#14\_if#15\_if#16\_if#17\_if#18\_if#19\_if#20\_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits \_global\_Udelcode#1=1 \_global\_tdigits
```

86
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:

```
\UnicodeMathSymbol{"00394}{\mupDelta}{\mathalpha}{capital delta, greek}\
\UnicodeMathSymbol{"00395}{\mupEpsilon}{\mathalpha}{capital epsilon, greek}\
\UnicodeMathSymbol{"00396}{\mupZeta}{\mathalpha}{capital zeta, greek}\
\UnicodeMathSymbol{"00397}{\mupEta}{\mathalpha}{capital eta, greek}\
\UnicodeMathSymbol{"00398}{\mupTheta}{\mathalpha}{capital theta, greek}\
\UnicodeMathSymbol{"00399}{\mupIota}{\mathalpha}{capital iota, greek}\
\UnicodeMathSymbol{"0039A}{\mupKappa}{\mathalpha}{capital kappa, greek}\
\UnicodeMathSymbol{"0039B}{\mupLambda}{\mathalpha}{capital lambda, greek}\
\UnicodeMathSymbol{"0039C}{\mupMu}{\mathalpha}{capital mu, greek}\
\UnicodeMathSymbol{"0039D}{\mupNu}{\mathalpha}{capital nu, greek}\
\UnicodeMathSymbol{"0039E}{\mupXi}{\mathalpha}{capital xi, greek}\
\UnicodeMathSymbol{"0039F}{\mupOmicron}{\mathalpha}{capital omicron, greek}\
\UnicodeMathSymbol{"003A0}{\mupPi}{\mathalpha}{capital pi, greek}\
\UnicodeMathSymbol{"003A1}{\mupRho}{\mathalpha}{capital rho, greek}\
\UnicodeMathSymbol{"003A3}{\mupSigma}{\mathalpha}{capital sigma, greek}\
\UnicodeMathSymbol{"003A4}{\mupTau}{\mathalpha}{capital tau, greek}\
```

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

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

Many special characters must be declared with care...

```
\begingroup % \input unimath-table.opm (it is a copy of unicode-math-table.tex):
\def\UnimathSymbol % #1#2#3#4{%\
  \global\Unimathcharnumdef#2=\Unimathcodenum#1\relax\
  \ifx#3\mathopen \gdef#2{\Udelimiter 4 1 #1}\fi\
  \ifx#3\mathclose \gdef#2{\Udelimiter 5 1 #1}\fi\
  \ifx#3\mathaccent \gdef#2{\Umathaccent fixed 7 1 #1}\fi\
}\input unimath-table.opm
\endgroup
```
The \texttt{\not} macro is redefined here. If the \texttt{\_not!(char)} is defined (by \texttt{\_negationof}) then this macro is used. Else centered / is printed over the (char).

\begin{verbatim}
\ProtectedDef\not#1{\
  \trycs{\not!\csstring#1}{\mathrel\mathstyles{
    \setbox0=\hbox{\math\currstyle#1}\
    \hbox to \wd0{\hss\currstyle/\hss}\kern-\wd0 \box0}}}
\def\negationof #1#2{\ea\let \csname _not!\csstring#1\endcsname =#2}
\negationof = \neq\negationof < \nless\negationof > \ngtr\negationof \gets \nleftarrow\negationof \simeq \nsime\negationof \equal \ne\negationof \le \nleq\negationof \ge \ngeq\negationof \greater \ngtr\negationof \forksnot \forks\negationof \in \notin\negationof \mid \nmid\negationof \cong \ncong\negationof \leftarrow \nleftarrow\negationof \rightarrow \nrightarrow\negationof \leftrightarrow \nleftrightarrow\negationof \Leftarrow \nLeftarrow\negationof \Leftrightarrow \nLeftrightarrow\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 \gtrless \ngtrless\negationof \prec \nprec\negationof \succ \nsucc\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 \OpTeX{}. We need to get new meanings of these control sequences in private name space.

\begin{verbatim}
\def\rvariables{} % The default value of \normalmath shows a combination of base Unicode Math font with 8bit Math font at family 4. See definition of the \script macro where \fam4 is used. Of course, we need to set \rvariables too, because 8bit font accepts only codes less than 255. XITSmath-bold needs correction: the norm symbol \[\|x\|\] is missing here. So, you can define: \\
\def\boldmath{% \\
  \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}

\subsection{Printed 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.

\begin{verbatim}
\begingroup \undefined {Printing Unicode-math table \string<2020-06-08>}
\end{verbatim}
2.16 Scaling fonts in document (high-level macros)

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

```
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
\textsize \mathsize \sizemtext \sizemscript \sizemsscript
\textsizes \mathsizes \setmainvalues \setbaselineskip
```

```
Codes U+00000 \_dots \ U+10000
\begmulti 3
\_input unimath-table.opm
\_endmulti
\medskip \goodbreak
Codes U+10001 \_dots \ U+1EEF1 \_let\UnicodeMathSymbol=\UnicodeMathSymbolA
\begmulti 4
\_input unimath-table.opm
\_endmulti
\endgroup
```
\_setbaselineskip \{(baselineskip)\} sets new \baselineskip and more values of registers which are dependent on the \baselineskip including the \strutbox.

\_def \_setbaselineskip #1{\_if$#1$\_else
\_tmpdim=#1\ptunit
\_baselineskip=\_tmpdim \_relax
\_bigskipamount=\_tmpdim plus 33333\_tmpdim minus 33333\_tmpdim
\_medskipamount=5\_tmpdim plus 16666\_tmpdim minus 16666\_tmpdim
\_smallskipamount=2.5\_tmpdim plus 08333\_tmpdim minus 08333\_tmpdim
\_normalbaselineskip=\_tmpdim
\_jot=2.5\_tmpdim
\_maxdepth=33333\_tmpdim
\_setbox\_strutbox=\_hbox{\_vrule height 0.709\_tmpdim depth 0.291\_tmpdim width 0pt}\
\_fi
}

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

\_def \_setmainvalues \{\% \_mainbaselineskip=\_baselineskip \_mainfontsize=\_optsize \_topskip=\_mainfontsize \_splittopskip=\_topskip \_ifmmode \_else \_bf \_it \_bi \_rm \_fi \_normalmath \_let \_setmainvalues=\_setmainvaluesL \_fi
\}

\_def \_setmainvaluesL \{\_ifmode \_normalmath \_else \_rm \_everymath=\_normalmath \_everydisplay=\_normalmath \_fi\}
\_def \_scalemain \{\% \_ifdim \_mainfontsize=0pt \_mainfontsize=10pt \_mainbaselineskip=12pt \_let \_setmainvalues=\_setmainvaluesL \_fi
\_optsize=\_mainfontsize \_mainbaselineskip=\_mainbaselineskip
\}
\_public \calemain \mainfontsize \mainbaselineskip ;

\_protected \_def \_thefontsize[#1]{\_if$#1$\_else
\_tmpdim=#1\ptunit
\_newcurrfontsize{at \_tmpdim}\%
\_fi
\}
\_protected \_def \_thefontscale[#1]{\_if$#1$\_else
\_tmpdim=\_divide\_tmpdim by 1000
\_tmpdim=\_ea\_ea\_ea\_ignorept \_pdffontsize\_font \_tmpdim
\_newcurrfontsize{at \_tmpdim}\%
\_fi
\}
\_public \calemain \thefontsize \thefontscale ;

\_protected \_def \_em \{\%
\_ea\_ifx \_the\_font \_tenit \_additcorr \_rm \_else
\_\fi
\}

\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 `\typsize` command and redefines itself be only the font switch for next pages).

```
\let\rmfixed=\tenrm % user can redefine it
```

### 2.17 Output routine

The output routine `\optexoutput` is similar as in plain \TeX. It does:

- `\begoutput` which does:
  - increments `\pageno`,
  - prints `\_Xpage{⟨\pageno⟩}{⟨\pageno⟩}` to the `.ref` file (if `\openref` is active),
  - calculates `\hoffset`,
  - sets local meaning of macros used in headlines/footlines (see `\regmacro`).
- `\shipout\_completepage`, which is `\vbox` of –
  - background box, if `\pgbackground` is non-empty,
  - headline box by `\_makeheadline`, if the `\headline` is non-empty,
  - `\vbox to\vsize` of `\_pagecontents` which consists of –
    - `\pagedest`, the page destination `pg:⟨\pageno⟩` for hyperlinks is created here,
    - `\topins` box if non-empty (from `\topinserts`),
    - `\box255` with completed vertical material from main vertical mode,
    - `\_footnoterule` and `\footins` box if non-empty (from `\fnote`, `\footnote`),
    - `\pgbottomskip` (default is 0 pt),
  - footline box by `\_makefootline`, if the `\footline` is non-empty
- `\endoutput` which does:
  - increments `\pageno` using `\advance\pageno`
  - runs output routine repeatedly if `\dosupereject` is activated.

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

\begin{verbatim}
\_def \_begoutput{\_incr\_gpageno
\_immediate\_wref\_Xpage{\_the\_gpageno}{\_folio}\
\_setxhsize \_prepoffs \_the\_regmark}
\_def \_endoutput{\_advancepageno
\_ifnum\_outputpenalty>-20000 \_else\_dosupereject\_fi
}\_def \_prepoffs { }
\end{verbatim}

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

\begin{verbatim}
\_newdimen \_xhsize
\_def \_setxhsize {\_global\_xhsize=\_hsize \_global\_let \_setxhsize=\_relax}
\end{verbatim}

\_gpageno counts pages from one in whole document.

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

The \_ensureblack sets the typesetting of its parameter locally to \Black color. We needn’t do this if colors are never used in the document. So, default value of the \_ensureblack macro is empty. But first usage of color macros in the document re-defines \_ensureblack. See the section 2.19 for more details.

\begin{verbatim}
\_def \_completepage{\_vbox{\_istoksempty \_pgbackground
\_iffalse \_ensureblack{\_backgroundbox{\_the\_pgbackground}}\_nointerlineskip \_fi
\_ensureblack{\_makeheadline}\
\_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
\_ensureblack{\_makefootline}\
}\_fi
}\_def \_makeheadline {\_istoksempty \_headline \_iffalse
\_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
\_ensureblack{\_makefootline}\
}\_iffalse \_ensureblack{\_backgroundbox{\_the\_pgbackground}}\_nointerlineskip \_fi
\_ensureblack{\_makeheadline}\
}\_vbox to\_vsize {\_boxmaxdepth=\_maxdepth \_pagecontents}% \pagebody in plainTeX
\_ensureblack{\_makefootline}\
}\_fi
\else
\_let \_openfnotestack = \_relax % will be re-defined by color macros
\_let \_openfnotestack = \_relax % will be re-defined by color macros
\_def \_backgroundbox #1{\_moveleft\_hoffset\_vbox to\_vsize{\_kern-\_voffset #1\_vss}}
\end{verbatim}

\_makeheadline creates \_vbox toOpt with its contents (the \_headline) shifted by \headlinedist up.

\begin{verbatim}
\_def \_makefootline {\_istoksemi\_fline \_iffalse
\_bbox to\_vsize \_boxmaxdepth=\_maxdimen
\_bbox to\_xhsize \_the\_fline \_bbox()\_nointerlineskip
\_fi
}\_def \_makefootline {\_istoksemi\_fline \_iffalse
\_bbox to\_vsize \_boxmaxdepth=\_maxdimen
\_bbox to\_xhsize \_the\_fline \_bbox()\_nointerlineskip
\_fi
\end{verbatim}

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

\begin{verbatim}
\_def \_footnoterule {\_istoksemi\_fline \_iffalse
\_bbox to\_vsize \_boxmaxdepth=\_maxdimen
\_bbox to\_xhsize \_the\_fline
\_fi
}\_def \_footnoterule {\_istoksemi\_fline \_iffalse
\_bbox to\_vsize \_boxmaxdepth=\_maxdimen
\_bbox to\_xhsize \_the\_fline
\_fi
\end{verbatim}

\_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.
\_def\_pagedest\ {\_def\_destheight\ {25pt}\_dest\ [pg:\ \the\ gpageno]}\}

\_def\_footnoterule\ {\_kern\-3pt\_hrule\ width\ 2truein\_kern\ 2.6pt}\}

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

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

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

The \_topins macros \_topinsert, \_midinsert, \_pageinsert, \_endinsert are the same as in plain \TeX.
The \draft macro is an example of usage \_pgbackground to create water color marks.

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 \hoffset=0pt \voffset=0pt \if$#3$\if$#4$\hoffset =\dimexpr (\pgwidth -\hsize)/2 \relax \else \hoffset =\dimexpr \pgwidth -\hsize - #4\relax \fi \else \if$#4$\hoffset = #3\relax \else \hsize=\dimexpr \pgwidth - #3 - #4\relax \hoffset = #3\relax \fi \fi \if$#5$\if$#6$\voffset =\dimexpr (\pgheight -\vsize)/2 \relax \else \voffset =\dimexpr \pgheight - #5\relax \voffset \relax \fi \else \if$#6$\voffset = #5\relax \else \vsize=\dimexpr \pgheight - #5 - #6\relax \voffset \relax \fi \fi \if 1#1\shiftoffset=0pt \else \if 2#1% double-page layout \shiftoffset = \dimexpr \pgwidth -\hsize -2\hoffset \relax \fi \fi \if #3\relax \fi \if #4\relax \fi \if #5\relax \fi \if #6\relax \fi\else \if 2#1% double-page layout \_shiftoffset=0pt \_define \preoffsets{} \else \if 2#1% double-page layout \_shiftoffset = \_dimexpr \_pgwidth -\_hsize -2\_hoffset \_relax \fi \fi

2.18 Margins

The \margins macro is documented in the section 1.2.1.
The common page dimensions are defined here.

\sdef\_pgs:a3\{\{297,420\}mm\} \sdef\_pgs:a4\{\{210,297\}mm\} \sdef\_pgs:a5\{\{148,210\}mm\}
\sdef\_pgs:a3l\{\{420,297\}mm\} \sdef\_pgs:a4l\{\{297,210\}mm\} \sdef\_pgs:a5l\{\{210,148\}mm\}
\sdef\_pgs:b5\{\{176,250\}mm\} \sdef\_pgs:letter\{\{8.5,11\}in\}

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

2.19 Colors

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

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

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

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

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

The color mixing processed by the \colordef is done in the 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 \useK command at the end of a linear combination used in \colordef. For example

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

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

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

\colordef \Color {\Brown-\Black}
can be used for removing black component from the color. You can use the ~\Black trick after \useK command in order to remove grey components occured during color mixing.

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

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

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

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

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

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

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

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.
The `\onlyrgb` declaration redefines `\_formatcmyk` in order it expands to its conversion to RGB (pdf-primitive). This conversion is done by the `\_cmyktorgb` macro. Moreover, `\onlyrgb` re-defines three basic RGB colors for RGB color space and re-declares `\colordef` as `\rgbcolordef`. The `\onlycmyk` macro does a similar work, it re-defines `\_formatrgb` macro. The Grey color space is unchanged and works in both main settings (RGB or CMYK) without collisions.

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` expands to `\_cmyktorgb` macro. `\_cmyktorgb` expands to `\_colorcrop`, `\_colordefFin` and `\_douseK` are auxiliary macros used in the `\colordef`. The `\_colorcrop` rescales color components in order to they are in [0,1] interval. The `\_colordefFin` expands to the values accumulated in Lua code `color_C, color_M, color_Y and color_K`. The `\_douseK` applies `\useK` to CMYK components.
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)⟨\K⟩⟨\R⟩⟨\G⟩⟨\what-define⟩⟨\data⟩ does the real work. It initializes the Lua variables for summation. It expands ⟨\data⟩ in the group where color selectors have special meaning, then it adjusts the resulting string by \replstring and runs it. Example shows how the ⟨\data⟩ are processed:

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

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

Next, \commoncolordef saves resulting values from Lua to \tmpb using \colordefFin. If \rgbcolordef is processed, then we must to remove the last ⟨\K⟩ component which is in the
format .0 in such case. The \stripK macro does it. Finally, the \(\text{what-define}\) is defined as \langle\text{selector}\rangle\{\langle\text{expanded \_tmpb}\rangle\}, for example \_setcmykc{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 \LaTeX{} file \texttt{x11nam.def} is read by \morecolors. The numbers 0,1,2,3,4 are transformed to letters O, ⟨none⟩, B, C, D in the name of the color. Colors defined already are not re-defined. The empty \_showcolor macro should be re-defined for color catalog printing. For example:

\begin{verbatim}
\def\vr{\vrule height10pt depth2pt width20pt}
\def\_showcolor{\hbox{\tt\_bslash\_tmpb: \csname\_tmpb\endcsname \vr}\space\space}
\begmulti 4 \texttt{\morecolors}
\endmulti
\end{verbatim}
2.20 The .ref file

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

\begin{verbatim}
\Xrefversion{⟨ref-version⟩}
\Xpage{⟨gpageno⟩}{⟨pageno⟩}
\Xtoc{⟨level⟩}{⟨type⟩}{⟨text⟩}{⟨title⟩}
\Xlabel{⟨label⟩}{⟨text⟩}
\Xpage{⟨gpageno⟩}{⟨pageno⟩}
\Xlabel{⟨label⟩}{⟨text⟩}
\end{verbatim}

where ⟨gpageno⟩ is internal page number globally numbered from one and ⟨pageno⟩ is a page number (\the\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.

\begin{verbatim}
\newwrite\reffile
\def\inputref{\if\isfile{\jobname.ref}\iftrue
\input {\jobname.ref}
\gfnoteum=0 \lfnoteum=0 \mnoteum=0
\openref{⟨string⟩}{\inputref}\fi}
\f i
}
\end{verbatim}

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.

\begin{verbatim}
\def\wrefrelax#1#2{}
\let\wref=\wrefrelax
\end{verbatim}
If a macro needs to create and to use .ref file then such macro must use \openref. When the file is created (using internal \_openrefA) then the \wref \{macro\}{\{data\}} is redefined in order to save the line \{macro\}/\{data\} to the .ref file using asynchronous \write primitive. Finally, the \openref destroys itself, because we need not to open the file again.

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{\textit{version}}

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

2.21 References

If the references are “forward” (i.e. the \ref is used first, the destination is created later) or if the reference text is page number then we must read .ref file first in order to get appropriate information. See section 2.20 for more information about .ref file concept.

\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.
\_\newcount\_unresolvedrefs
\_unresolvedrefs=0

\_\_Xlabel {\langle\text{label}\rangle}\{\langle\text{text}\rangle\}\text{saves the}\langle\text{text}\rangle\text{to}\_\lab:\langle\text{label}\rangle\text{and saves}\_\pgref:\langle\text{label}\rangle.}

\_\def\_Xlabel\#1\#2{\_sdef\_\lab:\#1\{\#2\}\_sxdef\_\pgref:\#1\{\_ea\_bracketspg\\_currpage\}}

\_\def\\bracketspg\#1\#2{\[\text{pg:}\#1\]{\#2}}

\_\label\\{\langle\text{label}\rangle\}\text{saves the declared label to}_\_\lastlabel\langle\text{label}\rangle\text{and}\wlabel\{\langle\text{text}\rangle\}\text{uses}_\_\lastlabel\text{andactivates}_\wref\_\Xlabel\{\langle\text{label}\rangle\}\{\langle\text{text}\rangle\}.}

\_\def\_\label[\#1]{\_isempty\{\#1\}\_iftrue\_global\_let\_\lastlabel=\_undefined\_else\_isdefined\{l0:\#1\}\%\_iftrue\_opwarning{duplicated label \#1, ignored}\_else\_xdef\_\lastlabel\{\#1\}\fi\_fi\_ignorespaces}

\_\def\_wlabel\#1{\_ifx\_\lastlabel\_undefined\_else\_dest[ref:\_\lastlabel]\%\_printlabel\_\lastlabel\_\edef\_tmp\{{\_\lastlabel}{\#1}\}\_ea\_wref\_ea\_ea\_\Xlabel\_ea\{\_tmp\}\%\_sxdef\_\lab:\_\lastlabel\{\#1\}\_sxdef\l0:\_\lastlabel\{}\_global\_let\_\lastlabel=\_undefined\_fi\_fi}

\_\public\_\label\_\wlabel\;\_\ref\{\langle\text{label}\rangle\}\text{uses saved}_\_\lab:\langle\text{label}\rangle\text{and prints (linked)\langle\text{text}\rangle. If the reference is backwarded then we know}_\_\lab:\langle\text{label}\rangle\text{without any need to read REF file. On the other hand, if the reference is forwarded, then we doesn’t know}_\_\lab:\langle\text{label}\rangle\text{in first run of }\_\TeX\text{and we print warning and do}_\_\openref.\_\pgref\{\langle\text{label}\rangle\}\text{uses}\_\{\langle\text{pageno}\rangle\}_\{\langle\text{pageno}\rangle\}\text{from}_\_\pgref\{\langle\text{label}\rangle\}\text{and prints (linked)\langle\text{pageno}\rangle\text{using}_\_\ilink\text{macro.}}

\_\def\_\ref[\#1]{\_isdefined\{\_\lab:\#1\}\%\_iftrue\_ilink[ref:\#1]\{\_csname _\lab:\#1\_endcsname\}\%\_else??\_opwarning{label \#1 unknown. Try to }\_\TeX\text{me again}\%\_incr\_unresolvedrefs \_\openref\_\fi\_fi}

\_\def\_\pgref[\#1]{\_isdefined\{\_\pgref:\#1\}\%\_iftrue\_ea\_ea\_ea\_ilink\_csname _\pgref:\#1\_endcsname\%\_else??\_opwarning{pg-label \#1 unknown. Try to }\_\TeX\text{me again}\%\_incr\_unresolvedrefs \_\openref\_\fi\_fi}

\_\public\_\ref\_\pgref\;\_\showlabels\redefines it as box with zero dimensions and with left lapped\{\langle\text{label}\rangle\}\text{in blue 10pt }\tttextfont shifted up by 1.7ex.\_\\fontdef\_\labelfont\{\setfontsize{at10pt}\setfontcolor{blue}\tt\kern1.7ex}\_\fi}

2.22 Hyperlinks

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

- \_\ref[\langle\text{label}\rangle] – the destination is created when \_\label[\{\langle\text{label}\rangle\}] is used, see also the section 2.21.
- \_\toc[\langle\text{tocrefnum}\rangle] – the destination is created at chap/sec/secct titles, see also the section 2.23.
- \_\pg[\langle\text{pageno}\rangle] – the destination is created at beginning of each page, see also the section 2.17.
- \_\cite[\langle\text{bibnum}\rangle] – the destination is created in bibliography reference, see also the section 2.31.1.
- \_\url[\langle\text{url}\rangle] – used by \url or \ulink, see also the end of this section.
The \texttt{tocrefnum}, \texttt{gpageno} and \texttt{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{label}}, then both types of internal links are created at the same destination place: \texttt{toc:tocrefnum} and \texttt{ref:bibnum}.

\texttt{\_hyperlinks{\texttt{\label{label}}}} \% preloaded in format

\texttt{\dest[type:spec]} creates a destination of internal links. The destination is declared in the format \texttt{\hyperlinks{type:spec}}. If the \texttt{\hyperlinks} command in not used, then \texttt{\dest} does nothing else it is set to \texttt{\destactive}. The \texttt{\destactive} is implemented by \texttt{\_pdfdest} primitive. It creates a box in which the destination is shifted by \texttt{\destheight}. The reason is that the destination is exactly at top border of the PDF viewer but we want to see the line where destination is. The destination box is positioned by different way dependent on current vertical or horizontal mode.

\texttt{\_protected\_def\_linkactive[#1:#2][\_leavemode\_pdfstartlink\_pdfendlink]} \% preloaded in format

\texttt{\_protected\_def\_urlactive[#1:#2][\_leavemode\_pdfstartlink\_pdfendlink]} \% preloaded in format

\texttt{\_protected\_def\_urlcolor{}\% preloaded in format}

\texttt{\_protected\_def\_urlcolor{}\% preloaded in format}

\texttt{\_protected\_def\_urlcolor{}\% preloaded in format}

The \texttt{\pdfstartlink} primitive uses \texttt{\pdfborder{type}} in its parameter (see \texttt{\linkactive} or \texttt{\urlactive} macros). The \texttt{\pdfborder{type}} expands to \texttt{attr{/Ci[-2]}/Border[0 0 .6]} if the \texttt{\hyperlinks} command is not used, then \texttt{\pdfborder} is defined. User can define it in order to create colored frames around active links. For example \texttt{\pdfborder{1 0 0}} causes red frames in TOC (not printed, only visible in PDF viewers).

\texttt{\hyperlinks{\texttt{\urlcolor}}\% preloaded in format}

\texttt{\hyperlinks{\texttt{\urlcolor}}\% preloaded in format}

\texttt{\hyperlinks{\texttt{\urlcolor}}\% 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.

2.23 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⟩}{⟨pageno⟩}{⟨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
- ⟨pageno⟩: the page number numbered from 1 independently of pagination
- ⟨pageno⟩: the page number used in the pagination

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

We read the ⟨title⟩ parameter by \scantoeol from .ref file because the ⟨title⟩ can include something like ‘\{’.

\_def\tocline{}  \_def\_tocr#1#2#{\_ifnum#1=0 \\_ischaptrue \_fi\_addto\_toclist{\_tocline{#1}{#2}{#3}{}{}{\_scantoeol \_Xtoc}}\_addto\_toclist{\_tocline{#1}{#2}{#3}{\_scantoeol \_Xtoc}}\_addto\_toclist{\_tocline{#1}{#2}{#3}{\_scantoeol \_Xtoc}}\_addto\_toclist{\_tocline{#1}{#2}{#3}{\_scantoeol \_Xtoc}}
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 text text
text text text text text text text text text text text text text text text text text text text ................. <pageno>

Margins given by \_leftskip and \_rightskip are denoted by | in the examle above. \_tocrefnum is global counter of all TOC records (used by hyperlinks).

You can re-define default macros for each level of tocline if you want. Parameters are \{⟨number⟩\}\{⟨title⟩\}\{⟨pageno⟩\}.

The auxiliary macros are:

• \_llaptoclink{text} does \_noindent\llap{⟨linked text⟩}.
• \_tocdotfill creates dots in the TOC.
• \_nofirst\macro applies the \macro only if we don’t print the first record of the TOC.
• \_pgn{⟨pageno⟩} creates ⟨pageno⟩ as link to real ⟨gpage⟩ 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.
2.24 PDF outlines

2.24.1 Nesting PDF outlines

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

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

Each outline is mapped to the control sequence of the type \_ol:<num> or \_ol:<num>:<num> or \_ol:<num>:<num>:<num> 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 \_toclist at given level. When the first step is processed then (above that) the \_ol:... sequence of the parent increases its value too. The \_ol:... sequences are implemented by \_ol:<_count0:<_count1:<_count2> etc. For example, when section (level 2) is processed in the first step then we do:

\begin{verbatim}
\advance \count2 by 1
% increases the mapping pointer of the type
% \_ol:<_count0:<_count1:<_count2> of this section
\advance \_ol:<_count0:<_count1:<_count2> by 1
% increases the number of descendants connected
% to the parent 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 {⟨text⟩} inserts one outline with zero direct descendants. It creates link destination of the type oul:<num> into the document (where \insertoutline is used) and the link itself is created too in the outline.
\end{verbatim}
2.24.2 Strings in PDF outlines

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

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

\376\377\000C\000v\000i\001\015\000e\000n\000\355\000\040\000j\000\040\000z\000\341t\001\033\001\176

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

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

```
\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 ..$x$.. to ..x2.., 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.
```
The \_prepinverb\{macro\}\{separator\}\{\text\} does, e.g., \_prepinverb\{aaa |bbb| cccc |dd| ee\} does \_def\tmpb{\{aaa \}bbb{\{ cccc \}dd{\{ ee\}}} where \{su\} is \scantextokens\unexpanded. It means that in-line verbatim are not argument of \scantextokens. First \_def\tmpb tokenizes again the \text but not the parts which were in the the in-line verbatim.

The \regmacro is used in order to sed the values of macros \em, \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 gets the \textit{⟨title⟩} text in its parameter. The common recommendations for these macros are:

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

- Use \_printrefnum[⟨pre⟩/⟨post⟩] in horizontal mode. It prints ⟨pre⟩⟨ref-num⟩⟨post⟩. The ⟨ref-num⟩ is \_thechapnum or \_thesecnum or \_theseccnum depending on what type 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 \nbpar for closing the paragraph for printing title. This command inserts \_nobreak between each line of such paragraph, so the title cannot be broken to more pages.

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

\begin{verbatim}
72 \_def \_printchap #1\{\_vfill\}\_supereject
73 \_vglue\medskipamount % shifted by topkip\medskipamount
74 \_chapfont \_noindent \_mtext(chap) \_printrefnum[8]\_par
75 \_nobreak \_smallskip
76 \_noindent \_raggedright \_nbpar\_mark\%
77 \_nobreak \_belowtitle{\_bigskip}\%
78 \_firstnoindent
79 }
80 \_def \_printsec#1\{\_par
81 \_abovetitle{\_penalty-400}\_bigskip
82 \_secfont \_noindent \_raggedright \_printrefnum[8]\_quad\_nbpar\_insertmark\%
83 \_nobreak \_belowtitle{\_medskip}\%
84 \_firstnoindent
85 }
86 \_def \_printsecc#1\{\_par
87 \_abovetitle{\_penalty-200}\_medskip\_smallskip
88 \_secfont \_noindent \_raggedright \_printrefnum[0]\_quad\_nbpar\%
89 \_nobreak \_belowtitle{\_medskip}\%
90 \_firstnoindent
91 }
\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 seccc\_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}.  

111
Default concept: Tables, figures and display maths are numbered from one in each section – subsections doesn’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 \_other\_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⟩ and subsections have numbers ⟨\num⟩.⟨\num⟩.⟨\num⟩.

sections.opm

\newcount \chapnum % chapters
\newcount \seccnum % sections
\newcount \seccnum % subsections
\newcount \tnum % table numbers
\newcount \fnum % figure numbers
\newcount \dnum % numbered display maths
\newcount \chapx
\newcount \seccx
\newcount \seccx
\newcount \tnum
\newcount \fnum
\newcount \dnum
\def \chapx {
  \secx \seccnum=0 \lfnotenum=0 }
\def \secx {
  \seccx \seccnum=0 \tnum=0 \dnum=0 \resetABCDE }
\def \seccx {}
\def \thecapnum {
  \thechapnum}
\def \theseccnum {
  \othechapnum.\thechapnum.\theseccnum}
\def \thetnum {
  \othechapnum.\thechapnum.\thechapnum.\thetnum}
\def \thefnum {
  \othechapnum.\thechapnum.\thechapnum.\thefnum}
\def \thednum {
  \thechapnum.\theseccnum.\thetnum}
\def \notoc {
  \global \notoctrue}
\def \nonum {
  \global \nonumtrue}
\def \resetnonumnotoc{
  \global \notoctrue \global \nonumtrue}
\def \othe #1.{
  \ifnum#1>0 \the#1. \fi}
\def \incr #1{
  \global \advance#1by1 }

\def \othe #1.{
  \ifnum#1>0 \the#1. \fi}
\def \incr #1{
  \global \advance#1by1 }

The \notoc and \nonum prefixes are implemented by internal \_ifnotoc and \_ifnonum. They are reset after each chapter/section/subsection by the \_resetnonumnotoc macro.

sections.opm

\newifi \_ifnotoc \_notoctrue \_def \_notoc {
  \global \notoctrue}
\newifi \_ifnonum \_nonumtrue \_def \_nonum {
  \global \nonumtrue}
\_def \resetnonumnotoc{
  \global \notoctrue \global \nonumtrue}
\_public \notoc \nonum ;

The \chap, \sec and \secc macros are implemented here. The \_inchap, \_insec and \_insecc macros does the real work. First, we read the optional parameter \{⟨label⟩⟩, if it exists. The \chap, \sec and \secc macro reads its parameter using \scantoeol. This causes that they cannot be used inside other macros. Use \_inchap, \_insec and \_insecc macros directly in such case.

sections.opm

\optdef \ chap\[ \{ \trylabel \scantoeol \inchap \}\]
\optdef \ sec\[ \{ \trylabel \scantoeol \insec \}\]
\optdef \ secc\[ \{ \trylabel \scantoeol \insecc \}\]
\def \_trylabel\{\_istoksempty\_opt\_iffalse \_label\[\_the\_opt\]\_fi\}
\_def \_inchap #1{\_par \_sectionlevel=1
\_def \_savedtitle {#1}% saved to .ref file
\_ifnonum \_else \_globaldefs=1 \_incr\chapnum \_chapx\_fi
\_edef \_therefnum {\_ifnonum \_space \_else \thechapnum \_fi}\_printchap\{_scantextokens{#1}\}_resetnonumnotoc
\_def \_insec #1{\_par \_sectionlevel=2
\_def \_savedtitle {#1}% saved to .ref file
\_ifnonum \_else \_globaldefs=1 \_incr\seccnum \_seccx\_fi
\_edef \_therefnum {\_ifnonum \_space \_else \theseccnum \_fi}\_printsec\{_scantextokens{#1}\}_resetnonumnotoc
\_def \_insecc #1{\_par \_sectionlevel=3
\_def \_savedtitle {#1}% saved to .ref file
\_ifnonum \_else \_globaldefs=1 \_incr\seccnum \_seccx\_fi
\_edef \_therefnum {\_ifnonum \_space \_else \theseccnum \_fi}\_printsecc\{_scantextokens{#1}\}_resetnonumnotoc

112
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 `\detokenize`d before `\wref`, so the problem of “fragile macros” from old \TeX never occurs.

```latex
\begin{verbatim}
\def \printrefnum [#1@#2] {\leavevmode % we must be in horizontal mode
  \ifnonum \else #1\therefnum #2\fi
  \wlabel \therefnum % references, if \label<\label>
  \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}}
\end{verbatim}
```

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

```latex
\begin{verbatim}
\newskip \savedtitleskip
\newskip \savedlastskip
\def \abovetitle #1#2 {\savedlastskip=\lastskip % <whatever-skip>
  \ifdim \lastskip>0pt \vskip-\lastskip \fi
  \ifnum \lastpenalty=11333 \vskip-\savedtitleskip \else #1\fi
  \ifdim \savedlastskip>\tmpdim \vskip \savedlastskip \else #2\fi
}
\def \belowtitle #1 {#1 \global \savedtitleskip=\lastskip \penalty11333}
\end{verbatim}
```

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

```latex
\begin{verbatim}
\def \nbpar {\{\interlinepenalty=10000\endgraf}}
\end{verbatim}
```

The `\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`.

```latex
\begin{verbatim}
\def \firstnoindent {\global \everypar=\wipeepar \setbox7=\lastbox}
\end{verbatim}
```

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

The `\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:

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

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

The \cskip macro inserts nobreakable vertical space between caption and the object.

```latex
\def\caption/#1{\def\tmpa{#1}\nospaceafter \capA} \\
\optdef\capA []{\trylabel \incaption} \\
\def\incaption {\bgroup \\
\ifcsname _\tmpa num\_endcsname \ea\incr \csname _\tmpa num\_endcsname \\
\else \opwarning{Unknown caption /\tmpa}\fi \\
\edef\thecapnum {\csname _the\tmpa num\_endcsname} \edef\thecaptitle{\mtext{\tmpa}} \\
\ifcsname _everycaption\tmpa\_endcsname \\
\ea\the \csname _everycaption\tmpa\_endcsname \fi \\
\parfillskip=0pt plus2fil \\
\let\par=\par \\
\cs{\printcaption\tmpa} \\
} \\
\def\cskip {\par\nobreak\medskip} % space between caption and the object
```

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

```latex
\def \thecaptitle \thecapnum {\bf \thecaptitle~\thecapnum} \enspace \\
```

The \eqmark is processed in display mode (we add \eqno primitive) or in internal mode when \eqalignno is used (we don’t add \eqno).

```latex
\def\eqmark []{\trylabel \ineqmark} \\
def \ineqmark{\incr \dnum} \\
\ifinner \else \eqno \_fi \\
\wlabel\thednum \thednum \\
} \\
```

The \numberedpar{⟨letter⟩}{⟨name⟩} is implemented here.
The \_printnumberedpar \theXnum \{⟨name⟩\} opens numbered paragraph and prints it. The optional parameter is in \_the\_opt. You can re-define it if you need another design. \_printnumberedpar needs not to be re-defined if you only want to print Theorems in italic and to insert vertical skips (for example). You can do this by the following code:

```
\def\theorem {\medskip\bgroup\it \numberedpar A\{Theorem\}}
\def\endtheorem {\par\egroup\medskip}
\theorem Let $M$ be... \endtheorem
```

2.26 Lists, items

\_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

```
\def\_aboveliskip {\removelastskip  \penalty-100  \vskip\listskipamount}
\def\_belowliskip {\penalty-200 \vskip\listskipamount}
\def\_interliskip {}\def\_listskipA {\medskipamount}
\def\_listskipB {0pt plus.5\smallskipamount}
\def\_setlistskip {\ifnum \_ilevel = 1 \listskipamount = \_listskipA \relax
\else \listskipamount = \_listskipB \relax
\fi}
```

The \itemnum is locally reset to zero in each group declared by \begitems. So nested lists are numbered independently. User can set initial value of \itemnum to another value after \beitems if he/she want. Each level of nested lists is indented by new \_iindent from left. Default item mark is \_printitem. The \begitems runs \_aboveliskip only if we are not near below a title, where a vertical skip is placed already and where the \penalty11333 is. It activates * and defines it as \_startitem. The \enditems runs \_isnextchar\_par\{\_noindent\} thus the next paragraph is without indentation if there is no empty line between the list and this paragraph (it is similar behavior as after display math).
Various item marks are saved in \_item:⟨letter⟩ macros. You can re-define them or define more such macros. The \_printitem={\_item:⟨letter⟩} does \_printitem=\defaultitem first, then \_printitem={\_item:⟨letter⟩} when it is used and finally, \_startitem alias * uses \_printitem.

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

2.27 Verbatim, listings

2.27.1 Inline and “display” verbatim
\texttt{\code{⟨text⟩}} expands to \texttt{\detokenize{⟨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{⟨space⟩}} (followed by space in its csname). This macro does the real work.

The \texttt{\printverbatim{⟨text⟩}} macro is used for \texttt{\code{⟨text⟩}} printing and for \texttt{\langle (text) \rangle printing. It is defined as \texttt{\box}, so the in-verbatim \texttt{⟨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 \texttt{-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 \texttt{⟨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 \texttt{⟨text⟩} by \texttt{\printverbatim} and closes group. Suppose that \texttt{\activettchar\"} is used. Then the following work is schematically done:

\texttt{\def\"{\begingroup \_setverb \_readverb}}
\texttt{\_def \_readverb #1\"{\printverbatim{#1}\_endgroup}}

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{\eoldef}, 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{\textasciitilde\textasciitilde\textasciitilde 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:

\texttt{\def\_startverb \_endtt \_2\textasciitilde\textasciitilde J\{\ldots\}}
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 `\endtt` and end of the same line and it is simply ignored.

The `\_startverb` puts the scanned data to `\_prepareverbdata`. It sets the data to `\_tmpb` without changes by default, but you should re-define it in order to do special changes, if you want. (For example, `\hisyntax` redefines this macro.) The scanned data have `\^\^J` at each end of line and all spaces are active characters (defined as `\␣`). Other characters have normal category 11 or 12.

When `\_prepareverbdata` finishes then `\_startverb` runs `\_printverb` loop over each line of the data and does a final work: last skip plus `\noindent` in the next paragraph.

The `\_printverb` macro calls `\_printverbline{⟨line⟩}` repeatedly to each scanned line of verbatim text. The `\_printverb` is used from `\begtt...\endtt` and from `\verbinput` too.

The `\_testcommentchars` replaces the following `\_iftrue` to `\_iffalse` by default unless the `\commentchars` are set. So, main body of the loop is written in `\_else` part of the `\_iftrue` condition. The `\_printverbline{⟨line⟩}` is called here.

The `\_printverbline{⟨line⟩}` expects that it starts in vertical mode and it must do `\par` in order to return the vertical mode. The `\_printverblinenum` is used here: it does nothing when `\_ttline<0` else it prints the line number using `\_llap`.

Macro `\verbinput` uses a file read previously or opens the given file. Then it runs the parameter scanning by `\viscanparameter` and `\viscannuminus`. 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 `\vidolines=` the number of lines we need to print. Similar preparation is done as in `\begtt` after the group is opened. Then we skip `\vinolines`
lines in a loop a and we read \_vidolines lines. The read data is accumulated into \_tmpb macro. The next steps are equal to the steps done in \_startverb macro: data are processed via \_prepareverbdata and printed via \_printverb loop.

\verbatim
\_def \verbinput #1(#2) #3 {
\_par \def \tmpa(#3) \%
\_def \tmpb(#1) \%
cmds used in local group
\_ifa \_vifilename \_tmpa \else
\_openin \_vifile ={#3} \%
\_global \_viline=0 \_global \_let \_vifilename\_tmpa
\_if \_eof \_vifile
\_opwarning {\_noexpand \verbinput - file "#3" is unable to reading}
\_ea \ea \ea \ea \skiptorelax
\_fi
\_fi
\_viscanparameter #2+\_relax
\_def \skiptorelax#1 \_relax{}
\_def \viscanparameter #1+#2 \_relax{
\_if$#2$ \_viscanminus(#1) \_else \_viscanplus(#1+#2) \_fi
\_def \viscanplus(#1+#2+){
\_if$#1$ \_tmpnum=\_viline
\_else \_ifnum #1<0 \_tmpnum=\_viline \_advance \_tmpnum by-#1
\_else \_tmpnum=#1 \_advance \_tmpnum by-1
\_ifnum \_tmpnum<0 \_tmpnum=0 \_fi % (0+13) = (1+13)
\_fi \_fi
\_edef \vinolines{\_the \_tmpnum} \%
\_edef \vidolines{0} \_else \edef \vidolines(#2) \_fi
\_doverbinput
\_def \doverbinput{\_tmpnum=\_vinolines \_advance \_tmpnum by-\_viline
\_ifnum \_tmpnum<\_vinolines \_space \_fi
\_edef \ttlinesave{\_ttline=\_the \_ttline} \%
\_ifnum \_ttline=-1 \_ttline=\_viline \_fi
\_tmpnum=0 \edef \tmpb{} \%
\_ifnum \_vidolines=0 \_tmpnum=-1 \_fi
\_ifeof \_vifile \_tmpnum=\_vidolines \_space \_fi
\_loop \_ifnum \_tmpnum<\_vidolines \_space \_fi
\_vireadline \_advance \_tmpnum by \_space \_repeat \_skip lines
\_edef \ttlinesave{\_ttline=\_the \_ttline} \%
\_ifnum \_ttline=-1 \_ttline=\_viline \_fi
\_tmpnum=0 \edef \tmpb{} \%
\_ifnum \_vidolines=0 \_tmpnum=-1 \_fi
\_loop \_ifnum \_tmpnum<\_vidolines \_space \_fi
\_vireadline
\_ifnum \_vidolines=0 \_else \_advance \_tmpnum by1 \_fi
\_fi
\_endlinechar=`\_tempkey \_tmpnum=0 \_def \tmpb{} \%
\_ifnum \_vidolines=0 \_else \edef \vidolines(#2) \_fi
\_doverbinput
\_def \doverbinput{\_ttlinesave{\_ttline=\_the \_ttline} \%
\_ifnum \_ttline=-1 \_ttline=\_viline \_fi
\_tmpnum=0 \edef \tmpb{} \%
\_ifnum \_vidolines=0 \_tmpnum=-1 \_fi
\_loop \_ifnum \_tmpnum<\_vidolines \_space \_fi
\_vireadline \_advance \_tmpnum by \_space \_repeat \_skip lines
\_edef \ttlinesave{\_ttline=\_the \_ttline} \%
\_ifnum \_ttline=-1 \_ttline=\_viline \_fi
\_tmpnum=0 \edef \tmpb{} \%
\_ifnum \_vidolines=0 \_tmpnum=-1 \_fi
\_loop \_ifnum \_tmpnum<\_vidolines \_space \_fi
\_vireadline
\_ifnum \_vidolines=0 \_else \_advance \_tmpnum by \_space \_fi

\_verbatim
If the language of your code printed by `\verbinput` supports the format of comments started by two characters from 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 2.38 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 to set the catcode to 12 (and space to 13). Examples:

```latex
\commentchars // % C++ comments
\commentchars -- % Lua comments
(\catcode"\%=12 \catcode"\%=9 % used in \commentchars comments
(\catcode"\%=12 \catcode":"=13 \catcode=10 \catcodespace \ea\verbinput \commentchars %% % TeX comments
(\catcode"\#=12 \catcode=13 \catcodespace \ea\verbinput \commentchars#{} % bash comments
```

There is one limitation when \TeX interprets the comments declared by `\commentchars`. Each block of comments are 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, `\_reloding \_rm` initializes normal font, `\catcodetable0` returns to normal catcode table used before `\verbinput` is started, and the text accumulated in `\_vcomments` must be printed by `\_scantextokens` primitive.
The \visiblesp sets spaces as visible characters \char9251. It redefines the \_dsp, so it is useful for verbatim modes only.

The \_dsp is equivalent to \char92 primitive. It is used in all verbatim environments: spaces are active and defined as \_dsp here.

verbatim.opm

\def \_visiblesp{}{\_ifx\_initunifonts\relax\_def\_dsp{\char9251}\else\_def\_dsp{\char32}\_fi}
\let\_dsp= \% primitive "direct space"

2.27.2 Listings with syntax highlighting

The user can write

\begtt \hisynax{C}
...
\endtt

and the code is colorized by C syntax. The user can write \everytt=\hisynax{C} and all verbatim listings are colorized.

The \hisynax{name} reads the file hisyntax-\langle name\rangle.opm where the colorization is declared. The parameter \langle name\rangle is case insensitive and the file name must include it in lowercase letters. For example the file hisyntax-c.opm looks like:

hisyntax-c.opm
OpT\TeX\ provides \texttt{hisyntax-{c,python,\textit{tex,html}}.opm} files. You can take inspiration from these files and declare more languages.

User can re-declare colors by \texttt{\_hicolors\{..\}}. This value has precedence before \texttt{\_hicolors\{name\}} values declared in the \texttt{hicolors-\{name\}.opm} file. What exactly to do: copy \texttt{\_hicolors\{name\}=\{..\}} from \texttt{hicolors-\{name\}.opm} to your document, rename it as \texttt{\_hicolors=\{..\}} and do you own colors modifications.

Another way to set non-default colors is to declare \texttt{\newtoks\_hicolors\{name\}} (without the \_ prefix) and set the colors palette here. It has precedence before \texttt{\_hicolors\{name\}} (with the \_ 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 \texttt{\_hisyntax(name)} tokens string where the action over verbatim text is declared typically by \texttt{\_replfromto} or \texttt{\_replthis} macros.

The verbatim text is prepared by \texttt{pre-processing phase}, then the \texttt{\_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 \texttt{\n\n\n}, so \texttt{\n\(word\)\n} should be a pattern to finding whole words (no subwords). The \texttt{\n} control sequence is removed in the post-processing phase.
- Each end of line is represented by \texttt{\n\n\n\n}.
- The \texttt{\_start} control sequence is added before the verbatim text and \texttt{\_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.

- \texttt{\n} means noting but should be used as a boundary of words as mentioned above.
- \texttt{\t} means a tabulator. It is prepared as \texttt{\n\n\n\t\n} because it can be at the boundary of a word.
- \texttt{\x \{letter\}\{\{text\}\}} can be used as replacing text. Suppose the example

\begin{verbatim}
\_replfromto/*\*/\{\x C/{\#*1*/}\}
\end{verbatim}

This replaces all C comments \texttt{/*...*/} by \texttt{\x C/{\#*...*/}}. But the C comments may span more lines, i.e. the \texttt{^^J} should be inside it.

The macro \texttt{\x \{letter\}\{\{}text\}\}} is replaced by one or more \texttt{\x \{letter\}\{\{}text\}\}} in post-processing phase where each parameter \texttt{\{text\}} of \texttt{\x} keeps inside one line. Inside-line parameters are represented by \texttt{\x C\{}text\}} and they are replaced to \texttt{\x C\{}text\}} without any change. But:

\begin{verbatim}
\x C\{}text1\}\"J\{}text2\}\"J\{}text3\}\}
\_start\ is replaced by
\x C\{}text1\}\"J\x C\{}text2\}\"J\x C\{}text3\}\}
\end{verbatim}

The \texttt{\x \{letter\}\{\{}text\}\}} is expanded to \texttt{\x \{letter\}\{\{}text\}\}} and if \texttt{\_hicolor \{letter\} \{color\}} is declared then \texttt{\x \{letter\}\{\{}text\}\}} expands to \texttt{\{color\}\{text\}\}}. So, required color is activated at all lines (separately) where C comment spans.

- \texttt{\y \{}text\}\}} is replaced by \texttt{\y \{}text\}\}} in the post-processing phase. It should be used for macros without a parameter. You cannot use unprotected macros as replacement text before the post-processing phase, because the post-processing phase is based on expansion whole verbatim text.

The following macros \texttt{\_replfromto} and \texttt{\_replthis} manipulate with the verbatim text which has been read already and stored in the \texttt{\_tmpb} macro.

The \texttt{\_replfromto \{from\}\{}\{to\}\{\{}what\}\}} finds first \texttt{\{from\}} then the first \texttt{\{to\}} following by \texttt{\{from\}} pattern and the \texttt{\{}text\}\} between them is packed to \texttt{\#1}. Then \texttt{\{from\}\{}text\}\{\} is replaced by \texttt{\{what\}}. The \texttt{\{}what\}\} parameter can use \texttt{\#1} which is replaced by the \texttt{\{}text\}\}.

The \texttt{\_replfromto} continues by finding next \texttt{\{from\}}, then, next \texttt{\{to\}} repeatedly over the whole verbatim text. If the verbatim text is ended by opened \texttt{\{from\}} but not closing by \texttt{\{to\}} then \texttt{\{to\}} is appended to the verbatim text automatically and the last part of verbatim text is replaced too.

\texttt{s \_codeincl \texttt{hisyntax} \{Syntax highlithing of verbatim listings <2020-04-04>\}} \% preloaded in \texttt{format} hi-syntax.opm

The following macros \texttt{\_replfromto} and \texttt{\_replthis} manipulate with the verbatim text which has been read already and stored in the \texttt{\_tmpb} macro.
First two parameters are expanded before usage of \replfromto. You can use \csstring\% or something else here.

\replfromto \#1\#2\replfromtoE \#1\#2\% \#1=from \#2=to \#3=what to replace
\_if\_end\#1\_ea\_replstop \_else \_afterfi{\_replto\#2}\_fi%
\_edef\_tmpa{\#1\#2}\_ea\_replfromtoE \_tmpa
\_def\_replfromto\#1\#2\#3{% #1=from #2=to #3=what to replace
\_edef\_tmpb{\#1\#2}\_ea\_replfromtoE \_tmpb\#3}
\_edef\_tmpa{\#1\#2\#3}\_ea\_replfromtoE \_tmpa
\_def\_replfrom\##1\#1\#2\{\_addto\_tmpb{\##1}\_fi%\_afterfi{\_replto\##2}\_fi}%
\_def\_replto\##1\#2\#3{\_if\_end\#3\_afterfi{\_replfin\##1}\_else\_addto\_tmpb{\#3}\_fi}%
\_def\_replfin\##1\#1\_end{\_addto\_tmpb{\#3}\_replstop}
\_edef\_tmpb{\_ea}\_ea\_replfrom\_tmpb\#1\_end\#2\_end\_end\_relax
\_edef\_tmpb{\_ea}\_ea\_replfromto\#1\_end\#2\_end\_end\_relax
\_def\_replstop\#1\_end\_relax{}


\_ifcsname _hicolors\_tmpa\_endcsname
\_global\_cs(_hicolors\_tmpa)=\_hicolors \_global\_hicolors={}
\_fi\_fi\_fi

\_ea\_the \_csname _hisyntax\_tmpa\_endcsname \_the\_hisyntax<name>
\_else\_opwarning{Syntax highlighting "\_tmpa" undeclared (no file hisyntax-\_tmpa.opm)}
\_fi\_fi

\_replthis{\_start
^^J}{}\_replthis{^^J\_end}{^^J}
\_def\{\_hskip \_dimexpr\_tabspaces em/2\_relax}

\_localcolor
}}
\_public \hisyntax \hicolor ;

Alias for languages can be declared like this. When \hisyntax{xml} is used then this is the same as \hisyntax{html}.

\sdef{\_hialias:xml}{html}
\sdef{\_hialias:json}{c}

2.28 Graphics

The \inspic is defined by \pdfximage and \pdfrefximage primitives. If you want to use one picture more than once in your document, then the following code is recommended:

\newbox\mypic
\setbox\mypic = \hbox{\picw=3cm \inspic{⟨picture⟩}}

My picture: \copy\mypic, again my picture: \copy\mypic, etc.

This code downloads the picture data to the 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}.

\inspic accepts old syntax \inspic {⟨filename⟩⟨space⟩} or new syntax \inspic {⟨filename⟩}. So, we need to define two auxiliary macros \_inspicA and \_inspicB.

You can include more \pdfximage parameters (like page⟨number⟩) in the \_picparams macro.

All \inspic macros are surrounded in \hbox in order user can write \moveright\inspic ... or something similar.

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 \inkinspic 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.
\public \inkinspic ;

\pdfscale\{x-scale\}\{y-scale\} and \pdfrotate\{degrees\} macros are implemented by \pdfsetmatrix primitive. We need to know values of sin, cos function in the \pdfrotate. We use Lua code for this.

\public \inkinspic ;

The \transformbox\{\text{transformation}\}\{\text{text}\} is copied from OPmac trick 0046. The \rotbox\{\text{degrees}\}\{\text{text}\} is a combination of \rotsimple from OPmac trick 0101 and the \transformbox. Note, that \rotbox\{-90\} puts the rotated text to the height of the outer box (depth is zero) because code from \rotsimple is processed. But \rotbox\{-90.0\} puts the rotated text to the depth of the outer box (height is zero) because \transformbox is processed.
\setbox0=\hbox{\_box0}\_ht0=0pt \_dp0=0pt
\pdfsave#1\_rlap{\_box0}\pdfrestore
\kern\newRt}
\def\preptransform #1{
\def\currmatrix{1 0 0 1 }
\def\pdfsetmatrix##1{\edef\tmpb{##1}\ea\multiplyMxM\tmpb\unskip}
\let\pdfsetmatrix=\pdfsetmatrix #1%
\setnewHtDp 0pt \_ht0 \setnewHtDp 0pt -\_dp0
\setnewHtDp \_wd0 \_ht0 \setnewHtDp \_wd0 -\_dp0
\protected\def\pdfsetmatrix {\pdfextension setmatrix}
\let\pdfsetmatrix=\pdfsetmatrix
\def\setnewHtDp #1 #2 {\vvalX=#1\relax \vvalY=#2\relax \ea\multiplyMxV\currmatrix
\ifdim\vvalX<\newLt \newLt=\vvalX \fi \ifdim\vvalX>\newRt \newRt=\vvalX \fi
\ifdim\vvalY>\newHt \newHt=\vvalY \fi \ifdim-\vvalY>\newDp \newDp=-\vvalY \fi}
\def\rotbox#1#2{\isequal{90}{#1}\iftrue \rotboxA{#1}{\kern\ht0 \tmpdim=\dp0}{\vfill}{#2}\else \isequal{-90}{#1}\iftrue \rotboxA{#1}{\kern\dp0 \tmpdim=\ht0}{\vfil}{#2}\else \transformbox{\pdfrotate{#1}}{#2}\fi \fi}
\def\rotboxA #1#2#3#4{\hbox{\setbox0=\hbox{{#4}}#2 \vbox to\wd0{#3 \wd0=0pt \_dp0=0pt \_ht0=0pt \pdfsave\pdfrotate{#1}\box0\pdfrestore\vfil}\kern\tmpdim}}
\public\transformbox \rotbox
\directlua{tex.print(string.format('{\_pcent d}{\_pcent d}',token.scan_dimen(),token.scan_dimen()))}
\def\puttext\{\ea\ea\ea\puttextA\scantwodimens\}
\def\puttextA#1#2#3{\setbox0=\hbox{\_box0=#1\ea\_box0=#2}\_dimen1=\_dimen2=\_dimen3=\_dimen4=\_puttextB}
\def\puttextB{\ifvmode \ifdim\prevdepth>0pt \vskip-\prevdepth \relax \fi \nointerlineskip \fi \wd0=0pt \_ht0=0pt \_dp0=0pt \_vbox to\wd0{\_box0=0pt \_box0=0pt}[\_kern]\dimen2=\_dimen1=\_box0=0pt \_hss}\_vss}}
\def\putpic\{\ea\ea\ea\putpicA\scantwodimens\}
\def\putpicA#1#2#3#4#5#6{\setbox0=\hbox{\_box0=#4\_box0=#5}\_picwidth=#1sp \_picheight=#2sp \inspic{#3}}
\newbox\bgbox
\def\backgroundpic\{\ea\ea\ea\backgroundpicA\scantwodimens\}
\def\backgroundpicA#1#2#3#4#5#6#7{\setbox0=\hbox{\_box0=#4\_box0=#5}\_picwidth=#1sp \_picheight=#2sp \inspic{#3}}
\public\puttext\putpic \backgroundpic
\scantwodimens scans two objects with the syntactic rule \(\text{dimen}\) and returns \(\{\text{number}\}\{\text{number}\}\) in sp unit.
\puttext \(\{\text{right}\}\{\text{up}\}\{\text{text}\}\) puts the \text{text} to desired place: From current point moves \(\text{down}\) and \(\text{right}\), puts the \text{text} and returns back. The current point is unchanged after this macro ends.
\putpic \(\{\text{right}\}\{\text{up}\}\{\text{width}\}\{\text{height}\}\{\text{image-file}\}\) does \puttext with the image scaled to desired \(\text{width}\) and \(\text{height}\). If \(\text{width}\) or \(\text{height}\) is zero, natural dimension is used. The \nospec is a shortcut to such natural dimension.
\backgroundpic \{\text{image-file}\} puts the image to the background of each page. It is used in the slides style, for example.
\_circle\{(x)\(\times\)\(\langle y \rangle\)} creates an ellipse with \(\langle x \rangle\) axis and \(\langle y \rangle\) axix. The origin is in the center.
\_oval\{(x)\(\times\)\(\langle y \rangle\)\(\langle \text{roundness} \rangle\)} 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\{(x)\(\times\)\(\langle y \rangle\)\(\langle \text{curve} \rangle\)} moves current point to \(\langle x \rangle\), \(\langle y \rangle\), \(\langle \text{curve} \rangle\) and returns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.

The \inoval\{(text)\} is an example of \_oval usage. The \incircle\{(text)\} is an example of \_circle usage. The \ratio, \lwidth, \fcolor, \lcolor, \shadow and \overlapmargins are parameters, they can be set by user in optional brackets \[\ldots\]. For example \fcolor=\Red does \_let\_fcolorvalue=\Red and it means filling color. The \_setflcolor uses the \_fillstroke macro to separate filling color and drawing color.
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 beginning of the output routine. The modification of the output routine is done using \_insertshadowresources only once when the shadow effect is used first.
The \_doshadow\{\textit{curve}\} does the shadow effect.

A generic macro \_clipinpath\(\langle x \rangle \langle y \rangle \langle \text{curve} \rangle \langle \text{text} \rangle\) declares a clipping path by the \langle curve\rangle shifted by the \langle x \rangle, \langle y \rangle. The \langle text\rangle is typeset when such clipping path is active. Dimensions are given by bp without the unit here. The macros \_clipinoval\(\langle x \rangle \langle y \rangle \langle \text{width} \rangle \langle \text{height} \rangle\)\{\text{text}\} and \_clipincircle\(\langle x \rangle \langle y \rangle \langle \text{width} \rangle \langle \text{height} \rangle\)\{\text{text}\} are defined here. These macros read normal \TeX\ dimensions in their parameters.
2.29 The \texttt{table} macro, tables and rules

2.29.1 The boundary declarator:

The \texttt{⟨declaration⟩} part of \texttt{\{⟨declaration⟩\}⟨data⟩} includes column declarators (letters) and other material: the \texttt{l} or \texttt{(cmd)}. The boundaries of columns are just before each column declarator (with exception of the first one) if the boundary declarator \texttt{:} is not used. For example, the declaration \{l|l|c(xx)(yy)c\} should be written more exactly using the boundary declarator: by \{l|l|c(xx)(yy):c\}. But you can set these boundaries to another places using the boundary declarator \texttt{:} explicitly, for example \{l|tl|l|c(xx):(yy)c\}. The boundary declarator \texttt{:} can be used only once between each two column declarators.

Each table item have its own group. The \texttt{⟨cmd⟩} 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 \texttt{(setting)} followed by column declarator. But if such column is not first, you must use \texttt{:(setting)}. Example. We have three centered columns, the second one have to be in bold font and the third one have to be in red: \texttt{\table{c:(\bf)c:(\Red)c}{⟨data⟩}}

2.29.2 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 2.29.3.

By default, \texttt{\tabskip} is 0pt. It means that only \texttt{\tabiteml}, \texttt{\tabitemr} and \texttt{(cmds)} can generate visual spaces between columns. But they are not real spaces between columns because they are in fact the part of the total column width.

The \texttt{\tabskip} value declared before the \texttt{\table} macro (or in \texttt{\everytable} or in \texttt{\thistable}) is used between all columns of the table. This value is equal for all spaces between columns. But you can set each such space individually if you use \texttt{(\tabskip=⟨value⟩)} in the \texttt{⟨declaration⟩} immediately before boundary character. The boundary character represents the column pair for which the \texttt{\tabskip} have individual value. For example \texttt{c(\tabskip=5pt):r} gives \texttt{\tabskip} value between \texttt{c} and \texttt{r} columns. You need not to use boundary character explicitly, so \texttt{c(\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{\crlli} and \texttt{\crlp} have another behavior than you probably expected: they are interrupted in each \texttt{\tabskip} space.

2.29.3 Tables to given width

There are two possibilities how to create tables to given width:

- \texttt{\table to⟨size⟩}\{⟨declaration⟩\}⟨data⟩ 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⟨size⟩}\{⟨declaration⟩\}⟨data⟩ expands the columns declared by \texttt{p⟨⟨size⟩⟩}, if the \texttt{⟨size⟩} is given by a virtual \texttt{\tsize} unit. See example below.

Example of \texttt{\table to⟨size⟩}:
\texttt{\thistable{\tabskip=0pt plus1fil minus1fil}\table to\hsize {lr}{⟨data⟩}}

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⟨size⟩} (means “paragraphs expanded to”):
\texttt{\table pxto\hsize {l|p⟨tsize⟩}{crl}aaa & Ddkas jd dsjds ds cgha sfgs dd fddzf dfhz xxz dras ffg hkad kds d sdjds h sd jd dsjds ds cgha sfgs dd fddzf dfhz xxz. \crl} bb ddd ggg & Dsjds ds cgha sfgs dd fddzf dfhz xxz ddkas jd dsjds ds cgha sfgs dd fddzf. \crl}
The first column is variable width (it gets the width of most wide item) and the resting space to given \hsize is filled by the p column.

You can declare more than one p{⟨coefficient\rangle\textsize} columns in the table when \pxtowx keyword is used. The total sum of ⟨coefficinets⟩ must be exactly one. For example,

\table \pxtowx{13cm} {r p{.3\textsize} p{.5\textsize} p{.2\textsize} l}{{⟨data⟩}}

This gives the ratio of widths of individual paragraphs in the table.

### 2.29.4 \eqbox: boxes with equal width across the whole document

The \eqbox [⟨label⟩]⟨text⟩ behaves like \hbox⟨text⟩ in the first run of \TeX. But the widths of all boxes with the same label are saved to .ref file and the maximum box width for each label is calculated at the beginning of the next \TeX run. Then \eqbox [⟨label⟩]⟨text⟩ behaves like \hbox to ⟨dim:label⟩ \hspace ⟨text⟩\hspace, where ⟨dim:label⟩ is the maximum width of all boxes labeled by the same [⟨label⟩]. The documentation of the \LaTeX package eqparbox includes more information and tips.

The \eqboxsize [⟨label⟩]⟨dimen⟩ expands to ⟨dim:label⟩ if this value is known, else it expands to the given ⟨dimen⟩.

The optional parameter r or l can be written before [⟨label⟩] (for example \eqbox r[⟨label⟩]{⟨text⟩}) if you want to put the text to the right or to the left side of the box width.

Try the following example and watch what happens after first \TeX run and after second one.

\def\leftitem#1{\par
  \noindent \hangindent=\eqboxsize[items]{2em}\hangafter=1
  \eqbox r[items]{#1 }\ignorespaces}

\leftitem {bf first} \lorem[1]
\leftitem {bf second one} \lorem[2]
\leftitem {bf final} \lorem[3]

### 2.29.5 Implementation of the \table macro and friends

The result of the \table{⟨declaration⟩}{⟨data⟩} macro is inserted into \_tablebox. You can change default value if you want by \let\_tablebox=\vtop or \let\_tablebox=\relax.

We save the to⟨size⟩ or pxtowx⟨size⟩ to \_tablew and \_tableW sets the to⟨size⟩ to the \_tablew macro. If pxtowx⟨size⟩ is used then \_tablew is empty and \_tmpdim includes given ⟨size⟩. The \_ifpxtowx returns true in this case.

The \table continues by reading {⟨declaration⟩} in the \_tableA macro. Catcodes (for example the | character) have to be normal when reading \table parameters. This is the reason why we use \catcodetable here.

The \tablinespace is implemented by enlarging given \tabstrut by desired dimension (height and depth too) and by setting \_lineskip=-2\tablinespace. Normal table rows (where no \hrule is between them) have normal baseline distance.
The \_tableA{(declaration)} macro scans the \{declaration\} by \_scantabdata\#1\_relax and continues by reading \{data\} by the \_tableB macro.

The \_tableB{\{data\}} saves \{data\} to \_tmpb and does four \replstring\s to prefix each macro \_crcr (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 \_crcr 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

\begin{verbatim}
\tabskip=\tabskipl \halign{\langle converted declaration\rangle \tabskip=\tabskipr \cr \langle data\rangle \crcr}
\end{verbatim}

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 \_hsize first. Then the first attempt of the table is created in box 0. Then the \_tsize is re-calculated using \_wd0 and the real table is printed by \halign in the second pass.

If no \pxto keyword was used, then we print the table using \halign directly. The \_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 \langle converted declaration\rangle by \bgroup and we end it by \egroup in the \_tableC macro. Each & character is stored as \egroup&\bgroup in \langle converted declaration\rangle. The \_halign\_tablew\_tableC really does:

\begin{verbatim}
\halign\_tablew{\bgroup \langle converted declaration\rangle \egroup \tabskip=\tabskipr \cr \langle data\rangle \crcr}
\end{verbatim}

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=|cr||cl|.

\begin{verbatim}
tabdata: \_vrule\_the\_tabiteml\_hfil\_unsskip\_hfil\_the\_tabitemr\_tabstrutA \_vrule\_kern\_vvkern\_vrule\_tabstrutA
\end{verbatim}
The second result in the \ddlinedata macro is a template of one row of the table used by \crrl macro.

The \addtabitemx adds the boundary code (used between columns) to the \converted declaration. This code is \bgroup &\group \colnum=value\relax. You can get the current number of column from the \colnum register, but you cannot write \the\colnum as the first object in a \converted declaration because in the \halign the \colnum is processed after this. Use \relax\the\colnum instead. Or you can write:

\def\showcolnum{\ea\def\ea\totcolnum\ea{\the\colnum/\totcolnum}}
\table{ccc}{\showcolnum & \showcolnum & \showcolnum}

This example prints 1/3 2/3 3/3, because the value of the \colnum is equal to the total number of columns before left part of the column declaration is processed.

This code converts || or | from \table \converted declaration to the \converted declaration.

The default “declaration letters” c, l, r and p are declared by setting \tabdeclarec, \tabdeclarel, \tabdeclarer and \paramtabdeclarer 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\hideskip&.

The \fL, \fR, \fC and \fX 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 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 \vspan{(\text{number})}{\{\text{text}\}} implementaiton is here. We need to lower the box by 

\((\text{number}) - 1)* (\text{ht} + \text{dp} \ of \ \text{\tabstrut}) / 2.\)

The \#1 parameter must be one-digit number. If you want to set more digits then use braces.

The parameters of primitive \vrule and \hrule keeps the rule “last wins”. If we re-define \hrule to \_orihrule height1pt then each usage of redefined \hrule uses 1pt height if this parameter isn’t overwritten by another following height parameter. This principle is used for settings another default rule thickness than 0.4pt by the macro \rulewidth.

The \frame{\text{text}} uses “\vbox in \vtop” trick in order to keep the baseline of the internal text at the same level as outer baseline. User can write \frame{abcxyz} in normal paragraph line, for example and gets the expected result: abcxyz. The internal margins are set by \vvkern and \hhkern parameters.

\eqbox and \eqboxsize are implemented here. The widths of all \eqboxes are saved to the .ref file in the format \_Xeqbox{\langle \text{label} \rangle}{\langle \text{size} \rangle}. The .ref file is read again and maximum box width for each \langle \text{label} \rangle is saved to \_eqb: \langle \text{label} \rangle. 

2.30 Balanced multi-columns

This code is documented in detail in the “\TeXbook naruby”, pages 244–246, free available, \url{http://petr.olsak.net/tbn.html}, but in Czech. Roughly speaking, macros complete all material between
\begmulti⟨num-columns⟩ and \endmulti into one \vbox 6. Then the macro measures the amount of free space at the current page using \pagegoal and \pagetotal and does \vsplit of \vbox 6 to columns with 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 \clubppenalty (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
\advance\hsize by\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=0
\loop \ifnum\Ncols>\tmpnum
\advance\tmpnum by1
\setbox1=\hbox{\unhbox1 \vsplit6 to\dimen1 \hss}
\repeat
\hbox{}\nobreak\vskip-\splittopskip \nointerlineskip
\line{\unhbox1\unskip}
\dimen0=\dimen1 \divide\dimen0 by\baselineskip \multiply\dimen0 by\Ncols
\egroup}

\multiskip\egroup

\multicolumns.opm

\def\makecolumns{\bgroup % full page, destination height: \dimen1
\vbadness=20000 \setbox1=\hbox()\tmppnum=0
\loop \ifnum\Ncols>\tmppnum
\advance\tmppnum by1
\setbox1=\hbox{\unhbox1 \vsplit6 to\dimen1 \hss}
\repeat
\hbox{}\nobreak\vskip-\splittopskip \nointerlineskip
\line{\unhbox1\unskip}
\dimen0=\dimen1 \divide\dimen0 by\baselineskip \multiply\dimen0 by\Ncols
\egroup}

\def\splitpart{%
\makecolumns % full page
\vskip Opt plus 1fil minus\baselineskip \break
\ifnum\Ncols<\tmppnum\dimen0=\ht6 \else \dimen0=.8\maxdimen \fi
\divide\dimen0 by\Ncols \relax
\if\_balancecolumns\_flushcolumns \advance\dimen0 by-.5\vsize \fi
\divide\dimen1 by\baselineskip \corrsize{\dimen1}\dimen2=\dimen1
\divide\dimen2 by\baselineskip \corrsize{\dimen2}\dimen1
\divide\dimen1 by\baselineskip \corrsize{\dimen1}\dimen0=\dimen1
\divide\dimen0 by\Ncols \relax
\ifvoid6 \else
\ifdim\dimen0>\dimen2 \ea\ea\ea \splitpart
\else \_balancecolumns % last balancing
\fi
\fi

136
Final balancing of the columns.

\multicolumns{opm}

\def\balancecolumns{
  \setbox7=\copy6 % destination height: \dimen0
  \ifdim\dimen0>\baselineskip \else \dimen0=\baselineskip \fi
  \vbadness=20000
  \def\tmp{
    \setbox1=\hbox{} \tmpnum=0
    \loop \ifnum Ncols>\tmpnum
      \advance\tmpnum by1
      \setbox1=\hbox{\unhbox1
        \ifvoid6 \hbox to \wd6{\hss}
        \else \vsplit6\textwidth \dimen0
        \fi\hss}
    \repeat
    \ifvoid6 \else
      \advance\dimen0 by 0.2\baselineskip
    \fi
    \setbox6=\copy7
    \ea \tmp \fi
  }\tmp
  \hbox{} \nobreak \vskip-\splittopskip \nointerlineskip
  \hbox to \hsize{\unhbox1\unskip} %
  \egroup

\def\corrsize #1{%% #1 := #1 + \splittopskip - \topskip
  \advance #1 by \splittopskip \advance #1 by-\topskip
}

\public \begmulti \endmulti ;

2.31 Citations, bibliography

2.31.1 Macros for citations and bibliography preloaded in the format

cite-bib.opm

\codedecl \cite {Cite, Bibliography <2020-03-09>} % loaded in format

\newcount\bibnum % the bibitem counter
\newtoks\bibmark % the bibmark used if \nonumcitations
\newcount\lastcitenum \lastcitenum=0 % for \shortcitations
\public \bibnum \bibmark ;

\cite \{[\text{label}], [\text{label}], ..., [\text{label}]\} manages \langle labels\rangle using \_citeA and prints \langle\text{bib-marks}\rangle using \_printsavedcites.

\nocite \{[\text{label}], [\text{label}], ..., [\text{label}]\} only manages \langle labels\rangle but prints nothing.

\rcite \{[\text{label}],[\text{label}], ...,[\text{label}]\} behaves like \cite but prints \langle\text{bib-marks}\rangle without brackets.

\ecite \{[\text{label}],[\text{text}]\} behaves like \rcite \{[\text{label}]\} but prints \langle\text{text}\rangle instead \langle\text{bib-marks}\rangle. The \langle\text{text}\rangle is hyperlinked like \langle\text{bib-marks}\rangle when \cite or \rcite is used. The empty internal macro \_savedcites will include the \langle\text{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\text{bib-marks}\rangle because new group is opened.

\_printsavedcites will include the \langle\text{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\text{bib-marks}\rangle because new group is opened.

\_def\cite\[#1\]{{\_citeA\#1,,0\_printsavedcites}}
\_def\nocite\[#1\]{{\_citeA\#1,,0\_printsavedcites}}
\_def\rcite\[#1\]{{\_citeA\#1,,0\_ea\eciteB\_savedcites;}}
\_def\ecite\[#1\]{{\_bgroup\_citeA\#1,,0\_ea\eciteB\_savedcites;}}
\_def\_savedcites{}
\_public \cite \nocite \rcite \ecite ;

\langle\text{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\text{label}\rangle to \langle\text{bib-mark}\rangle is done when \bib or \usebib is processed. These macros store the information to \_Xbib{\langle\text{label}\rangle}{\langle\text{number}\rangle}{\langle\text{nonumber}\rangle} where \langle\text{number}\rangle and \langle\text{nonumber}\rangle are two variants of \langle\text{bib-mark}\rangle (numbered or text-like). This information is read from \_ref file and it is saved to macros \_bib:\langle\text{label}\rangle and \_bibm:\langle\text{number}\rangle. First one includes number and
\citeA{name}, processes one label from list of labels given in the parameter of \cite, \nocite, \rcite or \ecite macros. It adds the \langle label⟩ to global list \citelist which will be used by \usebib (it must to know what \langle 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 \langle label⟩ to \citelist twice, we distinguish four cases:

- \langle label⟩ was not declared by \Xbib and it is first such \langle label⟩ in the document: Then \bib:⟨label⟩ is undefined and we save label using \adddcitlist, write warning on the terminal and define \bib:⟨label⟩ as empty.

- \langle 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).

- \langle label⟩ was declared by \Xbib and it is first such \langle label⟩ in the document: Then \bin:⟨label⟩ includes \bibnn{⟨number⟩}& and we test this case by \if &\bibnn{⟨number⟩}. This is true when \bibnn{⟨number⟩} expands to empty. The \langle label⟩ is saved by \adddcitlist and \bib:⟨label⟩ is re-defined directly as \langle number⟩.

- \langle 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 \langle labels⟩.

\def\citeA#1#2,{\if#1,\else \ifcsname _bib:#1#2\endcsname \else
\addcitelist{#1#2}%,\opwarning{The cite [#1#2] unknown. Try to TeX me again},% \fi \fi}

The \langle bib-marks⟩ (in numeric or text form) are saved in \savedcites macro separated by commas. The \printsavedcites prints them by normal order or sorted if \sortcitations is specified or condensed if \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 \langle bib-marks⟩ printing follows (sorry, without detail documentation). They are documented in opmac-d.pdf (but only in Czech).
The \texttt{\bib{[\textit{label}]} \{\textit{optional bib-mark}\}} prints one bib-entry without reading any database. The bib-entry follows after this command. This command counts the used \texttt{bib} from one by \texttt{\bibnum} counter and saves \texttt{\_Xbib{⟨label⟩}{\_the\_bibnum}{\_the\_bibmark}} into .ref file immediately using \texttt{\_wbib}. This is the core of creation of mapping from \langle\textit{labels}\rangle to \langle\textit{bib-marks}\rangle.
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 \lhap\{⟨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.

\begin{lstlisting}[language=TeX]
\def \_printbib {\hangindent\_indent
\ifx \_citelinkA\empty \hskip \_indent \_lhap{⟨the\_bibnum⟩} \fi
\def \_bibskip {⟨if\_bibnum>0 \_smallskip \fi}
\end{lstlisting}

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 2.31.2, where the file usebib.opm is documented.

\begin{lstlisting}[language=TeX]
\def \usebib{\par \opinput {usebib.opm} \_usebib}
\def \_usebib{\par \opinput {usebib.opm} \_usebib}
\end{lstlisting}

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 \_citelinkA. 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 \_citelinkA concats two lists of ⟨labels⟩ from \_citelinkA and \_citelinkB and uses this concatenated list.

\begin{lstlisting}[language=TeX]
\def \_Xcite#1{\_addto \_citelistB{\_citeI[#1]}}
\def \_writeXcite#1{\_openref \_immediate \_wref \_Xcite{(#1)}}
\end{lstlisting}

2.31.2 The \usebib command

The file usebib.opm implements the command \usebib{⟨sorttype⟩ ⟨⟨style⟩⟩ ⟨⟨bibfiles⟩⟩} where ⟨sorttype⟩ is one letter c (references ordered by citation order in the text) or s (references ordered by key in the style file), ⟨⟨style⟩⟩ is the part of the name bib-{⟨style⟩}.opm of the style file and ⟨⟨bibfiles⟩⟩ are one or more .bib file names without suffix separated by comma without space. Example:

\usebib/s (simple) mybase,yourbase

This command reads the ⟨⟨bibfiles⟩⟩ directly and creates the list of bibliographic references (only those declared by \cite[] or \nocite[] in the text). The formatting of such references is defined in the style file. The usage is mentioned in user documentation too.

The principle “first entry wins” is used. Suppose \usebib/s (simple) local,global. If an entry with the same label is declared in local.bib and in global.bib too then the first wins. So, you can set an exceptions in your local.bib file for your document.

Notes for style writers

The bib-{⟨style⟩}.opm file must define the commands:

- \_authorname ... formatting of one name in the authors list. The macro can use the following data: \_NameCount (the number of currently processed author name in the list), 0\_namecount (the total number of the authors in the list), \_Lastname, \_Firstname, \_Von, \_Junior (the parts of the name). See the documentation of the librarian package for more info.
- \_editorname ... the same as \_authorname, but for editors list.
- \_print⟨⟨entrytype⟩⟩ (defined by \_sdef) for formatting the entry of ⟨⟨entrytype⟩⟩. The ⟨⟨entrytype⟩⟩ have to be lowercase. This command can use the command:
- \_bprinta [⟨fieldname⟩] {⟨if defined⟩} {⟨if not defined⟩}. The part ⟨if defined⟩ is executed if ⟨fieldname⟩ is declared in .bib file for the entry which is currently processed. Else the part ⟨if not defined⟩ is processed. The part ⟨if defined⟩ can include the * parameter which is replaced by the value of the ⟨fieldname⟩. The part ⟨if not defined⟩ can include the \bibwarning command if the ⟨fieldname⟩ is mandatory.
\_bprintb \([\text{fieldname}]\) \{\langle \text{if defined} \rangle \} \{\langle \text{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\#1 parameter for internal \_bprintb. But if \_bprinta commands are nested then the parameter is not duplicated.

\_pbprintc \{\langle \text{if non-empty} \rangle \}. The \langle \text{if non-empty} \rangle part is executed if \macro is non-empty. The \_p parameter can be used, it is replaced by the \macro.

\_bprintv \{\langle \text{field1} \rangle, \langle \text{field2} \rangle, \ldots \} \{\langle \text{if defined} \rangle \} \{\langle \text{if not defined} \rangle \}. The part \langle \text{if defined} \rangle is executed if \langle \text{field1} \rangle or \langle \text{field2} \rangle or ... is defined, else the second part \langle \text{if not defined} \rangle is executed. There is one filed name or the list field names separated by commas. The parts cannot include any parameter.

There are two special fieldnames: \texttt{!author} and \texttt{!editor}. The processed list of authors or editors (by repeatedly calling \_authorname or \_editorname) are used here instead of raw data.

You can define \_print:BEGIN and/or \_print:END which is executed at the begin or end of each \langle entrytype \rangle. The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by \_printbib macro used in OpTEX (and may be redefined by the author or document designer).

You can declare \_bimark={something} in the \_print:END macro. This bibmark is saved to the .ref file (created by OpTEX) and used in the next \TeX run as \texttt{cite} marks when \texttt{nonumcitations} is set.

The whole style file is read in the group during \_usebib command is executed before typesetting the reference list. Each definition or setting is local here.

If you are using non-standard fieldnames in .bib database and bib. style, you have to declare them by \_CreateField \{\langle fieldname \rangle\}.

You can declare \_SortingOrder in the manner documented by librarian package.

If your style adds some words or abbreviations you can make them multilingual by saying \_mtext\{\langle label \rangle \} instead such word and \_mtdef\{\langle label \rangle \} \{\langle \text{English} \rangle \} \{\langle \text{Czech} \rangle \} \{\langle \text{Slovak} \rangle \} declaration. The right part is printed by current value of the \texttt{language} register. You can add more languages by re-defining the \texttt{mtdef} command. See the section 2.36.3 for more information.

If you are using \texttt{nonumcitations}, then the \_bimark tokens register have to be prepared in the style file (in \_print:BEGIN, \_print:END, in \_authorname etc.) This value will be used in the \texttt{cite} places in the document.

The example of the style file is in \texttt{bib-simple.opm}.

User or author of the \texttt{bib}. style can create the hidden field which has a precedence while sorting names. Example:

\_CreateField \{\langle sortedby \rangle \}
\_SpecialSort \{\langle sortedby \rangle \}

Suppose that the .bib file includes:

\begin{verbatim}
... \\
author = "Jan Chadima",
sortedby = "Hzadima Jan",
...
\end{verbatim}

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 \texttt{sortedby} = "@", because this character is sorted before A.

2.31.3 The usebib.opm macro file loaded when \_usebib is used

\begin{verbatim}
\_codedecl \MakeReference {Reading bib databases <2020-03-13>} % loaded on demand by \_usebib
\end{verbatim}

Loading the \texttt{librarian.tex} macro package. See \texttt{texdoc librarian} for more information about it.

We want to ignore \texttt{errmessage} and we want not to create \texttt{\jobname.lbr} file.

\begin{verbatim}
\_def\errmessage#1() \errmessage#1
\end{verbatim}
The `usebib` command.

```
\def\usebib/#1 (#2) #3 {%
  \ifx\citelist\empty
    \opwarning{No cited items. \__noexpand\usebib ignored}%
  \else
    \bgroup \par
    \emergencydelim=.3\hsize
    \def\bibpart{undefined} \def\bibfname{none}\fi
    \def\optexbibstyle{#2}%
    \setctable\optexcatcodes
    \input bib-#2.opm
    \the\bibtexhook
    \let\citeI=relax \xdef\citelist{\citelist\citelistB}%
    \global\let\addcitelist=writeXcitelistine\citelist\citelistB\relax
    \ifx\tmp\empty\else % there was \nocite\citelist\citelistB used.
      \setbox0=\vbox{\hsize=\maxdimen \def\citelist{}\edef@{\readbibentry}\input #3.bib\expandafter}
    \expandafter\def\expandafter\citelist\expandafter{\citelist}
    \fi
    \def\citeI[#1]{\csname lb@cite\endcsname{#1}{\bibpart}{}{}}\citelist
    \BibFile{#3}\
    \if s\bibpart\SortList{\bibpart}\fi
    \ReadList{\bibpart}\restorectable
  \egroup
\fi}
\def\readbibentry#1#{\readbibentryA}
\def\readbibentryA#1{\readbibentryB#1,,relax!.}
\def\readbibentryB#1#2,#3relax!.{\addto\citelist{\citeI[#1#2]}}
```

Corrections in librarian macros.

```
\tmpnum\catcode`@\relax
\def\lb@checkmissingentriesI#1,% we needn't \errmsg here, only \opmacwarning
\\\def\lb@checkmissingentries#1\endcsname \endinput}
\def\tmpb\catcode`\i\relax \input librarian \catcode`\i=11 \tmpb
\def\errmsg=\errmsg
\let\newwrite=\newwrite

\_def\newwrite#1\endcsname \def\newwrite#1{\csname lb@restoreat\endcsname \endinput}
\let\tmpb\catcode`\i\relax \input librarian \catcode`\i=11 \tmpb
\let\errmsg=\errmsg
\let\newwrite=\newwrite
```

142
Main action per every entry.

\def\MakeReference{\par \bibskip
  \advance\bibnum by1
  \ifdefined{bin: \the\bibnum} \iftrue
    \edef\tmpb{\csname _bim:\the\bibnum\endcsname}%
    \bibmark{\ea\tmpb}%
    \if \bibmark{} \fi
    \edef\tmpb{\EntryKey}%
    \noindent \dest[cite:\the\bibnum]\printlabel\EntryKey
  \else
    \if\wref\wrefrelax \else
      \immediate\wref\Xbib{{\EntryKey}{\the\bibnum}{\the\bibmark}}\fi
  \fi
  \par}
\def\WriteImmediateInfo#1{} % the existence of .lbr file bocks new reading of .bib
\catcode`@=\tmpnum

The \bprinta, \bprintb, \bprintc, \bprintv commands used in the style files:

\def\bprinta \bprintb*\par
\def\bprintb #1[#2#3]%
  \if!#2\relax
    \def\bibfieldname{#3}\
    \RetrieveFieldIn{#3} \bibfield
  \else
    \RetrieveFieldIn{#2#3} \bibfield
  \fi
\def\bprintc \bprintv%
Various macros + multilinguas.

\_let\_Readauthor=\ReadAuthor \_let\_Readeditor=\ReadEditor
\_def\_bprintaA #1#2{\_ifx\_bibfield\_empty #2\_else\_bprintaB #1**\_eee\_fi}
\_def\_bprintaB #1*#2*#3\eee{\_if^#3^#1\_else\_ea\_bprintaC\_ea{\_bibfield}{#1}{#2}\_fi}
\_def\_bprintaC #1#2{\_if^#1^\_tmpb\_else\_RetrieveFieldIn#1\_tmp\_if^\_tmp\_empty\_else\_tmp\_tmpa \_bprintaC\_tmpb\_tmpa\_fi\_fi}
\_ea\_bprintaB #1,()\_fi
\_if^#1^\_tmpb\_else\_RetrieveFieldIn#1\_tmp\_if^\_tmp\_empty\_else\_tmp\_tmpa \_bprintaC\_tmpb\_tmpa\_fi\_fi
\_ea\_bprintaB #1,()\_fi
\_sdef{\_pp:author}{\_letNames\_authorname}
\_sdef{\_pp:editor}{\_letNames\_editorname}
\_def\_letNames{\_let\_Firstname=\Firstname \_let\_Lastname=\Lastname
\_let\_Von=\Von \_let\_Junior=\Junior}

\_def\_bibwarning{\_opwarning{Missing field "\_bibfieldname" in \[\EntryKey\]}}
\_def\_mtdef#1#2#3#4{\_sdef{\_mt:#1:en}{#2} \_sdef{\_mt:#1:cs}{#3}
\_if$#4$\_slet{\_mt:#1:sk}{\_mt:#1:cs}\_else\_sdef{\_mt:#1:sk}{#4}\_fi}

2.31.4 Usage of the \bib-is690 style

This is the is690 bibliographic style used by Op\TeX.

See op-example.bib for an example of the .bib input. You can try it by:
\fontfam[LMfonts]
\nocite[*]
\usebib/s (is690) op-example
\end

Common rules in .bib files

There are entries of type @FOO{...} in the .bib file. Each entry consists of fields in the form name=","value", or name=\{value\}. No matter which form is used. If the value is pure numeric then you can say simply name=value. Warning: the comma after each field value is mandatory! If it is missing then the next field is ignored or bad interpreted.

The entry names and field names are case insensitive. If there exist a data field no mentioned here then it is simply ignored. You can use it to store more information (abstract, for example).

There are “standard fields” used in ancient bib\TeX (author, title, editor, edition, etc., see http://en.wikipedia.org/wiki/BibTeX). The is690 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):
Firstname(s) von Lastname
or
von Lastname, Firstname(s)
or
von Lastname, After, Firstname(s)

Only the Lastname part is mandatory. Examples:

Petr Olšák
or
Olšák, Petr

Leonardo Piero da Vinci
or
da Vinci, Leonardo Piero
or
da Vinci, painter, Leonardo Piero

The separator “and” between authors will be converted to comma during printing, but between 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_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.
\endtd

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:` \{\textit{value}\} option. The \{\textit{value}\} will be printed instead of the authors list. The \{\textit{value}\} can include \texttt{\\textbackslash AU} macro which expands to the authors list. Example:

```latex
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:` \{\textit{number}\} option. All Firstnames of all authors are trimmed (i. e. reduced to initials) if the number of authors in the author field is greater than or equal to \{\textit{number}\}. There is an exception: `autrim:0` means that no Firstnames are trimmed. This is default behavior. Another example: `autrim:1` means that all Firstnames are trimmed.

```latex
author = "John Green and Bob Brown and Alice Black",
option = "autrim autstr: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 `\texttt{\ \textbackslash \} in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after comma. Example:

```latex
author = "Czech\ Technical\ University\ in\ Prague,
Faculty\ of\ Electrical\ Engineering",
```

output: CZECH TECHNICAL UNIVERSITY IN PRAGUE, Faculty of Electrical Engineering.

### The editor field

The editor field is used for 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 `\texttt{\ and \}`. The Firstnames, Lastnames etc. are interpreted and you can use the options `edmax:` \{\textit{number}\}, `edmin:` \{\textit{number}\}, `edetal`, `edtrim:` \{\textit{number}\} and `edprint:` \{\textit{value}\} (with `\ED` macro). Example:

```latex
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, \texttt{\ eds.}}` or `{\ED, \texttt{\ 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.

```latex
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:` \{\textit{value}\}. Example:

```latex
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$^\text{nd}$",
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
definition = "Second full revised edition",
option = "editionprint:{\EDN}",

The address, publisher, year fields
This is an anachronism from ancient Bib\TeX{} (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, 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 specify 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: @BACHELORTHESIS, @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, \PG, \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 @MISC entry because we need to save the simplicity. They are implemented only for (almost) backward compatibility with the ancient BibTeX. But the ednote is mandatory field here, so you cannot use these entries from the old databases without warnings and without some additional work with the .bib file.

The cite-marks (bibmark) used when \nonumcitations is set
When \nonumcitations is set then \cite prints text orientes bib-marks instead numbers. This style file autogenerates these marks in the form “Lastname of the first author, comma, space, the year” if bibmark field isn’t declared. If you need to set an exception from this common format, then you can use bibmark field.

The OPmac trick http://petr.olsak.net/opmac-tricks-e.html#bibmark describes how to redefine the algorithm for bibmark auto-generating when you need the short form of the type [Au13].

Sorting
If \usebib/c is used then entries are sorted by citation order in the text. If \usebib/s is used then entries are sorted by “Lastname, Firstname(s)” of the first author and if more entries have this value equal, then the year is used (from older to newer). This feature follows the recommendation of the ISO 690 norm.

If you have the same authors and the same year, you can control the sorting by setting years as 2013, 2013a, 2013b, etc. You can print something different to the list using yearprint{⟨value⟩} option, see the section about address, publisher and year above. The real value of year field (ie. not yearprint value) is also used in the text oriented bib-marks when \nonumcitations is set.

If you have some problems with name sorting, you can use the hidden field key, which is used for sorting instead of the “Lastname Firstname(s)” of authors. If the key field is unset then the “Lastname Firstname(s)” is used for sorting normally. Example:
author = "Světla Čmejrková",
key = "Czzmejrkova Svetla",

This entry is now sorted between C and D.

The norm recommends to place the autocitations to the top of the list of references. You can do this by setting \key{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 value parts of the fields (see edition, volume etc. fields) so the automaton have to add some text during processing. But what language have to be chosen?

The current value of the \language{} register at the start of the .bib processing is decided as the language of the outer document. This language is used for technical notes regardless of the entry language. Each entry can have the lang field with the two-letter mark of the entry language. This language is used for ed./vyd., vol./ročník etc. and it is used for hyphenation too. If the entry language is not set then the outer document language is used.

If the outer document language is known before creating of the .bib file, you can store some language-dependent phrases into it. On the other hand, if the main document language is unknown, you can use the \Mtext macro to create the text multilingual. Example:

\howpublished{} = "\Mtext{blue-ray}"

Now, you can set the variants of blue-ray into your macros:

\_mtdef {blue-ray} {Blue-ray disc} {Blue-ray disk} {} 

Tips for using more languages
This style prefers English, Czech and Slovak languages. However, you can add more languages. Use the shortcuts of language names (de and pl in the example below). You can define all phrases for your language:

\mtdefx#1#2#3{\sdef{_mt:#1:de}{#2}\sdef{_mt:#1:pl}{#3}}
\mtdefx {bib.and} { und } { a }
\mtdefx {bib.phdthesis} {Ph.D. Dissertation} {Praca doktorska}
...

See more about language phrases in the 2.36.3 section.

Summary of non-standard fields
This style uses the following fields unknown by \bib\TeX{}:

\begin{verbatim}
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
\end{verbatim}

Summary of options

\begin{verbatim}
aumax:<number> ... maximum number of printed authors
aumin:<number> ... number of printed authors if aumax exceeds
autrim:<number> ... full Firstnames iff number of authors are less than this
\end{verbatim}
2.3.1.5 Implementation of the bib-iso690 style

Option field.

Formatting of Author/Editor lists.
\let\editorname=\authorname
\def\prepareauthername{%  
  \def\mabyetal{}\csname lb@abbreviatefalse\endcsname
  \biboptionvalue{#1max}\authormax
  \biboptionvalue{#1min}\authormin
  \biboptionvalue{#1pre}\authorpre
  \biboptionvalue{#1print}\authorprint
  \iftrue \def\maybeetal{bib.etal}\fi
  \biboptionvalue{#1trim}\autrim
  \ifnum\namecount<\autrim\relax \else \AbbreviateFirstname \fi
}
\def\maybeetal{}

\ifx\upper\undefined\ifx\caps\undefined\def\upper{\uppercase}\else\def\upper{\caps\rm}\fi\fi\let\upper=\upper
\def\setbibmark{%  
  \if\dobibmark\undefined\def\dobibmark{}\fi
  \RetrieveFieldIn{bibmark}\tmp
  \if\tmp\undefined\RetrieveFieldIn{year}\tmp\edef\tmp{\dobibmark, \tmp}\fi
  \bibmark=\ea{\tmp}\relax
}

% Multilinguals: English Czech Slovak
\mtdef{bib.and}{, and }{a }{}
\mtdef{bib.etal}{{ et al.}}{{ a~kol.}}{}
\mtdef{bib.edition}{ed.}{vyd.}{}
\mtdef{bib.bachthesis}{Bachelor’s Thesis}{Bakalářská práce}{Bakalárska práca}
\mtdef{bib.masthesis}{Master’s Thesis}{Diplomová práce}{Diplomová práca}
\mtdef{bib.phdthesis}{Ph.D. Thesis}{Disertační práce}{Dizertačná práca}
\mtdef{bib.available}{Available from}{Dostupné na}{}
\mtdef{bib.availablealso}{Available also from}{Dostupné též na}{}
\mtdef{bib.citedate}{cit.~}{vid.~}{}
\mtdef{bib.volume}{Vol.~}{ročník~}{}
\mtdef{bib.number}{No.~}{č.~}{}
\mtdef{bib.prepages}{pp.~}{s.~}{}
\mtdef{bib.postpages}{~p.}{~s}{}
\mtdef{bib.editor}{,~ed.}{,~editor}{}
\mtdef{bib.editors}{,~eds.}{,~editori}{}
\chardef\documentlanguage\language
\def\Mtext#1{\csname _mt:#1\endcsname\csname _lan:\the\documentlanguage\endcsname\csname _endcsname\endcsname}
\CreateField {lang}
Non-standard fieldnames.

```latex
\_CreateField {ednote}
\_CreateField {citedate}
\_CreateField {numbering}
\_CreateField {isbn}
\_CreateField {issn}
\_CreateField {doi}
\_CreateField {url}
\_CreateField {bibmark}
```

Sorting.

```latex
\_SortingOrder{name,year}{lfvj}
\_SpecialSort {key}
```

Supporting macros.

```latex
\_def\_bibwarninga{\_bibwarning}
\_def\_bibwarningb{\_bibwarning}
\_def\_docitedate #1/#2/#3/#4{[\_Mtext{bib.citedate}\
\_if^#2^#1\_else\_if^#3^#1/#2\_else \_docitedateA{#1}{#2}{#3}\_fi\_fi][\_Mtext{bib.citedate}]
\_ifnum\_documentlanguage=\_csPatt \_docitedateCS{#1}{#2}{#3}\_else \_ifnum\_documentlanguage=\_skPatt \_docitedateSK{#1}{#2}{#3}\_else \_docitedateEN{#1}{#2}{#3}\_fi\_fi}
\_def\_docitedateA#1#2#3{\_ifnum\_documentlanguage=\_csPatt \_docitedateCS{#1}{#2}{#3}\_else \_ifnum\_documentlanguage=\_skPatt \_docitedateSK{#1}{#2}{#3}\_else \_docitedateEN{#1}{#2}{#3}\_fi\_fi}
\_def\_docitedateEN#1#2#3{#1-#2-#3}
\_def\_docitedateCS#1#2#3{\_hbox{\_tmpnum=#3 \_the\_tmpnum. \_tmpnum=#2 \_the\_tmpnum. #1}}
\_def\_doyear#1{\_biboptionvalue{yearprint}\_yearprint\_ifx\_yearprint\_empty#1\_else \_def\_YEAR{#1}\_yearprint\_fi}
```

153
Entry types.

```latex
\_sdef\{\text{print}:\text{BEGIN}\}{\%
\_readbiboptions
\_biboptionvalue(titlepost)\_titlepost
\_isbiboption(unpublished)\_iffalse \_let\_bibwarninga=\_relax \_let\_bibwarningb=\_relax \_fi
\_isbiboption(nowarn)\_iffalse \_let\_bibwarning=\_relax \_fi
\_isbiboption(urlalso)\_iffalse \_def\{\text{preurl}\{\text{bib.availablealso}\}\_fi
\_RetrieveFieldIn(lang)\_langentry \_setlang=\_langentry
\} \_sdef\{\text{print}:\text{END}\}{\%
\_bprinta [note] \{.*\}\%
\_setbibmark
\} \_def=\_bookgeneric#1{\%
\_bprinta [howpublished] \{[*]\}. \%
\_bprintb [edition] \{\_doedition{##1}\{.\}\%
\_bprinta [ednote] \{+. \%
\_bprinta [address] \{\_bprintf[\text{publisher}]{\{\_bprintf[\text{year}]{.\{.\}\}}\{\_bibwarninga}\%
\_bprintf[\text{year}] \{\_doyear{##1} \_bprintv[\text{citedate}]{\_bprintv[\text{numbering}]{.\}}{.}\ \}}{\_bibwarning}\%
\_bprintf [numbering] \{\_preparenumbering*\_bprintf[\text{citedate}]{\{.\}}\{.\}\%
\_bprintfa [volume] \{\_prevolume*\_bprintf[\text{number}]{\_bprintf[\text{pages}]{.\{.\}\}}{.}\%
\_bprintfb [pages] \{\_prepages\_hbox{##1} \_bprintv[\text{citedate}]{\{.\}}{.}\%
\_bprintfa [citedate] \{\_docitedate*///\_relax.\%
\_bprintfa [isbn] {ISBN~*.\%
\_bprintfb [issn] {ISSN~*.\%
\_bprintfb [doi] \{_predoi DOI \_ulink[http://dx.doi.org/#1]{##1}.\%
\_bprintfb [url] \{_preurl\_url{##1}.\%
\} \_bprintb [!author] \{\_doauthor1{##1}\{.\}\{\_bibwarning\%
\_bprintfa [title] \{\_em##1\_bprintc[\text{titlepost}]{\_bprintv[\text{howpublished}]{.\}}{.}\%
\_bookgeneric\%
\} \_sdef\{\text{print}:\text{article}\}{\%
\_biboptionvalue(journalpost)\_journalpost
\_bprintf [author] \{\_doauthor1{##1}\{.\}\{\_bibwarning\%
\_bprintfa [title] \{\_em##1\_bprintc[\text{journalpost}]{\_bprintv[\text{howpublished}]{.\}}{.}\%
\_bookgeneric\%
\} \_sdef\{\text{print}:\text{inbook}\}{\%
\_let\_bibwarningb=\_relax
\_bprintf [!author] \{\_doauthor1{##1}\{.\}\{\_bibwarning\%
\_bprintfa [title] \{\_em##1\_bprintfc[\text{titlepost}]{\_bprintfv[\text{howpublished}]{.\}}{.}\%
\_bookgeneric\%
\} \_sdef\{\text{print}:\text{book}\}{\%
\_biboptionvalue(journalpost)\_journalpost
\_bprintf [author] \{\_doauthor1{##1}\{.\}\{\_bibwarning\%
\_bprintfa [title] \{\_em##1\_bprintfc[\text{journalpost}]{\_bprintfv[\text{howpublished}]{.\}}{.}\%
\_bookgeneric\%
\} \_sdef\{\text{print}:\text{inbook}\}{\%
\_biboptionvalue(journalpost)\_journalpost
\_bprintf [author] \{\_doauthor1{##1}\{.\}\{\_bibwarning\%
\_bprintfa [title] \{\_em##1\_bprintfc[journalpost]{\_bprintfv[\text{howpublished}]{.\}}{.}\%
\_bookgeneric\%
\} \_sdef\{\text{print}:\text{article}\}{\%
\_biboptionvalue(journalpost)\_journalpost
\_bprintf [author] \{\_doauthor1{##1}\{.\}\{\_bibwarning\%
\_bprintfa [title] \{\_em##1\_bprintfc[journalpost]{\_bprintfv[\text{howpublished}]{.\}}{.}\%
```

154
2.32 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 strings is equal in primary pass then secondary pass is started. It distinguishes between variously accented letters. Czech rules, for example says: not accented before diacritic 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}
\def \_sortingdatacs {
155
\end{verbatim}
Characters ignored by sorting algorithm are declared in \_ignoredchars\{iso-code\}. The compound characters (two or more characters interpreted as one character in sorting algorithm) is mapped to single invisible characters in \_compoundchars\{iso-code\}. Czech rules declares ch or Ch or CH as a single letter sorted between H and I. See \_sortingdatacs above where these declared characters are used.

The characters declared in \_ignoredchars are ignored in first pass without additional condition. All characters are taken into account in second pass: ASCII characters with code ‘65’ are sorted first if they are not mentioned in the \_sortingdata\{iso-code\} macro. Others not mentioned characters have undefined behavior during sorting.

Slovak sorting rules are the same as Czech. The macro \_sortingdata includes Slovak letters too. Compound characters are the same. English sorting rules can be defined by \_sortingdata too because English alphabet is subset of Czech and Slovak alphabets. Only difference: \_compoundcharsen is empty in English rules.

You can declare these macros for more languages, if you wish to use \makeindex with sorting rules in respect to your language. Note: if you need to map compound characters to a character, don’t use ‘\^I’ or ‘\^M’ because these characters have very specific category code. And use space to separate more mappings, like in \_compoundcharscs above.

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.
Preparing to secondary pass is implemented by the \_setsecondarysorting macro.

Strings to be sorted are prepared in \langle string\rangle 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\rangle&\_relax\langle converted string2\rangle\_relax \langle string1\rangle\langle string2\rangle 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 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. \OpTeX{} uses comma at this place for sorting indexes: \texttt{⟨word1⟩, ⟨word2⟩, ⟨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.

\_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}\uppercase{#1}}\medskip}
```
\_printiipages (pglist) & gets \langle pglist \rangle in the form \langle pg \rangle : \langle type \rangle , \langle pg \rangle : \langle type \rangle , \ldots \langle pg \rangle : \langle type \rangle and it converts them to \langle pg \rangle , \langle pg \rangle , \langle from \rangle \ldots \langle to \rangle , \langle pg \rangle etc. The same pages must be printed only once and continuous consequences of pages must be compressed to the form \langle from \rangle \ldots \langle to \rangle . Moreover, the consequence is continuous only if all pages have the same \langle type \rangle . Empty \langle type \rangle is most common, pages with b \langle type \rangle must be printed as bold and with i \langle type \rangle as italics. Moreover, the \langle pg \rangle mentioned here are \langle gpgeno \rangle , but we have to print \langle pageo \rangle . The following macros solves these tasks.

You can re-define \_pgprint \langle gpgeno \rangle : \langle iitype \rangle if you need to implement more \langle iitype \rangle .

The \_index\{\word\} 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 gpgeno \rangle : \langle iitype \rangle is appended to this list. Moreover, we need a mapping from \langle gpgeno \rangle to \langle pageo \rangle , because we print \langle pageo \rangle in the index, but hyperlinks are implemented by \langle gpgeno \rangle . So, the macro \_pgi: \langle gpgeno \rangle is defined as \langle pageo \rangle .

The implementation of macros \_ii, \_iid, \_iiis 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
The \iindex{\word} directly.

The \iitype{\type} saves the \type{} to the \_iitypesaved macro. It is used in the \iindex macro.

\makeindex

\begin{verbatim}
438 \def\ii #1 \{\_leavevmode\_def\_tmp(#1)\_ii #1 \_,\_\_def\_iitypesaved{}\}
439 \_def\iiA #1,\{\_if$#1$\_else\_def\tmpa{#1}\
440 \_ifx\tmpa\_iiatsign \_ea\_iiB\_tmp,\_else\_iindex{#1}\_fi\}
441 \_def\_iiatsign{@}
442 \_def\iiB #1,\{\_if$#1$\_else \_iiC#1/\_relax \_ea\_iiB\_fi\}
443 \_def\iiC #1/#2 \_relax\{\_if$#2$\_else\_iindex{#2#1}\_fi\}
444 \def\ iid #1 \{\_leavevmode\_iindex{#1}#1\_futurelet\_tmp\_iiD\_def\_iitypesaved{}\}
445 \_def\_iiD\{\_ifx\_tmp,\_else\_ifx\_tmp.\_else\_space\_fi\_fi\}
446 \_def\_iis #1 #2\{\_def~{ }\_global\_sdef{_,#1}{#2}\_ignorespaces\}
447 \_def\_iitypesaved{} \_def\_iitype #1\{\_def\_iitypesaved{#1}\_ignorespaces\}
448 \_public \ii \iid \iis \iitype ;
\end{verbatim}

\section*{2.33 Footnotes and marginal notes}

\_gfnotenum is counter which counts footnotes globally in the document. whole document, chapters, pages. 
\_lfnotenum is counter which counts footnotes at each chapter from one. It is used for local page foot-
note counters too. 
\_ifpgfnotenote says that footnote numbers are counted on each page from one. We need to run \openref in such case. 
\_gfnotenum is a macro which expands to footnote number counted in declared part. 
\_fnotenumberchapters declares footnotes numbered in each chapter from one (default), \_fnotenumglobal declares footnotes numbered in whole document from one and \_fnotenumpages declares footnotes num-
bered at each page from one. 

\begin{verbatim}
19 \newcount\_gfnotenum \_gfnotenum=0
20 \newcount\_lfnotenum
21 \newif\_ifpgfnotenote
22 \_def\_fnotenumberglobal {\_def\_fnotenum{\_the\_gfnotenum}\_pgfnotefalse}
23 \_def\_fnotenumberchapters {\_def\_fnotenum{\_the\_gfnotenum}\_pgfnotefalse}
24 \_def\_fnotenumpages {\_def\_fnotenum{\_trycs{\_fn:\_the\_gfnotenum(??)}\_pgfnotetrue}}
25 \_fnotenumberchapters \% default are footnotes counted from one in each chapter
26 \_def\_fnotenum{\_gfnotenum}
27 \_public \_fnotenumberglobal \_fnotenumberchapters \_fnotenumpages ;
28 \_let \_runningfnotenote = \_fnotenumberglobal \% for backward compatibility
\end{verbatim}

The \_printfnotemark prints the footnote mark. You can re-define this macro if you want another design of footnotes. For example

\begin{verbatim}
\_fnotenumpages
\def\_printfnotemark \{\ifcase 0\fnotenum
\*\or**\or***\or$^\mathbox{†}$\or$^\mathbox{‡}$\or$^\mathbox{††}$\fi\}
\end{verbatim}

This code gives footnotes* and ** and*** and† etc. and it supposes that there are no more than 6 footnotes at one page.

If you want to distinguish between footnote marks in the text and in the front of footnote itself, then you can define \printfnotemarkA and \printfnotemarkB.

The \fnotelinks{\colorA}{\colorB} implements the hyperlinked footnotes (from text to footnote and backward).
Each footnote saves the \Xfnote (without parameter) to the .ref file (if \openref). We can create the mapping from ⟨gfnotenum⟩ to ⟨pgfnotenum⟩ in the macro \fn:\(fnotenum⟩. Each \Xpage macro sets the \lfnotenum to zero.

The \fnotetext{⟨text⟩} macro is simple, \fnote and \fnotetext does the real work.

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\leq #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 \mnoteskip is a dimension which denotes the vertical shift of marginal note from its normal position. Positive value means shift up, negative down. The \mnoteskip register is set to zero after the marginal note is printed. The new syntax \mnote up\{dimen\}\{text\} is possible too, but public \mnoteskip is kept for backward compatibility.

\begin{verbatim}
138 \_long\_def\_mnoteA #1{\_ifvmode {\_mnoteA{#1}}\nobreak\_vskip-\_baselineskip \_else
139 \lower\dp\strutbox\hbox{}\vadjust{\_kern-\dp\strutbox \_mnoteA{#1}\_kern\dp\strutbox}\%}
140 }\fi
141 }
142 \_public \_mnote ;
\end{verbatim}

We don't want to process \fnote, \fnotemark, \mnote in TOC, headlines nor outlines.

\begin{verbatim}
162 \_def\_mnoteA #1{\_incr\_mnotenum
163 \_ifx\_mnotesfixed\_undefined
164 \_edef\_mnotesfixed{\_cs{\_mn:\_the\_mnotenum}}\%
165 \_else
166 \_opwarning{unknown \noexpand\mnote side. \TeX me again}\_openref
167 \_incr\_unresolvedrefs
168 \_def\_mnotesfixed{\_right}\%
169 \_fi\_fi
170 \_hbox to0pt{\_wref\_Xmnote{}\_everypar={}\%
171 \_lrmnote{\_kern-\_mnotesize \_kern-\_mnoteindent}{\_kern\hsize \_kern\mnoteindent}\%
172 \_vbox to0pt{\_vss \_setbox0=\_vtop{\_hsize=\_mnotesize
173 \_lrmnote{\_leftskip=0pt plus 1fill \_rightskip=0pt}{\_rightskip=0pt plus 1fil \_leftskip=0pt}\%
174 {\_the\_everymnote\_noindent#1\_endgraf}}\%
175 \_dp0=0pt \_box0 \_kern\_mnoteskip \_global\_mnoteskip=0pt}\_hss}\%
177 }
178 \_def \_lrmnote#1#2{\_ea\_ifx\_mnotesfixed\_left #1\_else #2\_fi}
\end{verbatim}

2.34 Styles

Op\TeX provides three styles: \report, \letter and \slides. Their behavior is documented in user part of the manual in the section 1.7.2 and \slides style (for presentations) is documented in op-slides.pdf which is an example of the presentation.

2.34.1 \report and \letter styles

\begin{verbatim}
3 \_codedecl \report {Basic styles of Op\TeX \<2020-03-28\>} \% preloaded in format
\end{verbatim}

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 \hs. The \{col\} is set active and \everypar starts \hbox{} and active \{col\} closes this \hbox by .

\begin{verbatim}
16 \_def\_boxlines\{\%
17 \_def\_boxlinesE{\_ifhmode\_egroup\_empty\_fi}\%
18 \_def\_nl{\_boxlinesE}\%
19 \_bgroup \_lccode`\~=`\^^M \_lowercase{\_egroup\_let~}\_boxlinesE\%
20 \_everypar{\_setbox0=\_vtop{\_hs=\_mnotesize
21 \_lrmnote{\_leftskip=0pt plus 1fill \_rightskip=0pt}{\_rightskip=0pt plus 1fil \_leftskip=0pt}\%
22 {\_the\_everymnote\_noindent#1\_endgraf}}\%
23 \_dp0=0pt \_box0 \_kern\_mnoteskip \_global\_mnoteskip=0pt}\_hs}\%
24 }
25 \_def\_boxlinesC{\_futurelet\_next\_boxlinesD}
\end{verbatim}
The \texttt{\report} and \texttt{\letter} style initialization macros are defined here. The \texttt{\letter} defines \texttt{\address} and \texttt{\subject} macros.

\begin{verbatim}
\_def\report{\_typosize[11/13.2] \_vsize=\_dimexpr \_topskip + 52\_baselineskip \_relax % added 2020-03-28
\_let\_titfont=\_chapfont
\_titskip=3ex
\_esodef\_author##1{\_removelastskip\_bigskip
\{\_leftskip=0pt plus1fill \_rightskip=\_leftskip \_it \_noindent ##1\_par}\_nobreak\_bigskip
}\}
\_public \_author ;
\_parindent=1.2em \_iindent=\_parindent \_ttindent=\_parindent
\_footline={\_global\_footline={\_hss\_rmfixed\_folio\_hss}}
}\_def\letter{
\_def\address{\_vtop\_bgroup\_boxlines \_parskip=0pt \_let\par=\_egroup}
\_def\subject{\{\_bf \_mtext{subj}: \}}
\_public \_address \_subject ;
\_typosize[11/14]
\_vsize=\_dimexpr \_topskip + 49\_baselineskip \_relax % added 2020-03-28
\_parindent=0pt
\_parskip=\_medskipamount
\_nopagenumbers
}\_public \_letter \_report ;
\end{verbatim}

The \texttt{\slides} macro reads macro file \texttt{slides.opm}, see the section \texttt{2.34.2}.

\begin{verbatim}
\_def\slides{\_par
\_input slides.opm
}\_public \_slides ;
\end{verbatim}

\texttt{2.34.2 \_slides style for presentations}

\texttt{\_codecl \_slideshow \{Slides style for \LaTeX{} <2020-03-19>\} % loaded on demand by \_slides}

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.

\begin{verbatim}
\_margins/1 a5l (14,14,10,3)mm % landscape A5 format
\_def\_wideformat{\_margins/1 (263,148) (16,16,10,3)mm } % 16:9 format
\_fontfan[\texttt{\_fontfamily[\_Heros]}]
\_typosize[16/19]
\_fanvardef\_ttfont{\_setfontsize{mag1.15}\_tt}
\_def\_urlfont{}
\_everytt={\_typosize[13/16] \_advance\_hsize by10mm}
\_fontdef\_fixbf{\_bf}
\_nopagenumbers
\_parindent=0pt
\_ttindent=5mm
\_parskip=5pt plus 4pt minus2pt
\_rightskip=0pt plus 1fill
\_ttindent=10pt
\_def\_ttskip{\_smallskip}
\_onlyrgb % RGB color space is better for presentations
\_footline=
\end{verbatim}

The bottom margin is set to 3 mm. If we use 1 mm, then baseline of \texttt{\footline} is 2 mm from the bottom page. This is depth of the \texttt{\grey} rectangle used for page numbers. It is r-lapped to \texttt{\hoffset} width because left margin = \texttt{\hoffset} = right margin. It is 14 mm for narrow pages or 16 mm for wide pages.
The \subtit is defined analogically like \tit.

The \pshow\langle num\rangle prints the text in invisible (transparent) font when \layernum<\langle num\rangle. The transparency is set by \pdfpagemessages\primitive.

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.

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 \slideshow is active then each page is opened by \setbox\_slidepage=\vbox\bgroup (roughly speaking) and closed by \egroup. The material is \unvboxed and saved for the usage in the next usage if \pg+ is in process. The \_slidelayer is incremented instead \pageno if \pg+. This counter is equal to \count1, so it is printed to the terminal and log file next to \pageno. The code is somewhat more complicated when \layers is used. Then \langle layered-text\rangle is saved to the \_slidetext macro, the material before it is in \_slidepage box and the material after it is in \_slidepageB box. The pages are completed in the \loop which increments the \layernum register.
Default \texttt{\textbackslash layers\numexpr\texttt{num}+1\relax} macro (when \texttt{\textbackslash slideshow} is not activated) is simple. It prints the \texttt{⟨layered-text⟩} with \texttt{\layernum=\numexpr\texttt{num}+1\relax} because we need the result after last layer is processed.
We must redefine \texttt{\fnotenumpages} because the data from \texttt{.ref} file are less usable for implementing such feature: the footnote should be in more layers repeatedly. But we can suppose that each page starts by \texttt{\pg}; macro, so we can reset the footnote counter by this macro.

\begin{verbatim}
211 \def \fnotenumpages {\def \fnotenum {\the \lfnotenum} \pgfnotefalse
212 \let \lfnotenumreset = \relax
213 \public \fnotenumpages ;
\end{verbatim}

\section{Logos}

Despite plain \TeX each macro for logos ends by \texttt{\ignoreslash}. This macro ignores next slash if it is present. You can use \texttt{\TeX/ like this} for protecting the space following the logo. This is visually more comfortable. The macros \texttt{\TeX}, \texttt{\OpTeX}, \texttt{\LuaTeX}, \texttt{\XeTeX} are defined.

\begin{verbatim}
13 \protected \def \TeX {T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\ignoreslash}
14 \protected \def \OpTeX {Op\kern-.1em\TeX}
15 \protected \def \LuaTeX {Lua\TeX}
16 \protected \def \XeTeX {X\kern-.125em\phantom E\pdfsave\rlap{\pdfscale{-1}{1}\lower.5ex\hbox{E}}\pdfrestore \TeX}
19 \def \ignoreslash {\futurelet\next \ignoreslashA}
20 \def \ignoreslashA {\ifx\next/\ea\ignoreit\fi}
21 \public \TeX \OpTeX \LuaTeX \XeTeX \ignoreslash ;
\end{verbatim}

The \texttt{\slantcorr} macro expands to slant-correction of current font. It is used to shifting A if the \LaTeX logo is in italic.

\begin{verbatim}
29 \protected \def \LaTeX{\tmpdim=.42ex L\kern-.36em \kern \slantcorr \% slant correction
30 \raise \texpr \hbox{\thefontscale[710]A}}\TeX
32 \def \slantcorr{\ea\ignorept \the\fontdimen1\font\tmpdim}
33 \public \LaTeX ;
\end{verbatim}

The expandable versions of logos used in Outlines needs the expandable \texttt{\ignslash} (instead of the \texttt{\ignoreslash}).

\begin{verbatim}
52 \def \ignslash#1{\ifx#1/\else #1\fi}
53 \regmacro {}{}{% conversion for PDF outlines
54 \def \TeX{\TeX\ignslash}\def \OpTeX{\OpTeX\ignslash}\TeX
55 \def \LaTeX{\LaTeX\ignslash}\def \OpMac{\OpMac\ignslash}\LaTeX
56 \def \CS{\CS\ignslash}\def \csplain{\csplain\ignslash}\CS
57 }\public \ignslash ;
\end{verbatim}
2.36 Multilingual support

2.36.1 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.

```
\_wterm{Setting lccodes and uccodes for Unicode characters}
\def\_tmp #1 #2 {\_ifx^#1^\_else
\_lccode"#1="#1
\_ifx.#2%
\_uccode"#1="#1
\_else
\_uccode"#2="#2
\_lccode"#2="#1
\_uccode"#1="#2
\fi
\_ea \_tmp \_fi}
\tmp
00AA .
00B5 039C
00BA .
00E0 00C0
00E1 00C1
00E2 00C2
00E3 00C3
00E4 00C4
...etc. (see uni-lcuc.opm)
```

2.36.2 Hyphenations

The \texttt{iso-code} means a shortcut of language name (mostly by ISO 639-1). The following control sequences are used for language switching:

- \texttt{\l an \langle number \rangle} expands to \texttt{\langle iso-code \rangle} of the language. The number is internal number of languages used as a value of \texttt{\language} register.
- \texttt{\ulan \langle long-lang \rangle} expands to \texttt{\langle iso-code \rangle} too. This is transformation from long name of language (lowercase letters) to \texttt{\langle iso-code \rangle}.
- \texttt{\langle iso-code \rangle}Patt (for example \texttt{\csPatt}) is the language \texttt{\langle number \rangle} declared by \texttt{\chardef}.
- \texttt{\langle iso-code \rangle}lang (for example \texttt{\enlang}, \texttt{\cslang}, \texttt{\sklang}, \texttt{\delang}, \texttt{\pllang}) is language selector. It exists in two states
  - Initialization state: when \texttt{\langle iso-code \rangle}lang is used first then it must load the patterns into memory using Lua code. If it is done then the \texttt{\langle iso-code \rangle}lang re-defines itself to processing state.
  - Processing state: it only sets \texttt{\language=\langle iso-code \rangle}Patt, i.e it selects the hyphenation patterns. It does a little more language-dependent work, as mentioned below.
- \texttt{\langspecific \langle isocode \rangle} is processed by \texttt{\langle iso-code \rangle}lang and it should include language-specific macros declared by user or macro designer.
The USenglish patterns are preloaded first:

\chardef\enPatt=0
\def\pattlist{\enPatt=0}
\edef\langlist{en}(USenglish)\en
\def\enlang{\en\enPatt=0} % \lefthyph=2 \righthyph=3
\def\enlang{\en}\en
\edef\langspecific{en}{\nonfrenchspacing}

\input hyphen \enPatt=0 % \nonfrenchspacing
\input hyphen % en(USenglish) patterns from TeX82

\chardef\lanopm\enPatt=0\_chardef\enPatt=0\_def\pattlist{\enPatt=0}
\_def\langlist{en}(USenglish)\en
\_sdef\_lan:0{en}\en
\_sdef\_ulan:usenglish{en}\en
\_def\enlang{\en\enPatt23} % \lefthyph=2 \righthyph=3
\_def\enlang{\en}\en
\_sdef\_langspecific:en{\nonfrenchspacing}
\_lefthyphenmin=2 \_righthyphenmin=3 % disallow x- or -xx breaks
\_input hyphen % en(USenglish) patterns from TeX82

\_preplang ⟨iso-code⟩ ⟨long-lang⟩ ⟨hyph-file-spec⟩ ⟨number⟩ ⟨pre-hyph⟩⟨post-hyph⟩ prepares the ⟨iso-code⟩lang
to its initialization state. Roughly speaking, it does:
\chardef\lanopm\enPatt=0\_chardef\enPatt=0\_def\pattlist{\enPatt=0}
\_def\langlist{en}(USenglish)\en
\_sdef\_lan:0{en}\en
\_sdef\_ulan:usenglish{en}\en
\_def\enlang{\en\enPatt23} % \lefthyph=2 \righthyph=3
\_def\enlang{\en}\en
\_sdef\_langspecific:en{\nonfrenchspacing}
\_lefthyphenmin=2 \_righthyphenmin=3 % disallow x- or -xx breaks
\_input hyphen % en(USenglish) patterns from TeX82

\_preplang ⟨iso-code⟩ ⟨long-lang⟩ ⟨hyph-file-spec⟩ ⟨number⟩ ⟨pre-hyph⟩⟨post-hyph⟩ prepares the ⟨iso-code⟩lang
to its initialization state. Roughly speaking, it does:
\chardef\lanopm\enPatt=0\_chardef\enPatt=0\_def\pattlist{\enPatt=0}
\_def\langlist{en}(USenglish)\en
\_sdef\_lan:0{en}\en
\_sdef\_ulan:usenglish{en}\en
\_def\enlang{\en\enPatt23} % \lefthyph=2 \righthyph=3
\_def\enlang{\en}\en
\_sdef\_langspecific:en{\nonfrenchspacing}
\_lefthyphenmin=2 \_righthyphenmin=3 % disallow x- or -xx breaks
\_input hyphen % en(USenglish) patterns from TeX82

You can see that \_loadpattrs \_uselang first (in initialization state) and it runs only \_uselang when it is called again (in processing state).

\_loadpattrs ⟨hyph-file-spec⟩ ⟨number⟩ ⟨long-lang⟩ loads hyphenation patterns and hyphenation exceptions for given language and registers them as \language=⟨number⟩.

The (hyph-file-spec) is a part of full file name which is read: hyph-{hyph-file-spec}.tex. The patterns and hyphenation exceptions are saved here in UTF-8 encoding. The (hyph-file-spec) should be a list of individual ⟨hyph-file-spec⟩’s separated by comma, see the language Serbian below for example.

\_uselang{⟨iso-code⟩}⟨iso-code⟩Patt ⟨pre-hyph⟩⟨post-hyph⟩ sets \language, \lefthyphenmin, \righthyphenmin and runs \frenchspacing. This default language-dependent settings should be re-declared by \langspecific{iso-code} which is run finally (it is \relax by default, only \langspecific:en runs \nonfrenchspacing).
The $\texttt{uselanguage}$ \{⟨long-lang⟩\} is defined here (for compatibility with e-plain users).

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.

```plaintext
\message{Language hyph.patterns ready to load: \langlist.
Use \string\shortname\lang to initialize language,
\string\cslang\space for example}
```

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.

```latex
\def\langset #1 #2{\sdef{_langspecific:#1}{#2}}
\langset fr {... declare French quotation marks}
\langset de {... declare German quotation marks}
\langset gr {... switch to Greek fonts family}
... etc.
```

Note that you need not to set language specific phrases (like `\today`) by this code. Another concept is used for such tasks. See the section 2.36.3 for more details.

### 2.36.3 Multilingual phrases and quotation marks

Only four words are generated by `\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).

```latex
\def\mtext#1{\trycs{\mt:#1:en:en}}
```

---

<table>
<thead>
<tr>
<th>Language</th>
<th>Shortname</th>
<th>Code</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oriya</td>
<td>or</td>
<td>219 11</td>
<td></td>
</tr>
<tr>
<td>Panjabi</td>
<td>pa</td>
<td>220 11</td>
<td></td>
</tr>
<tr>
<td>Tamil</td>
<td>ta</td>
<td>221 11</td>
<td></td>
</tr>
<tr>
<td>Telugu</td>
<td>te</td>
<td>222 11</td>
<td></td>
</tr>
<tr>
<td>Belarusian</td>
<td>be</td>
<td>223 22</td>
<td></td>
</tr>
<tr>
<td>Bulgarian</td>
<td>bg</td>
<td>224 22</td>
<td></td>
</tr>
<tr>
<td>Bengali</td>
<td>bn</td>
<td>225 11</td>
<td></td>
</tr>
<tr>
<td>Church Slavonic</td>
<td>cu</td>
<td>226 12</td>
<td></td>
</tr>
<tr>
<td>Old German</td>
<td>de-1901</td>
<td>227 22</td>
<td></td>
</tr>
<tr>
<td>Swiss German</td>
<td>de-ch-1901</td>
<td>228 22</td>
<td></td>
</tr>
<tr>
<td>Esperanto</td>
<td>eo</td>
<td>229 22</td>
<td></td>
</tr>
<tr>
<td>Friulian</td>
<td>fur</td>
<td>230 22</td>
<td></td>
</tr>
<tr>
<td>Gujarati</td>
<td>gu</td>
<td>231 22</td>
<td></td>
</tr>
<tr>
<td>Georgian</td>
<td>ka</td>
<td>232 12</td>
<td></td>
</tr>
<tr>
<td>Macedonian</td>
<td>mk</td>
<td>233 22</td>
<td></td>
</tr>
<tr>
<td>Occitan</td>
<td>oc</td>
<td>234 22</td>
<td></td>
</tr>
<tr>
<td>Pali</td>
<td>pi</td>
<td>235 12</td>
<td></td>
</tr>
<tr>
<td>Piedmontese</td>
<td>pms</td>
<td>236 22</td>
<td></td>
</tr>
<tr>
<td>Romansh</td>
<td>rm</td>
<td>237 22</td>
<td></td>
</tr>
<tr>
<td>Serbian</td>
<td>sr</td>
<td>238 22</td>
<td></td>
</tr>
<tr>
<td>Swedish</td>
<td>sv</td>
<td>239 22</td>
<td></td>
</tr>
<tr>
<td>Thai</td>
<td>th</td>
<td>240 23</td>
<td></td>
</tr>
<tr>
<td>Ethiopic</td>
<td>ethi</td>
<td>241 11</td>
<td></td>
</tr>
</tbody>
</table>
Using \texttt{\_langw} \texttt{\lang} \texttt{\{chapter\}} \texttt{\{table\}} \texttt{\{figure\}} \texttt{\{subject\}} you can declare these words more effectively:

\begin{verbatim}
\_def \_langw \#1 \#2 \#3 \#4 \#5 {%
\_sdef{\mt:chap:\#1}{\#2}\_sdef{\mt:t:\#1}{\#3}\_sdef{\mt:f:\#1}{\#4}\%
\_sdef{\mt:subj:\#1}{\#5}\%
}%
\_langw en Chapter Table Figure Subject
%------------------------------------------------------------
\_langw cs Kapitola Tabulka Obrázek Věc
\_langw de Kapitel Tabelle Abbildung Betreff
\_langw es Capítulo Tabla Figura Sujeto
\_langw fr Chaptire Tableau Figure Matière
\_langw it Capitolo Tabella Fig. Oggetto
\_langw pl Rozdział Tabela Ilustracja Temat
\end{verbatim}

... etc. \texttt{(see languages.opm)}

You can add more words as you wish. For example \texttt{\today} macro:

\begin{verbatim}
\_def \_monthw \#1 \#2 \#3 \#4 \#5 \#6 \#7 {%
\_sdef{\mt:m1:\#1}{\#2}\_sdef{\mt:m2:\#1}{\#3}\_sdef{\mt:m3:\#1}{\#4}\%
\_sdef{\mt:m4:\#1}{\#5}\_sdef{\mt:m5:\#1}{\#5}\_sdef{\mt:m6:\#1}{\#5}\%
\_monthwB \#1\}
\_def \_monthwB \#1 \#2 \#3 \#4 \#5 \#6 \#7 {%
\_sdef{\mt:m7:\#1}{\#2}\_sdef{\mt:m8:\#1}{\#3}\_sdef{\mt:m9:\#1}{\#4}\%
\_sdef{\mt:m10:\#1}{\#5}\_sdef{\mt:m11:\#1}{\#5}\_sdef{\mt:m12:\#1}{\#5}\%
}\
\_monthw en January February March April May June
\_monthw cs ledna února března dubna května června
\_monthw sk januára februára marca apríla mája júna
\_monthw it gennaio febbraio marzo aprile maggio giugno
\_monthw pl sierpnia września września października listopada grudnia
\_sdef{\mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
\_sdef{\mt:today:cs}{\_the\_day. \_mtext{m\_the\_month} \_the\_year}
\_slet{\mt:today:sk}{\mt:today:cs}
\_def \_today{\_mtext{today}}
\_public \_today ;
\end{verbatim}

Quotes should be tagged by \"\langle\text\rangle\" and \'\langle\text\rangle\' if \texttt{\{iso-code\}quotes} is declared at beginning of the document (for example \texttt{\enquotes}). If not, then the control sequences \" and \' are undefined. Remember, that they are used in another meaning when \texttt{\oldaccents} command is used. The macros \texttt{\"} and \’ are not defined as \texttt{\protected} because we need their expansion when \texttt{\outlines} are created.

User can declare quotes by \texttt{\quoteschars}\texttt{\langle\text\rangle\langle\text\rangle}\texttt{\langle\text\rangle\langle\text\rangle}, where \langle\text\rangle\ldots\langle\text\rangle are normal quotes and \langle\text\rangle\ldots\langle\text\rangle are alternative quotes. or use \texttt{\altquotes} to swap between meaning of these two types of quotes. \texttt{\enquotes, \csquotes, \dequotes, \frquotes} etc. are defined here.

\begin{verbatim}
\_def \_enquotes {\_quoteschars \"\'}
\_def \_csquotes {\_quoteschars \"\’}
\_def \_frquotes {\_quoteschars \"«»}
\_let \_plquotes = \_frquotes
\_let \_esquotes = \_frquotes
\_let \_grquotes = \_frquotes
\_let \_ruquotes = \_frquotes
\_let \_itquotes = \_frquotes
\_let \_skquotes = \_csquotes
\_let \_dequotes = \_csquotes
\_sdef{\mt:today:en}{\_mtext{m\_the\_month} \_the\_day, \_the\_year}
\_sdef{\mt:today:cs}{\_the\_day. \_mtext{m\_the\_month} \_the\_year}
\_slet{\mt:today:sk}{\mt:today:cs}
\_def \_today{\_mtext{today}}
\_public \_today ;
\end{verbatim}
The \quoteschars\langle\text{lqq}\rangle\langle\text{rq}\rangle defines \text{"}" and \text{'}" as \text{\qqA}\text{\qqB}\langle\text{lqq}\rangle\langle\text{rq}\rangle and \text{"}" as \text{\qqA}\text{\qqB}\langle\text{lq}\rangle\langle\text{rq}\rangle. \text{\qqA}\text{\qqB}\langle\text{lqq}\rangle\langle\text{rq}\rangle defines \text{"}" and \text{'}" as expandable macros in outline mode. We want to process the common cases: \text{"}\text{"}" or \text{'\text{'\text{'}. This is reason why the sub-verbatim mode is used when first character is \text{"}" in the parameter.

The \text{"}" is defined as \text{\qqA}\text{\qqB}\langle\text{lqq}\rangle\langle\text{rq}\rangle and \text{'}" as \text{\qqA}\text{\qqB}\langle\text{lq}\rangle\langle\text{rq}\rangle. The \text{\qqA}\text{\qqB}\langle\text{clqq}\rangle\langle\text{crqq}\rangle defines \text{"}" and \text{\qqA}\text{\qqB}\langle\text{clq}\rangle\langle\text{crq}\rangle as expandable macros in normal mode and as expadable macros in outline mode. We want to well process the common cases: \text{"}`&`" or \text{"}`{`". This is reason why the quotes parameter is read in verbatim mode and retokenized again by \text{\scantextokens}. We want to allow to quote the quotes mark itself by \text{"}`{\text{"}"}. This is reason why the sub-verbatim mode is used when first character is \text{"}" in the parameter.

\text{"}" is defined as \text{\qqA}\text{\qqB}\langle\text{lqq}\rangle\langle\text{rq}\rangle and \text{'}" as \text{\qqA}\text{\qqC}\langle\text{lq}\rangle\langle\text{rq}\rangle. The \text{\qqB}\langle\text{clqq}\rangle\langle\text{crqq}\rangle runs \text{\qqB}\langle\text{text}\rangle.

\text{\codesdecl} \uv \{Miscellaneous <2020-05-22}\} % preloaded in format
\text{\useOpTeX} and \text{\useoptex} are declared as \text{\relax}.

\text{\def\lastpage} and \text{\totalpages} get the information from the \text{\currpage}. The \text{\Xpage} from .ref file sets the \text{\currpage}.

\text{\def\totalpages} \{\text{\openref\ea\lastpageA}\text{\currpage}\}
\text{\def\lastpage} \{\text{\openref\ea\lastpageB}\text{\currpage}\}
\text{\def\lastpageA} \{\text{\#1}\}
\text{\def\lastpageB} \{\text{\#2}\}
\text{\def\currpage} \{\text{0}()??\}
\text{\public\lastpage\totalpages} ;

We need \uv, \text{\cqqq}, \text{\lfqq}, \text{\frqq}, \text{\uslang}, \text{\ehyph} \text{\chyph}, \text{\shyph}, for backward compatibility with \text{\csplain}. Codes are set according to Unicode, because we are using Czech only in Unicode when \text{Lua\TeX} is used.

\text{\chardef\clqq=8222 \chardef\crqq=8220}
\text{\chardef\lfqq=171 \chardef\frqq=187}
\text{\chardef\promile=8240}

\begin{verbatim}
\text{\codesdecl} \uv \{Miscellaneous <2020-05-22}\} % preloaded in format
\text{\useOpTeX} and \text{\useoptex} are declared as \text{\relax}.

\text{\def\lastpage} and \text{\totalpages} get the information from the \text{\currpage}. The \text{\Xpage} from .ref file sets the \text{\currpage}.

\text{\def\totalpages} \{\text{\openref\ea\lastpageA}\text{\currpage}\}
\text{\def\lastpage} \{\text{\openref\ea\lastpageB}\text{\currpage}\}
\text{\def\lastpageA} \{\text{\#1}\}
\text{\def\lastpageB} \{\text{\#2}\}
\text{\def\currpage} \{\text{0}()??\}
\text{\public\lastpage\totalpages} ;

We need \uv, \text{\cqqq}, \text{\lfqq}, \text{\frqq}, \text{\uslang}, \text{\ehyph} \text{\chyph}, \text{\shyph}, for backward compatibility with \text{\csplain}. Codes are set according to Unicode, because we are using Czech only in Unicode when \text{Lua\TeX} is used.

\text{\chardef\clqq=8222 \chardef\crqq=8220}
\text{\chardef\lfqq=171 \chardef\frqq=187}
\text{\chardef\promile=8240}
\end{verbatim}
The \_letfont was used in Cyrillic instead of \fontlet.

\_def\uv#1{\clqq#1\crqq}

\_let\uslang=\enlang \_let\ehyph=\enlang
\_let\chyph=\cslang \_let\shyph=\sklang
\_let\csUnicode=\csPatt \_let\czUnicode=\csPatt \_let\skUnicode=\skPatt

The \_letfont was used in Cyrillic instead of \fontlet.

\_let \_letfont = \_fontlet

Non breaking space in Unicode.

\let \^a0=~

TikZ needs these funny control sequences.

\_def\OPmacversion{OpTeX}

We don’t want to read opmac.tex unless \input opmac is specified.

\_def \_lipsum {\
{\_long\_def\ProvidesFile##1[##2]##3{\_ifx\_par##3\_relax\_else \_ea##3\_fi}\_tmpnum=0
\_def\NewLipsumPar{\_advance\_tmpnum by1
\_afterassignment\_negativermnm \_sxdef{lips:\_the\_tmpnum}}%
\_opinput {lipsum.ltd.tex}%
\_global\_let \_lipsum=\_reallipsum
\)
\_lipsum}

2.38 Lua code embedded to the format

The file optex.lua is loaded into the format in optex.ini as byte-code and initialized by \everyjob, see section 2.1.

The file implements part of the functionality from luatexbase namespace, nowadays defined by L\AT\TEX kernel. luatexbase deals with modules, allocators and callback management. Callback management is a nice extension and is actually used in \TeX. Other functions are defined more or less just to suit luaotfload’s use.

2.38 Lua code embedded to the format

The file optex.lua is loaded into the format in optex.ini as byte-code and initialized by \everyjob, see section 2.1.

The file implements part of the functionality from luatexbase namespace, nowadays defined by L\AT\TEX kernel. luatexbase deals with modules, allocators and callback management. Callback management is a nice extension and is actually used in \TeX. Other functions are defined more or less just to suit luaotfload’s use.
For a `\chardef`, `\countdef`, etc., `csname` return corresponding register number. The responsibility of providing a `\XXdef` name is on the caller.

```lua
function register_number(name)
    return token.create(name).index
end
```

**ALLOCATORS**

```lua
alloc = alloc or {}
```

An attribute allocator in Lua that cooperates with normal OpTeX allocator.

```lua
local attributes = {}
local attribute_max = register_number("_maiattribute")
function alloc.new_attribute(name)
    local cnt = tex.count["_attributealloc"] + 1
    if cnt > attribute_max then
        tex.error("No room for a new attribute")
    else
        tex.setcount("global", "_attributealloc", cnt)
        texio.write_nl("log", `'..name..'"\attribute"..tostring(cnt))
        attributes[name] = cnt
        return cnt
    end
end
```

**CALLBACKS**

```lua
callback = callback or {}
```

Save `callback.register` function for internal use.

```lua
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.

```lua
local callback_functions = {}
```

Table that maps callback name to a list of descriptions of its added functions. The order corresponds with `callback_functions`.

```lua
local callback_description = {}
```

Table used to differentiate user callbacks from standard callbacks. Contains user callbacks as keys.

```lua
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).

```lua
local default_functions = {}
```

Table that maps standard (and later user) callback names to their types.
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",
build_page_insert = "exclusive",
pre_linebreak_filter = "list",
linebreak_filter = "exclusive",
append_to_vlist_filter = "exclusive",
post_linebreak_filter = "reverselist",
hpack_filter = "list",
vpack_filter = "list",
hpack_quality = "list",
vpack_quality = "list",
process_rule = "exclusive",
pre_output_filter = "list",
hyphenate = "simple",
ligaturing = "simple",
kerning = "simple",
insert_local_par = "simple",
mlist_to_hlist = "exclusive",

-- information reporting
pre_dump = "simple",
start_run = "simple",
stop_run = "simple",
start_page_number = "simple",
stop_page_number = "simple",
show_error_hook = "simple",
show_error_message = "simple",
show_lua_error_hook = "simple",
start_file = "simple",
stop_file = "simple",
call_edit = "simple",
finish_synctex = "simple",
wrapup_run = "simple",

-- pdf related
finish_pdffile = "data",
finish_pdfpage = "data",
page_order_index = "data",
process_pdf_image_content = "data",

-- font related
define_font = "exclusive",
glyph_not_found = "exclusive",
glyph_info = "exclusive",

-- undocumented
glyph_stream_provider = "exclusive",
}
Return a list containing descriptions of added callback functions for specific callback.

```lua
function callback.callback_descriptions(name)
    return callback_description[name] or {}
end

local valid_callback_types = {
    exclusive = true,
    simple = true,
    data = true,
    list = true,
    reverselist = true,
}

Create a user callback that can only be called manually using `call_callback`. A default function is only needed by "exclusive" callbacks.

```lua
function callback.create_callback(name, cbtype, default)
    if callback_types[name] then
        err("cannot create callback '"..name.."' - it already exists")
    elseif not valid_callback_types[cbtype] then
        err("cannot create callback '"..name.. '' with invalid callback type '"..cbtype.."'")
    elseif ctype == "exclusive" and not default then
        err("unable to create exclusive callback '"..name.."', default function is required")
    end

    callback_types[name] = cbtype
    default_functions[name] = default or nil
    user_callbacks[name] = true
end

Add a function to the list of functions executed when callback is called. For standard luatex callback a proxy function that calls our machinery is registered as the real callback function. This doesn’t happen for user callbacks, that are called manually by user using `call_callback` or for standard callbacks that have default functions – like `mlist_to_hlist` (see below).

```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
    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

local fn = table.remove(callback_functions[name], index)

if #descriptions == 0 then
  -- Delete the list entirely to allow easy checking of "truthiness".
  callback_functions[name] = nil
  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
end
return fn, description
end

helper iterator generator for iterating over reverselist callback functions

local function reverse ipairs(t)
  local i, n = #t + 1, 1
  return function()
    i = i - 1
    if i >= n then
      return i, t[i]
    end
  end
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.

function callback.call_callback(name, ...)
  local cbtype = callback_types[name]
  -- either take added functions or the default function if there is one
  local functions = callback_functions[name] or {default_functions[name]}
  if cbtype == nil then
    err("cannot call callback ", name, ": no such callback exists")
  elseif cbtype == "exclusive" then
    -- only one function, atleast default function is guaranteed by
    -- create_callback
    return functions[1](...)
  elseif cbtype == "simple" then
    -- call all functions one after another, no passing of data
    for _, fn in ipairs(functions) do
      fn(...) end
    return
  elseif cbtype == "data" then
    -- pass data (first argument) from one function to other, while keeping
    -- other arguments
    local data = (...)
    for _, fn in ipairs(functions) do
      data = fn(data, select(2, ...))
    end
    return data
  end
-- list and reverselist are like data, but "true" keeps data (head node)
-- unchanged and "false" ends the chain immediately
local iter
if cbtype == "list" then
    iter = ipairs
else if cbtype == "reverselist" then
    iter = reverse_ipairs
end
local head = (...)
local new_head
local changed = false
for _, fn in iter(functions) do
    new_head = fn(head, select(2, ...))
    if new_head == false then
        return false
    elseif new_head ~= true then
        head = new_head
        changed = true
    end
end
return not changed or head
end

Create "virtual" callbacks pre/post_mlist_to_hlist_filter by setting mlist_to_hlist callback. The default behaviour of mlist_to_hlist is kept by using a default function, but it can still be overriden 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)
    new_head = callback.call_callback("mlist_to_hlist", head, ...)
    if new_head == false then
        node.flush_list(head)
        return nil
    elseif new_head ~= true then
        head = new_head
    end
    -- 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
end)
```

Compatibility with \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,
    callbacktypes = {}
}
```
2.39 Printing documentation

The \printdoc{filename}(space) and \printdoctail{filename}(space) commands are defined after the file doc.opm is loaded by \load[doc].

The \printdoc starts reading of given \filename from the second line. The file is read in the listing mode. The \printdoctail starts reading given \filename from the first occurrence of the \_encode. The file is read in normal mode (like \input{filename}).

The listing mode prints the lines as listing of a code. This mode is finished when first \_doc occurs or first \_endcode occurs. At least two spaces must precede such \_doc. On the other hand, the \_encode must be at the left edge of the line without spaces. If this rule is not met then the listing mode continues.

If the first line or the last line of the listing mode is empty then such lines are not printed. The maximal number of printed lines in the listing mode is \maxlines. Is set to almost infinity (100000). You can set it to a more sensible value. Such setting is valid only for the first following listing mode.

When the listing mode is finished by \_doc then next lines are read in the normal way, but the material between \begtt ... \endtt pair is shifted by three letters left. The reason is that the three spaces of indentation is recommended in the \_doc ... \_cod pair and this shifting is a compensation of this indentation.

The \_cod macro ignores the rest of current line and starts the listing mode again.

When the listing mode is finished by the \_endcode then the \endinput is applied, the reading of the file opened by \printdoc is finished.

You cannot reach the end of the file (without \_endcode) in the listing mode.

The listing mode creates all control sequences which are listed in the index as active link to the main documentation point of such control sequence and prints them in blue. Other text is printed in black.

The main documentation point is denoted by \sequence in red, for example \sequence. The user documentation point is the first occurrence of \sequence, for example \sequence. 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 \sequence) are hyperlinks to the user documentation point.

\begin{Verbatim}
3 \_codedecl \printdoc {Macros for documentation printing <2020-04-28>}
\end{Verbatim}

General declarations.

\begin{Verbatim}
9 \_fontfam[lmfonts]
10 \_hyperlinks \Green \Green
11 \_enlang
12 \_enquotes
\end{Verbatim}

Maybe, somebody needs \seccc or \secccc?

\begin{Verbatim}
18 \_eoldef\seccc#1{\_medskip \_noindent{\_bf#1}\_par\_nobreak\_firstnoindent}
19 \_def\secccc{\_medskip\_noindent $\_bullet$ }
\end{Verbatim}

\end{document} can be redefined.

\begin{Verbatim}
25 \_let\enddocument=\_bye
\end{Verbatim}

Full page of listing causes underfill \vbox in output routine. We need to add a small tolerance.

\begin{Verbatim}
32 \_pgbottomskip=0pt plus10pt minus2pt
\end{Verbatim}

The listing mode is implemented here. The \maxlines is maximal lines of code printed in the listing mode.

\begin{Verbatim}
39 \newcount \maxlines \_maxlines=100000
40 \public \maxlines ;
41 \_eoldef\_cod#1{\_par \_wipepar
42 \_vskip\_parskip \_medskip \_ttskip
43 \_begingroup
44 \_typosize[8/10]
45 \_let\_printverbline=\_printcodeline
46 \_let\_printcodeline=\_printcodeline
\end{Verbatim}
The scanner of the control sequences in the listing mode.

The lines in the listing mode have Yellow background.
\docfile is currently documented file. \printdoc and \printdoctail macros are defined here.

You can do \verb!\vitt{⟨filename⟩} ((⟨from⟩)-(⟨to⟩)) ⟨⟨filename⟩⟩ 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⟩.
\_bbox{\_link[cs:\_tmp]{\Blue}{\tt\_string#1}}\%
\_futurelet\_next\_cslinkA
}

## Index

<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>_aboveliskip</td>
<td>115</td>
</tr>
<tr>
<td>_abovetitle</td>
<td>111, 113</td>
</tr>
<tr>
<td>\activequotes</td>
<td>173</td>
</tr>
<tr>
<td>\activettchar</td>
<td>16–17, 117</td>
</tr>
<tr>
<td>_addcolor</td>
<td>100</td>
</tr>
<tr>
<td>_additcorr</td>
<td>92</td>
</tr>
<tr>
<td>\address</td>
<td>24–25, 163–164</td>
</tr>
<tr>
<td>_addtabitemx</td>
<td>133</td>
</tr>
<tr>
<td>\addto</td>
<td>36, 93</td>
</tr>
<tr>
<td>\adef</td>
<td>16–17, 35</td>
</tr>
<tr>
<td>\adots</td>
<td>77</td>
</tr>
<tr>
<td>\advancepageno</td>
<td>93, 95</td>
</tr>
<tr>
<td>_afteritcorr</td>
<td>92</td>
</tr>
<tr>
<td>_afterload</td>
<td>48</td>
</tr>
<tr>
<td>_allocator</td>
<td>37</td>
</tr>
<tr>
<td>\allowbreak</td>
<td>53</td>
</tr>
<tr>
<td>_backgroundbox</td>
<td>94</td>
</tr>
<tr>
<td>\backgroundpic</td>
<td>126</td>
</tr>
<tr>
<td>\bbchar</td>
<td>71, 85</td>
</tr>
<tr>
<td>\begitems</td>
<td>13–14, 115–116</td>
</tr>
<tr>
<td>\begmulti</td>
<td>19, 136</td>
</tr>
<tr>
<td>\begoutput</td>
<td>93–94, 107</td>
</tr>
<tr>
<td>\begtt</td>
<td>16–17, 94, 117</td>
</tr>
<tr>
<td>_belowliskip</td>
<td>115</td>
</tr>
<tr>
<td>_belowtitle</td>
<td>111, 113</td>
</tr>
<tr>
<td>\bf</td>
<td>8–9, 71, 85</td>
</tr>
<tr>
<td>_bf</td>
<td>8–9, 71, 85</td>
</tr>
<tr>
<td>\bbgroup</td>
<td>35</td>
</tr>
<tr>
<td>\bbfont</td>
<td>8–9, 71, 85</td>
</tr>
<tr>
<td>\bib</td>
<td>20, 139</td>
</tr>
<tr>
<td>\bibmark</td>
<td>137</td>
</tr>
<tr>
<td>\bibnum</td>
<td>105, 137</td>
</tr>
<tr>
<td>_bibskip</td>
<td>140</td>
</tr>
<tr>
<td>_bibtexhook</td>
<td>45</td>
</tr>
<tr>
<td>\big</td>
<td>76</td>
</tr>
<tr>
<td>\Big</td>
<td>76</td>
</tr>
<tr>
<td>\bigbreak</td>
<td>51</td>
</tr>
<tr>
<td>\bigg</td>
<td>76</td>
</tr>
<tr>
<td>\Bigg</td>
<td>76</td>
</tr>
<tr>
<td>\biggl</td>
<td>76</td>
</tr>
<tr>
<td>\Biggl</td>
<td>76</td>
</tr>
<tr>
<td>\biggm</td>
<td>76</td>
</tr>
<tr>
<td>\Biggm</td>
<td>76</td>
</tr>
<tr>
<td>\biggr</td>
<td>76</td>
</tr>
<tr>
<td>\Biggr</td>
<td>76</td>
</tr>
<tr>
<td>\bigskip</td>
<td>51</td>
</tr>
<tr>
<td>\Black</td>
<td>98</td>
</tr>
<tr>
<td>\Blue</td>
<td>21, 98</td>
</tr>
<tr>
<td>\bmod</td>
<td>79</td>
</tr>
<tr>
<td>\boldify</td>
<td>93</td>
</tr>
<tr>
<td>\boldmath</td>
<td>9, 71–73, 83</td>
</tr>
<tr>
<td>_boldunimath</td>
<td>83</td>
</tr>
<tr>
<td>\boxlines</td>
<td>163</td>
</tr>
<tr>
<td>\bp</td>
<td>50</td>
</tr>
<tr>
<td>_bp</td>
<td>50</td>
</tr>
<tr>
<td>_bprinta</td>
<td>143</td>
</tr>
<tr>
<td>_bprintb</td>
<td>143</td>
</tr>
<tr>
<td>_bprintc</td>
<td>143</td>
</tr>
<tr>
<td>_bprintv</td>
<td>143</td>
</tr>
<tr>
<td>\bracedparam</td>
<td>49</td>
</tr>
<tr>
<td>\break</td>
<td>51</td>
</tr>
<tr>
<td>\Brown</td>
<td>98</td>
</tr>
<tr>
<td>\balash</td>
<td>35</td>
</tr>
<tr>
<td>\ buildup</td>
<td>79</td>
</tr>
<tr>
<td>_bye</td>
<td>36, 54</td>
</tr>
<tr>
<td>_byehook</td>
<td>36</td>
</tr>
<tr>
<td>\c</td>
<td>53</td>
</tr>
<tr>
<td>_cal</td>
<td>71, 85</td>
</tr>
<tr>
<td>_caption</td>
<td>10–11, 114</td>
</tr>
<tr>
<td>_cases</td>
<td>79</td>
</tr>
<tr>
<td>_catalogexclude</td>
<td>70</td>
</tr>
<tr>
<td>_catalogmathsample</td>
<td>70</td>
</tr>
<tr>
<td>_catalogonly</td>
<td>70</td>
</tr>
<tr>
<td>_catalogsample</td>
<td>70</td>
</tr>
<tr>
<td>_catcode</td>
<td>49</td>
</tr>
<tr>
<td>_cdots</td>
<td>77</td>
</tr>
<tr>
<td>_centerline</td>
<td>51</td>
</tr>
<tr>
<td>_chap</td>
<td>10–11, 17–18, 49, 110, 112</td>
</tr>
<tr>
<td>_chapfont</td>
<td>110</td>
</tr>
<tr>
<td>_chapx</td>
<td>111</td>
</tr>
<tr>
<td>_chyp</td>
<td>24, 173</td>
</tr>
<tr>
<td>_circle</td>
<td>127</td>
</tr>
<tr>
<td>_circleparams</td>
<td>47</td>
</tr>
<tr>
<td>_cite</td>
<td>12, 19–20, 137</td>
</tr>
<tr>
<td>_citeA</td>
<td>138</td>
</tr>
<tr>
<td>_citeborder</td>
<td>12, 105</td>
</tr>
<tr>
<td>_clipincircle</td>
<td>23, 129</td>
</tr>
<tr>
<td>_clipinpath</td>
<td>129</td>
</tr>
<tr>
<td>_clipinovale</td>
<td>23, 129</td>
</tr>
<tr>
<td>_cliq</td>
<td>173</td>
</tr>
<tr>
<td>_cmykcolordef</td>
<td>100</td>
</tr>
<tr>
<td>_cmyktorgb</td>
<td>99</td>
</tr>
<tr>
<td>_code</td>
<td>31–32, 50</td>
</tr>
<tr>
<td>_code16–17, 117</td>
<td>117</td>
</tr>
<tr>
<td>_codedecl</td>
<td>31–32</td>
</tr>
<tr>
<td>_colnum</td>
<td>133</td>
</tr>
<tr>
<td>_colorrcrop</td>
<td>99</td>
</tr>
<tr>
<td>_colordef</td>
<td>21, 97–99, 101</td>
</tr>
<tr>
<td>_colordeffin</td>
<td>99</td>
</tr>
<tr>
<td>_colorstackpop</td>
<td>99</td>
</tr>
<tr>
<td>_colorstackpush</td>
<td>99</td>
</tr>
<tr>
<td>_colorstackset</td>
<td>99</td>
</tr>
<tr>
<td>_colsep</td>
<td>45</td>
</tr>
<tr>
<td>_commentchars</td>
<td>17, 118, 120</td>
</tr>
<tr>
<td>_commoncolordef</td>
<td>100</td>
</tr>
<tr>
<td>_completepage</td>
<td>93–94</td>
</tr>
<tr>
<td>_compoundchars</td>
<td>156</td>
</tr>
<tr>
<td>_compoundcharscs</td>
<td>156</td>
</tr>
<tr>
<td>_compoundcharsen</td>
<td>156</td>
</tr>
<tr>
<td>_cong</td>
<td>79</td>
</tr>
<tr>
<td>_corrmsizes</td>
<td>72</td>
</tr>
<tr>
<td>_crl</td>
<td>15, 132, 134</td>
</tr>
<tr>
<td>_crlci</td>
<td>15, 130, 133–134</td>
</tr>
<tr>
<td>_crlil</td>
<td>15, 134</td>
</tr>
<tr>
<td>_crlili</td>
<td>15, 130, 134</td>
</tr>
<tr>
<td>_crlp</td>
<td>15, 130, 134</td>
</tr>
<tr>
<td>_crq</td>
<td>173</td>
</tr>
<tr>
<td>_cs</td>
<td>36</td>
</tr>
<tr>
<td>_cs</td>
<td>167</td>
</tr>
<tr>
<td>_cskip</td>
<td>10, 114</td>
</tr>
<tr>
<td>_cslang</td>
<td>23–24, 168</td>
</tr>
<tr>
<td>_csplain</td>
<td>167</td>
</tr>
<tr>
<td>_csquotes</td>
<td>24, 172</td>
</tr>
<tr>
<td>_ctablelist</td>
<td>48</td>
</tr>
<tr>
<td>_cufamily</td>
<td>64, 67</td>
</tr>
<tr>
<td>_currence</td>
<td>103, 106, 173</td>
</tr>
<tr>
<td>_currstyle</td>
<td>81</td>
</tr>
<tr>
<td>_currV</td>
<td>63, 67</td>
</tr>
<tr>
<td>_curvar</td>
<td>8–9, 58, 60–61, 68</td>
</tr>
<tr>
<td>_cyan</td>
<td>21, 98</td>
</tr>
<tr>
<td>_d</td>
<td>53</td>
</tr>
<tr>
<td>_ddlinedata</td>
<td>133</td>
</tr>
<tr>
<td>_ddots</td>
<td>77</td>
</tr>
<tr>
<td>_decadigits</td>
<td>50</td>
</tr>
<tr>
<td>_defaultfontfeatures</td>
<td>70</td>
</tr>
<tr>
<td>_defaultitem</td>
<td>14, 45, 116</td>
</tr>
<tr>
<td>_delang</td>
<td>23, 168</td>
</tr>
<tr>
<td>_dequotes</td>
<td>24, 172</td>
</tr>
<tr>
<td>_dest</td>
<td>13, 105</td>
</tr>
<tr>
<td>_destactive</td>
<td>105</td>
</tr>
<tr>
<td>_destheight</td>
<td>105</td>
</tr>
<tr>
<td>_displaylines</td>
<td>80</td>
</tr>
<tr>
<td>_do</td>
<td>39</td>
</tr>
<tr>
<td>_do</td>
<td>39</td>
</tr>
<tr>
<td>_dobystyle</td>
<td>81</td>
</tr>
<tr>
<td>_doc</td>
<td>31–32, 50</td>
</tr>
<tr>
<td>_docompound</td>
<td>157</td>
</tr>
<tr>
<td>_doloadmath</td>
<td>82</td>
</tr>
<tr>
<td>_doresizefont</td>
<td>57, 66</td>
</tr>
<tr>
<td>_dorezetfompes</td>
<td>57</td>
</tr>
<tr>
<td>_doreizeunfont</td>
<td>57, 66, 70</td>
</tr>
<tr>
<td>_doshadow</td>
<td>129</td>
</tr>
<tr>
<td>_dosorting</td>
<td>158</td>
</tr>
<tr>
<td>_dostyle</td>
<td>50</td>
</tr>
<tr>
<td>_dospecials</td>
<td>50</td>
</tr>
<tr>
<td>_dospereject</td>
<td>51, 93</td>
</tr>
<tr>
<td>_doteq</td>
<td>79</td>
</tr>
<tr>
<td>_dotfill</td>
<td>54</td>
</tr>
<tr>
<td>_dots</td>
<td>53</td>
</tr>
</tbody>
</table>