OpTeX

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

Petr Olšák, 2020

http://petr.olsak.net/optex

Op\TeX{} is Lua\TeX{} format with Plain \TeX{} and \OPmac{}. Only Lua\TeX{} engine is supported.

Op\TeX{} 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 Op\TeX{} is:

• Op\TeX{} keeps the simplicity (like in Plain \TeX{} and \OPmac{} macros).
• There is no old obscurities concerning with various 8-bit encodings and various engines.
• Op\TeX{} provides a powerful Fonts Selection System (for Unicode font families, of course).
• Op\TeX{} supports hyphenations of all languages installed in your \TeX{} system.
• All features from \OPmac{} macros are copied. For example sorting words in the Index\footnote{The \OPmac{} package is a set of simple additional macros to Plain \TeX{}. It enables users to take advantage of \LaTeX{} functionality but keeps Plain \TeX{} simplicity. See http://petr.olsak.net/opmac-e.html for more information about it.}, reading \file{.bib} files directly\footnote{All these features are implemented by \TeX{} macros, no external program is needed.}, syntax highlighting\footnote{All these features are implemented by \TeX{} macros, no external program is needed.}, 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 Op\TeX{} and primitives (\texttt{\textbackslash 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 Op\TeX{} macros into your macro file and do changes of these macros here. This is significant difference from \LaTeX{} or Con\TeXt{}, 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 Op\TeX{} 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 Op\TeX{} macros, understand them an modify them.

Op\TeX{} 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 Op\TeX{}. There are many hyperlinks from one part to second and vice versa.

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

### 1 User documentation

1.1 Starting with Op\\TeX

1.2 Page layout

1.2.1 Setting the margins

1.2.2 Concept of default page

1.2.3 Footnotes and marginal notes

1.3 Fonts

1.3.1 Font families

1.3.2 Font sizes

1.3.3 Typesetting math

1.4 Typical elements of document

1.4.1 Chapters and sections

1.4.2 Another numbered objects

1.4.3 References

1.4.4 Hyperlinks, outlines

1.4.5 Lists

1.4.6 Tables

1.4.7 Verbatim

1.5 Autogenerated lists

1.5.1 Table of contents

1.5.2 Making the index

1.5.3 Bib\\TeXing

1.6 Graphics

1.6.1 Colors

1.6.2 Images

1.6.3 PDF transformations

1.6.4 Ovals, circles

1.6.5 Putting images and texts wherever

1.7 Others

1.7.1 Using more languages

1.7.2 Pre-defined styles

1.7.3 Loading other macro packages

1.7.4 Lorem ipsum dolor sit

1.7.5 Logos

1.7.6 The last page

1.7.7 Use Op\\TeX

1.8 Summary

1.9 Compatibility with Plain \TeX

### 2 Technical documentation

2.1 The main initialization file

2.2 Concept of name spaces of control sequences

2.2.1 Prefixing internal control sequences

2.2.2 Name space of control sequences for users

2.2.3 Macro files syntax

2.2.4 Name spaces for package writers

2.2.5 Summary about rules for external macro files published for Op\\TeX

2.2.6 The implementation of the name spaces

2.3 pdf\\TeX initialization
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.27.2 Listings with syntax highlighting</td>
<td>114</td>
</tr>
<tr>
<td>2.28 Graphics</td>
<td>118</td>
</tr>
<tr>
<td>2.29 The <code>\table</code> macro</td>
<td>123</td>
</tr>
<tr>
<td>2.30 Balanced multi-columns</td>
<td>126</td>
</tr>
<tr>
<td>2.31 Citations, bibliography</td>
<td>127</td>
</tr>
<tr>
<td>2.31.1 Macros for citations and bibliography preloaded in the format</td>
<td>127</td>
</tr>
<tr>
<td>2.31.2 The <code>\usebib</code> command</td>
<td>130</td>
</tr>
<tr>
<td>2.31.3 The <code>usebib.optim</code> macro file loaded when <code>\usebib</code> is used</td>
<td>131</td>
</tr>
<tr>
<td>2.31.4 Usage of the <code>bib-iso690</code> style</td>
<td>134</td>
</tr>
<tr>
<td>2.31.5 Implementation of the <code>bib-iso690</code> style</td>
<td>141</td>
</tr>
<tr>
<td>2.32 Sorting and making Index</td>
<td>145</td>
</tr>
<tr>
<td>2.33 Footnotes and marginal notes</td>
<td>151</td>
</tr>
<tr>
<td>2.34 Styles</td>
<td>153</td>
</tr>
<tr>
<td>2.34.1 <code>\report</code> and <code>\letter</code> styles</td>
<td>153</td>
</tr>
<tr>
<td>2.34.2 <code>\slides</code> style for presentations</td>
<td>154</td>
</tr>
<tr>
<td>2.35 Logos</td>
<td>157</td>
</tr>
<tr>
<td>2.36 Multilingual support</td>
<td>157</td>
</tr>
<tr>
<td>2.36.1 Lowercase, uppercase codes</td>
<td>157</td>
</tr>
<tr>
<td>2.36.2 Hyphenations</td>
<td>158</td>
</tr>
<tr>
<td>2.36.3 Multilingual phrases and quotation marks</td>
<td>161</td>
</tr>
<tr>
<td>2.37 Other macros</td>
<td>162</td>
</tr>
<tr>
<td>2.38 Printing documentation</td>
<td>163</td>
</tr>
</tbody>
</table>

Index 167
Chapter 1
User documentation

1.1 Starting with OpTEX

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

```
optex document
```

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

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

A minimal document should be

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

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

A somewhat larger example with common settings should be:

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

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

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

1.2 Page layout

1.2.1 Setting the margins

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

```
\margins/⟨pg⟩ ⟨fmt⟩ ⟨⟨left⟩, ⟨right⟩, ⟨top⟩, ⟨bot⟩⟩⟨unit⟩
```

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

Parameters are:

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

---

\(^1\) This is a technical limitation of LuaTEX for fonts downloaded in formats: only 8bit fonts can be preloaded.
Each of the parameters \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 \texttt{\hsize} is set. Else \texttt{\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 (\texttt{\vsize} instead \texttt{\hsize} is used). Examples:

\begin{verbatim}
\margins/1 a4 (,,,)mm % \texttt{\hsize, \vsize} untouched, 
% typesetting area centered
\margins/1 a4 (,2,,)cm % right margin set to 2cm
% \texttt{\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.

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

The \texttt{\langle fmt \rangle} can be also in the form \langle width \rangle,\langle height \rangle\langle unit \rangle where \langle unit \rangle is optional. If it is missing then \langle unit \rangle after margins specification is used. For example:

\begin{verbatim}
\margins/1 (100,200) (7,7,7,7)mm
\magscale[1414] \margins/1 a4 (,,,)mm
\end{verbatim}

declares the paper 100×200 mm with all four margins 7 mm. The spaces before and after \langle fmt \rangle parameter are necessary.

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

\begin{verbatim}
\margins/2 a5 (22,17,19,21)mm
\magscale[1414] \margins/1 a4 (,,,)mm
\end{verbatim}

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

1.2.2 Concept of default page

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

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

The distance between the \texttt{\headline} and the top of the page body is given by the \texttt{\headlinedist} register. The distance between bottom of the page body and the \texttt{\footline} is given by \texttt{\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 files parameters.opm and output.opm.

1.2.3 Footnotes and marginal notes

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

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

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

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

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

1.3 Fonts

1.3.1 Font families

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

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

\footnote{You can declare \fnotenumglobal if you want footnotes numbered in whole document from one or \fnotenumpages if you want footnotes numbered at each page from one. Default setting is \fnotenums chapters}
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 \textit{font modifiers} applicable in such font family are listed on the terminal: \texttt{(\caps, \cond\text{ for example})}. And there are four basic \textit{variant selectors} (\texttt{\rm, \bf, \it, \bi}). The usage of variant selectors is the same as in Plain \TeX{}: \texttt{\{\it italics text\}}, \texttt{\{\bf bold text\}} etc.

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

There is one special variant selector \texttt{\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{\rm ↔ \it} and \texttt{\bf ↔ \bi} is kept by the \texttt{\em} macro (emphasis text). It switches from current \texttt{\rm} to \texttt{\it}, from current \texttt{\it} to \texttt{\rm}, from current \texttt{\bf} to \texttt{\bi} and from current \texttt{\bi} to \texttt{\bf}. The italics correction \texttt{/} is inserted automatically, if needed.

Example:

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

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

\subsection{Font sizes}

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

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

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

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

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

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

The \texttt{\typoscale} command does scaling in respect to current values by default. If you want to do it in respect to main values, type \texttt{\scalemain} immediately before \texttt{\typoscale} command.
\texttt{\textbackslash typosize[12/14.4]} % first usage in document, sets main values internally
\texttt{\textbackslash typosize[15/18]} % bigger font
\texttt{\textbackslash scalemain \textbackslash typoscale[800/800]} % reduces from main values, no from current.

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

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

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

Hundreds math symbols and operators like in AMSTeX are accessible. For example \texttt{\textbackslash alpha α, \textbackslash geq ≥, ∑ \textbackslash sum Σ, \textbackslash sphericalangle ≢, \textbackslash bumpeq ≲}. See AMSTeX manual for complete list of symbols.

The following math alphabets are available:

\texttt{\mit % mathematical variables} \hspace{1cm} abc−xyz, ABC−XYZ
\texttt{\it % text italics} \hspace{1cm} abc−xyz, ABC−XYZ
\texttt{\rm % text roman} \hspace{1cm} abc−xyz, ABC−XYZ
\texttt{\cal % normal calligraphics} \hspace{1cm} ABC−XYZ
\texttt{\script % script} \hspace{1cm} ABC−XYZ
\texttt{\frak % fracture} \hspace{1cm} abc−xyz, ABC−XYZ
\texttt{\bbchar % double stroked letters} \hspace{1cm} ABC−XYZ
\texttt{\bf % sans serif bold} \hspace{1cm} abc−xyz, ABC−XYZ
\texttt{\bi % sans serif bold slanted} \hspace{1cm} abc−xyz, ABC−XYZ

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 the file \texttt{unimath−codes.opm} where Unicode math variants of \texttt{\bf} and \texttt{\bi} selectors are defined.

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

You can use \texttt{\textbackslash mathbox[⟨text⟩]} inside math mode. It behaves as \texttt{\textbackslash hbox[⟨text⟩]} (i.e. the \texttt{⟨text⟩} is printed in horizontal non-math mode) but the size of the \texttt{⟨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 document can be divided into chapters (\chap), sections (\sec), subsections (\secc) and they can be titled by \tit command. The parameters are separated by the end of current line (no braces are used):

\tit Document title \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

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

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

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

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

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

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

1.4.2 Another numbered objects

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

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

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

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

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

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

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:

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

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

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

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

### 1.4.3 References

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

\(^3\) This feature can be changed, see the section 2.25 in the technical documentation.
\sec{beatle} About Beatles

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

\label{pythagoras}
$$ a^2 + b^2 = c^2 \eqmark $$

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

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

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

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

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

1.4.4 Hyperlinks, outlines

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

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

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

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

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

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

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

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

There are “low level” commands to create the links. You can specify the destination of
the internal link by \dest[(type):(label)]. The active text linked to the \dest can be created
by $\texttt{\link[(type):(label)]{(text)}}$. The (type) parameter is one of the toc, pg, cite, ref or another special for your purpose. These commands create internal links only when \hyperlinks is declared.

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

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

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

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

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

1.4.5 Lists

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

\begin{verbatim}
\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
\end{verbatim}

For example:

\begin{verbatim}
\begitems
* First idea
* Second idea in subitems:
  \begitems \style i
  * First sub-idea
  * Second sub-idea
  * Last sub-idea
\enditems
\end{verbatim}

$^4$ More exactly, there are the same rules as for \code command, see section 1.4.7.
produces:

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

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

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

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

1.4.6 Tables

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

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

generates the following result:

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

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

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

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

The command \cr used in the ⟨data⟩ part of the table is generally known. It marks end row of the table. Moreover Op\TeX defines following similar commands:
• \texttt{\textbackslash cr} \ldots the end of the row with a horizontal line after it.
• \texttt{\textbackslash crli} \ldots like \texttt{\textbackslash cr} but the horizontal line doesn’t intersect the vertical double lines.
• \texttt{\textbackslash crlli} \ldots like \texttt{\textbackslash crli} but horizontal line is doubled.
• \texttt{\textbackslash crlp\{\langle list\rangle\}} \ldots like \texttt{\textbackslash crli} but the lines are drawn only in the columns mentioned in comma separated \langle list\rangle of their numbers. The \langle list\rangle can include \langle from\rangle-\langle to\rangle declarators, for example \texttt{\textbackslash crlp\{1-3,5\}} is equal to \texttt{\textbackslash crlp\{1,2,3,5\}}.

The \texttt{\textbackslash tskip\langle dimen\rangle} command works like the \texttt{\textbackslash noalign{\textbackslash vskip\langle dimen\rangle}} after \texttt{\textbackslashcr*} 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.

\begin{verbatim}
\everytable={} % code used in \vbox before table processing
\thisstable={} % code used in \vbox, it is removed after using it
\tabiteml={\enspace} % left material in each column
\tabitemr={\enspace} % right material in each column
\tabstrut={\strut} % strut which declares lines distance in the table
\tablinespace=2pt % additional vert. space before/after horizontal lines
\vvkern=1pt % space between lines in double vertical line
\hhkern=1pt % space between lines in double horizontal line
\end{verbatim}

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

The \texttt{\textbackslash frame\{\langle text\rangle\}} makes a frame around \langle 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{\table{|c||l||r|}{ \crl
  \mspan3\{c\}\{\bf Title\} \crl \noalign{\kern\hhkern}\crli
  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{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 the section 2.29.

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

Many tips about tables can be seen on the site http://petr.olsak.net/optex/optex-tricks.html.
1.4.7 Verbatim

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

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

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

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

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

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

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

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

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

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

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

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

```
\code{\text, \%#} ... prints \text, %#
\code{@{..}*~\$} ... prints @{..}*~\$ without escaping, but you can
escape these characters too, if you want.
\code{a \ b} ... two spaces between a b, second one must be escaped
\code{xy\{z} ... xy{z ... unbalanced brace must be escaped
\code{\~\M} ... prints \~\M, the second \ must be escaped
```
You can print verbatim listing from external files by the \texttt{\verbatiminput} command. Examples:

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

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

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

The \verbatiminput can be controlled by \texttt{\everytt}, \texttt{\ttindent} just like in \texttt{\begtt...\endtt}. The \texttt{\begtt...\endtt} pair or \verbatiminput can be used for listings of codes. Automatic syntax highlighting is possible, for example \texttt{\begtt \hisyntax{C}} activates colors for C programs. Or \verbatiminput \hisyntax{HTML} (-) file.html can be used for HTML or XML codes. \TeX{} implements C, Python, \TeX{}, HTML and XML syntax highlighting. More languages can be declared, see the section 2.27.2.

1.5 Autogenerated lists

1.5.1 Table of contents

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

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

\verbatiminput \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 \texttt{\regmacro{\langle case-toc\rangle}{\langle case-mark\rangle}{\langle case-outline\rangle}}. The parameters are applied locally in given cases. The \texttt{\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:

\verbatiminput

% default value of \texttt{\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 \texttt{\makeindex} macro. No external program is needed, the alphabetical sorting are done inside \TeX{} at macro level.

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

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

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

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

or:

Now we'll deal with the \iid resistor.

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

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

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

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

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

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

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

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

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

The \ii or \iid commands can be preceded by \iitype ⟨letter⟩, then such reference (or more references generated by one \ii) has specified type. The page numbers of such references should be formatted specially in the index. \TeXX 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 \TeXXbook 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. \TeXX 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, \TeXX primitive, math selector etc. But there is another possibility
to create such complex index entry. Use “pure equivalent” in the \ii parameter and map this equivalent to the real word which is printed in the index by the \iis command. Example:

The \ii chi quadrat $\chi$-quadrat method is

If the \ii relax `relax` command is used then \TeX/ is relaxing.

\iis chi quadrat {\$\chi$-quadrat}
\iis relax {\$\text{\relax}$}

The \iis \{equivalent\} \{\langle\text{text}\rangle\} 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 BibT\TeX{}Xing

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

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

The \rcite\[(\langle\text{labels}\rangle)\] creates the same list as \cite\[(\langle\text{labels}\rangle)\] but without the outer brackets. Example: \[\rcite\[\text{\facitelb}\], pg.-13\] creates \[4, pg.13\].

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

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

\def\cite\[#1\]\{\rcite\[#1\]\}\% \cite\[<\text{label}>\] creates (27)
\def\cite\[#1\]\{$^\text{\rcite\[#1\]}$}\% \cite\[<\text{label}>\] 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\[(\langle\text{label}\rangle)\] commands.
• By \usebib\{/\langle\text{type}\rangle\} \{\langle\text{style}\rangle\} \langle\bib-base\rangle 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\[(\langle\text{label}\rangle)\] command.

\bib\[\text{\facittst}\]\ P. Olšák. \{\it Typografický systém \TeX{}\}
\hphantom{\text{\facittst}\[\text{\facitbn}\]}\quad 269-s. Praha: CSTUG, 1995.

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

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

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

1.6 Graphics

1.6.1 Colors

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}}

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 Op\TeX. For example \Chocolate is Chocolate1, \ChocolateB is Chocolate2 etc.

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

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

You can define your color by a linear combination of previously defined colors using \colordef. For example:
\colordef \myCyan {.3\Green + .5\Blue} % 30 % green, 50 % blue, 20% white
\colordef \DarkBlue \{\Blue + .4\Black} % Blue mixed with 40 % of black
\colordef \myGreen\{\Cyan+\Yellow\} % exact the same as \Green
\colordef \MyColor {.3\Orange+.5\Green+.2\Yellow}

The linear combination is done in CMYK 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.

\(^5\) Printed output is more equal to the monitor preview specially if you are using ICC profile for your printer.
The following example defines the macro for the colored text on the colored background.

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

The \coloron can be defined as follows:

\def\coloron#1#2#3{% 
\setbox0=\hbox{{#2#3}}% 
\leavevmode \rlap{#1\strut \vrule width\wd0}\box0 
}

\coloron\Yellow\Brown\{The brown text on the yellow background\}

1.6.2 Images

The \inspic \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 the img subdirectory. Note: the directory name must end by / in the \picdir declaration.

Inkscape\(^6\) is able to save a picture to PDF and labels of the picture to another file\(^7\). This second file should be read by \TeX in order to print labels in the same font as document font. Op\TeX supports this feature by \inkinspic \langle filename\rangle.pdf command. It reads and displays both: PDF image and labels generated by Inkscape.

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

1.6.3 PDF transformations

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

Op\TeX provides two special transformation macros \pdfscale and \pdfrotate:

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

These macros simply calls the properly \pdfsetmatrix command.

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

\(^6\) A powerfull and free wysiwyg editor for creating vector graphics.
\(^7\) Chose “Omit text in PDF and create LaTeX file” option.
First: \pdfsave \pdfrotate{30}\pdfscale{-2}{2}\rlap{text1}\pdfrestore
% text1 is scaled two times and it is reflected about vertical axis
% and next it is rotated by 30 degrees left.

second: \pdfsave \pdfscale{-2}{2}\pdfrotate{30}\rlap{text2}\pdfrestore
% text2 is rotated by 30 degrees left then it is scaled two times
% and reflected about vertical axis.

third: \pdfsave \pdfrotate{-15.3}\pdfsetmatrix{2 0 1.5 2}\rlap{text3}\%
\pdfrestore % first slanted, then rotated by 15.3 degrees right

This gives the following result. First: `text1`
second: `text2`
third: `text3`

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

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

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

1.6.4 Ovals, circles

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

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

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

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

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

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

The macros \clipinoval ⟨x⟩ ⟨y⟩ ⟨width⟩ ⟨height⟩ {⟨text⟩} and \clipincircle (with the same parameters) print the ⟨text⟩ when a clipping path (oval or 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 does’n not change the position of the current point. Assume coordinate system with origin in the current point. Then \puttext{⟨x⟩ ⟨y⟩ {⟨text⟩}} puts the text at the coordinates ⟨x⟩, ⟨y⟩. More exactly the left edge of its baseline is at that position.

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

1.7 Others

1.7.1 Using more languages

OpTEX prepares hyphenation patterns for all languages if such patterns are available in your \TeX{} system. Only USenglish patterns (original from Plain \TeX) are preloaded. Hyphenation patterns of all other 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) engb(UKenglish) it(Italian) ia(Interlingua) id(Indonesian) cs(Czech) sk(Slovak)
de(nGerman) fr(French) pl(Polish) cy(Welsh) da(Danish) es(Spanish) sl(Slovenian) fi(Finnish) hy(Hungarian)
tr(Turkish) et(Estonian) eu(Esperanto) ga(Irish) nb(Bokmal) nn(Nynorsk) pt(Portuguese) ro(Romanian) hr(Croatian)
zh(Pinyin) is(Icelandic) csb(Upper Sorbian) af(Afrikaans) gl(Galician) km(Kurmanji) trk(Turkmen) ia(Latin)
lac(classicalLatin) lal(liturgicalLatin) elm(monoGreek) elp(Greek) grc(ancientGreek) ca(Catalan) co(Coptic)
nm(Mongolian) sa(Sanskrit) ru(Russian) uk(Ukrainian) hy(Armenian) an(Extensive Latin) ita(Italian)
la(Latin) lac(Old Latin) lt(Lithuanian) nl(Malayalam) mr(Marathi) or(Oriya) pa(Punjabi) ta(Tamil) te(Telugu)

For compatibility with \texttt{e-plain}, there are macros \ehyph, \chyph, \shyph which are equivalent to \enlang, \csclang and \sklang.

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

OpTEX generates three phrases used for captions and titles in technical articles or books: “Chapter”, “Table” and “Figure”. These phrases need to be known in used language and it depends on the previously used language selectors \lang{⟨iso-code⟩}. OpTEX declares these words only for few languages: Czech, German, Spanish, French, 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 \lang{} commands as shown in the section 2.36.3.

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

If you declare \lang{⟨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 \lang{⟨iso-code⟩}. OpTEX declares quotes only for Czech, German, Spanish, French, Greek, Italian, Polish, Russian, Slovak and English (\csquotes, \dequotes, ..., \enquotes). You can simply define your own quotes as shown in languages.opm file. The \" is used for quotes 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 \lang{⟨isocode⟩}quotes\altquotes{} in such case.
1.7.2 Pre-defined styles

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

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

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

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

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

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

1.7.3 Loading other macro packages

You can load more macro packages by `\input{⟨file-name⟩}` or by `\load{⟨file-names⟩}`. The first case is TeX primitive command, it can be used in the alternative old syntax `\input ⟨filename⟩⟨space⟩` too. The second case allows to specify a comma separated list of included files. Moreover, the `\load` macro 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 occurence 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 document is structured to more files, 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
\texttt{\textbackslash lorem[\{number\}]} or \texttt{\textbackslash 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 \texttt{\textbackslash lipsum} macro is equivalent to \texttt{\textbackslash lorem}. Example \texttt{\textbackslash 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:

\begin{quote}
We are using \TeX/ because it is cool. \LaTeX/ is better than \TeX.
\end{quote}

### 1.7.6 The last page

The number of the last page (it may be different from number of pages) is expanded by \texttt{\textbackslash 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”:

\begin{quote}
\texttt{\textbackslash footline}={\hss \texttt{\textbackslash fixedrm} \texttt{\textbackslash folio}/\texttt{\textbackslash lastpage} \texttt{\textbackslash hss}}
\end{quote}

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

### 1.7.7 Use Op\TeX

The command \texttt{\textbackslash useOpTex} (or \texttt{\textbackslash useoptex}) does nothing in Op\TeX but it causes an error (undefined control sequence) when another format is used. You can put it as the first command in your document:

\begin{quote}
\texttt{\textbackslash useOpTex} % we are using Op\TeX format, no LaTeX :)
\end{quote}

### 1.8 Summary

\begin{quote}
\texttt{\textbackslash tit Title (terminated by end of line)}
\texttt{\textbackslash chap Chapter Title (terminated by end of line)}
\texttt{\textbackslash sec Section Title (terminated by end of line)}
\texttt{\textbackslash secc Subsection Title (terminated by end of line)}
\texttt{\textbackslash maketoc} % table of contents generation
\texttt{\textbackslash ii item1,item2} % insertion the items to the index
\texttt{\textbackslash makeindex} % the index is generated
\texttt{\textbackslash label [labname]} % link target location
\texttt{\textbackslash ref [labname]} % link to the chapter, section, subsection, equation
\texttt{\textbackslash poref [labname]} % link to the page of the chapter, section, ...
\texttt{\textbackslash caption/t} % a numbered table caption
\texttt{\textbackslash caption/f} % a numbered caption for the picture
\texttt{\textbackslash eequmark} % a numbered equation
\texttt{\textbackslash begitems} % start list of the items
\texttt{\textbackslash enditems} % end of list of the items
\texttt{\textbackslash begtt} % start verbatim text
\texttt{\textbackslash endtt} % end verbatim text
\texttt{\textbackslash activettchar X} % initialization character X for in-text verbatim
\texttt{\textbackslash code} % another alternative for in-text verbatim
\texttt{\textbackslash verbinput} % verbatim extract from the external file
\texttt{\textbackslash begmulti num} % start multicolumn text (num columns)
\texttt{\textbackslash endmulti} % end multicolumn text
\end{quote}

25
1.9 Compatibility with Plain \TeX

All macros of Plain \TeX are re-written in Op\TeX. 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 _\hbox). 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 Op\TeX have and use only prefixed form. User should use unprefix forms, but prefixed forms are accessible too, because the _ is set as a letter category code globally (in macro files and in user document too). User should re-define unprefix forms of control sequences with no worries that something internal will be broken (only the sequence \par cannot be re-defined without internal change of \TeX behavior because it is hard-coded in \TeXs tokenization processor).

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

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

The text accents macros \", \', \v, \u, \=, \^, \., \H, \~, \` are undefined8 in Op\TeX. Use real letters like á, í, ř 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 1 in 1 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 seans.

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

Technical documentation

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

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

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

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

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

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

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

2.1 The main initialization file

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

```
%% This is part of OpTeX project, see http://petr.olsak.net/optex
%% OpTeX ini file
%% Petr Olsak <project started from: Jan. 2020>
\catcode `\{=1 % left brace is begin-group character
\catcode `\}=2 % right brace is end-group character
\catcode `\$=3 % dollar sign is math shift
\catcode `\&=4 % ampersand is alignment tab
\catcode `\#=6 % hash mark is macro parameter character
\catcode `\^=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 `\^^a0=13 % non breaking space in Unicode
\catcode 127=12 % normal character
\optexversion{Beta 0.12 Apr 2020}
\fmtname{OpTeX}
\newlinechar=`
\ifx\directlua\undefined
\message{This format is based only on LuaTeX, use luatex -ini optex.ini\newline}
\endinput \fi
```

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

The \optexversion and \fmtname are defined.

```
\def\optexversion{Beta 0.12 Apr 2020}
\def\fmtname{OpTeX}
```

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

```
\newlinechar="\^^J
```

(Kommentare)
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 \everyjob register is initialized and the format is saved by the \dump command.
Concept of name spaces of control sequences

Prefixing internal control sequences

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

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

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

OpTEX defines all new macros as prefixed. For public usage of such macros we need to set non-prefixed version. This is done by \public <list of control sequences> ;

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

At the end of each code segment in \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.

Name space of control sequences for users

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

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

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

  \int_a^b \ldots \text{ is interpreted as } \int _a^b
  
  \max_M \ldots \text{ is interpreted as } \max _M
  
  \alpha_{ij} \ldots \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 \_⟨pkg⟩_{‘word’} is intended for package writers as internal macros for a package with ⟨pkg⟩ identifier, see section 2.2.4.

The single letter control sequences like \%, \$, ^ etc. are not used in internal macros. 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 OpTEX macros is stored in one file with .opm extension (means OPtex Macros). Your local macros should be in normal *.tex file.

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

\_codedecl \sequence {Name <version>}

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

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

2.2.4 Name spaces for package writers

Package writer should use internal names in the form \_⟨pgk⟩_⟨sequence⟩, where ⟨pgk⟩ is a package label.

For example: \_qr_utfstring from qrcode.opm package.

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

The \public macro does \let\foo = \_⟨pkg⟩_foo when \_namespace{⟨pkg⟩} is declared. And the \private macro does reverse operation to it. Example: you can define \def\macro{...} and then set it to the user name space by \_public \macro;.

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

2.2.5 Summary about rules for external macro files published for OpTEX

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

- Don’t use a control sequences from user name space in the macro bodies if there is not explicit and documented reason to do this.
- Don’t declare control sequences in the user name space if there is not explicit and documented reason to do this.
- Use control sequences from OpTEX and primitive name space in read only mode if there is not explicit and documented reason to redefine them.
- Use \_⟨pkg⟩_⟨name⟩ for your internal macros or \_⟨name⟩ if the \_namespace{⟨pkg⟩} is declared. See section 2.2.4.
- Use \_load (or better: \_load) for loading more external macros if you need them. Don’t use \_input explicitly in such cases. The reason is: the external macro file is not loaded twice if another macro or the user needs it explicitly too.
- Use \_codedecl as your first command in the macro file and \_endcode to close the text of macros.
- Use \_doc ... \_cod pairs for documenting the code pieces 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.

\footnote{We have not adatped the idea from expl3 language:)}
2.2.6 The implementation of the name spaces

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

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

\ea is useful shortcut for \expandafter. We recommend to use always the private form of \_ea because there is high probability that \ea will be redefined by the user.
\public \langle \text \rangle \langle \text \rangle ; \ldots ; \text \let \langle \text \rangle = \_ \langle \text \rangle
\private \langle \text \rangle \langle \text \rangle ; \ldots ; \text \let \langle \text \rangle = \text \langle \text \rangle
\xargs \langle \text \rangle \langle \text \rangle \langle \text \rangle ; \ldots ; \text \langle \text \rangle \langle \text \rangle

Each macro file should begin with \ codedecl \macro \{\langle \text \rangle\}. If the \macro is defined already then the \endinput protects to read such file more than one times. Else the \langle \text \rangle is printed to the terminal and the file is read.
The \endcode is defined as \endinput in the optex.ini file. \wterm \{\langle \text \rangle\} prints \langle \text \rangle to the terminal and to the \log file (as in plain \TeX{}).

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 macro \_namespace \{\langle pkg label \rangle\} for package writers, see section 2.2.4.
2.3 pdf\TeX{} initialization

Common pdf\TeX{} primitives equivalents are declared here. Initial values are set.

```latex
3 \_codedecl \pdfprimitive {Lua\TeX{} initialization code <2020-02-21>} \% preloaded in format
4 \_let \pdfpagewidth \pagewidth\_protect
5 \_let \pdfpageheight \pageheight\_protect
6 \_let \pdfadjustspacing \adjustspacing\_protect
7 \_let \pdfprotrudechars \protrudechars\_protect
8 \_let \pdfnoligatures \ignoreligaturesinfont\_protect
9 \_let \pdffontexpand \expandglyphsinfont\_protect
10 \_let \pdfcopyfont \copyfont\_protect
11 \_let \pdfxform \saveboxresource\_protect
12 \_let \pdflastxform \lastsavedboxresourceindex\_protect
13 \_let \pdfximage \saveimageresource\_protect
14 \_let \pdflastximage \lastsavedimageresourceindex\_protect
15 \_let \pdflastximagepages \lastsavedimageresourcepages\_protect
16 \_let \pdfoutput \outputmode\_protect
17 \_let \pdfdraftmode \draftmode\_protect
18 \_let \pdfpxdimen \pxdimen\_protect
19 \_let \pdfinsertht \insertht\_protect
20 \_let \pdfnormaldeviate \normaldeviate\_protect
21 \_let \pdfuniformdeviate \uniformdeviate\_protect
22 \_let \pdfsetrandomseed \setrandomseed\_protect
23 \_let \pdfrandomseed \randomseed\_protect
24 \_let \pdfprimitive \primitive\_protect
25 \_let \ifpdfprimitive \ifprimitive\_protect
26 \_let \ifpdfabsnum \ifabsnum\_protect
27 \_let \ifpdfabsdim \ifabsdim\_protect

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

\pdfpagewidth \pdfpageheight \pdfadjustspacing \pdfprotrudechars
\pdfnoligatures \pdffontexpand \pdfcopyfont \pdfxform \pdflastxform
\pdfximage \pdflastximage \pdflastximagepages \pdfoutput \pdfdraftmode \pdfpxdimen
\pdfinsertht \pdfnormaldeviate \pdfuniformdeviate \pdfsetrandomseed
\pdfrandomseed \pdfprimitive \ifpdfprimitive \ifpdfabsnum \ifpdfabsdim
\_directlua {tex.enableprimitives('pdf',{'tracingfonts'})}
\_protected\_def \_pdftexversion {\_numexpr 140\_relax}
\_def \_pdftexrevision {7}
\_protected\_def \_pdffeedback \_numexpr\_pdffeedback lastlink\_relax
\_protected\_def \_pdffeedback \_numexpr\_pdffeedback retval\_relax
\_protected\_def \_pdffeedback \_numexpr\_pdffeedback lastobj\_relax
\_protected\_def \_pdffeedback \_numexpr\_pdffeedback lastannot\_relax
\_def \_pdffeedback \_numexpr\_pdffeedback xformname\_relax
\_outputmode=1
\_sdef \_pdffeedback \_numexpr\_pdffeedback creationdate\_relax
\_def \_pdffeedback \_numexpr\_pdffeedback fontname\_relax
\_def \_pdffeedback \_numexpr\_pdffeedback fontobjnum\_relax
```
\def \pdffontsize {\pdffeedback fontsize}
\def \pdfpageref {\pdffeedback pageref}
\def \pdfcolorstackinit {\pdffeedback colorstackinit}
\protected\def \pdfliteral {\pdfextension literal}
\protected\def \pdfcolorstack {\pdfextension colorstack}
\protected\def \pdfsetmatrix {\pdfextension setmatrix}
\protected\def \pdfsave {\pdfextension save\_relax}
\protected\def \pdfrestore {\pdfextension restore\_relax}
\protected\def \pdfobj {\pdfextension obj }
\protected\def \pdffobj {\pdfextension refobj }
\protected\def \pdfannot {\pdfextension annot }
\protected\def \pdfstructlink {\pdfextension structlink }
\protected\def \pdfendlink {\pdfextension endlink\_relax}
\protected\def \pdfoutline {\pdfextension outline }
\protected\def \pdfdest {\pdfextension dest }
\protected\def \pdfthread {\pdfextension thread }
\protected\def \pdfstartthread {\pdfextension startthread }
\protected\def \pdfendthread {\pdfextension endthread\_relax}
\protected\def \pdfinfo {\pdfextension info }
\protected\def \pdfcatalog {\pdfextension catalog }
\protected\def \pdfnames {\pdfextension names }
\protected\def \pdfincludechars {\pdfextension includechars }
\protected\def \pdffontattr {\pdfextension fontattr }
\protected\def \pdfmapfile {\pdfextension mapfile }
\protected\def \pdfmapline {\pdfextension mapline }
\protected\def \pdftrailer {\pdfextension trailer }
\protected\def \pdfglyphtounicode {\pdfextension glyphtounicode }
\protected\edef \pdfcompresslevel {\pdfvariable compresslevel}
\protected\edef \pdfobjcompresslevel {\pdfvariable objcompresslevel}
\protected\edef \pdfdecimaldigits {\pdfvariable decimaldigits}
\protected\edef \pdfgamma {\pdfvariable gamma}
\protected\edef \pdfimageresolution {\pdfvariable imageresolution}
\protected\edef \pdfimageapplygamma {\pdfvariable imageapplygamma}
\protected\edef \pdfimagegamma {\pdfvariable imagegamma}
\protected\edef \pdfimagehicolor {\pdfvariable imagehicolor}
\protected\edef \pdfimageaddfilename {\pdfvariable imageaddfilename}
\protected\edef \pdfpkresolution {\pdfvariable pkresolution}
\protected\edef \pdfinclusioncopyfonts {\pdfvariable inclusioncopyfonts}
\protected\edef \pdfinclusionerrorlevel {\pdfvariable inclusionerrorlevel}
\protected\edef \pdfgentounicode {\pdfvariable gentounicode}
\protected\edef \pdfpagebox {\pdfvariable pagebox}
\protected\edef \pdfminorversion {\pdfvariable minorversion}
\protected\edef \pdfuniqueresname {\pdfvariable uniqueresname}
\protected\edef \pdfhorigin {\pdfvariable horigin}
\protected\edef \pdfvorigin {\pdfvariable vorigin}
\protected\edef \pdflinkmargin {\pdfvariable linkmargin}
\protected\edef \pdfdestmargin {\pdfvariable destmargin}
\protected\edef \pdfthreadmargin {\pdfvariable threadmargin}
\protected\edef \pdfpagesattr {\pdfvariable pagesattr}
\protected\edef \pdfpageattr {\pdfvariable pageattr}
\protected\edef \pdfxformattr {\pdfvariable xformattr}
\protected\edef \pdfxforresources {\pdfvariable xforresources}
\protected\edef \pdfpkmode {\pdfvariable pkmode}
\public
\pdftexversion \pdftexrevision \pdftrunclink \pdfreval \pdflastobj
\pdflastannot \pdfxformname \pdfcreationdate \pdfonclick \pdffontnum
\pdffontsize \pdfpageref \pdfcolorstackinit \pdfliteral \pdfcolorstack
\pdfsetmatrix \pdfsave \pdfrestore \pdfobj \pdfrestore \pdfannot
\pdftrunclink \pdftrunclink \pdfdest \pdfthread \pdfthread
\pdfinfo \pdfcatalog \pdfnames \pdfincludechars \pdfannot
\pdfmapfile \pdfmapline \pdftrailer \pdfglyphtounicode \pdfcompresslevel
\pdfobj \pdfobj \pdfimageresolution \pdfgamma \pdfimagegamma \pdfimagehicolor
\pdfimageaddfilename \pdfimageaddfilename \pdfimageaddfilename
\pdfpkresolution \pdfpkresolution \pdfpkresolution
\pdfinclusioncopyfonts \pdfinclusioncopyfonts \pdfinclusioncopyfonts
\pdfinclusionerrorlevel \pdfinclusionerrorlevel \pdfinclusionerrorlevel
\pdfgentounicode \pdfgentounicode \pdfgentounicode
\pdfpagebox \pdfpagebox \pdfpagebox
\pdfminorversion \pdfminorversion \pdfminorversion
\pdfuniqueresname \pdfuniqueresname \pdfuniqueresname
\pdfhorigin \pdfhorigin \pdfhorigin
\pdfvorigin \pdfvorigin \pdfvorigin
\pdflinkmargin \pdflinkmargin \pdflinkmargin
\pdfdestmargin \pdfdestmargin \pdfdestmargin
\pdfthreadmargin \pdfthreadmargin \pdfthreadmargin
\pdfpagesattr \pdfpagesattr \pdfpagesattr
\pdfpageattr \pdfpageattr \pdfpageattr
\pdfxformattr \pdfxformattr \pdfxformattr
\pdfxforresources \pdfxforresources \pdfxforresources
\pdfpkmode \pdfpkmode \pdfpkmode
2.4 Basic macros

We define first bundle of basic macros.

\begin{verbatim}
\_codedecl \sdef {Basic macros for OpTeX <2020-02-14>} % loaded in format
\bgroup,
\let \_bgroup={ \_let \_egroup=}
\def \_empty {}\def \_space { }\def \_null {\_hbox{}}\def \_wlog {\_immediate\_write-1} % write on log file (only)
\spublic \bgroup \_egroup \_empty \_space \_null \_wlog;
\bslash is "normal backslash" with category code 12. \nbb and \pcent are double backslash and normal \%, they should be used in lua codes, for example.
\edef \_bslash {\_csstring\}
\edef \_nbb {\_bslash\_bslash}
\edef \_pcent {\_csstring\%}
\spublic \bslash \nbb \pcent;
\adef {⟨char⟩}{⟨body⟩} puts the ⟨char⟩ as active character and defines it as ⟨body⟩. You can declare a macro with parameters too. For example \adef @#1{...$1...}.
\cs {⟨text⟩} is only a shortcut to \csname ⟨text⟩\endcsname, but you need one more \_ea if you need to get the real control sequence \textbackslash ⟨text⟩.
\trycs {⟨csname⟩} {⟨text⟩} expands to ⟨csname⟩ if it is defined else to the ⟨text⟩.
\addto \macro {⟨text⟩} adds ⟨text⟩ to your \macro, which must be defined.
\opwarning {⟨text⟩} prints warning on the terminal and to the log file.
\end{verbatim}
2.5 Allocators for \TeX{} registers

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

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

User can use \texttt{\dimen0} to \texttt{\dimen200} and similarly for \texttt{\skip}, \texttt{\muskip}, \texttt{\box} and \texttt{\toks} directly. User can use \texttt{\count20} to \texttt{\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 form 254 to 201 using \texttt{\newinsert}.

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

\begin{verbatim}
% redefine maximal allocation index as variable
\_maicount = \maicount % first value is top of the array

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

\def\usecount #1[#2]{% \usecount \foo[2]
  \count \numexpr#1+#2 \relax
}
\end{verbatim}

The limits are set first.

Each allocation macro needs its own counter.
The common allocation macro \_allocator \{sequence\} \{\type\} \{\primitive\} is defined. This idea was used in classical plain \TeX{} by Donald Knuth too but the macro from plain \TeX{} seems to be more complicated.

 alloc.opm

\begin{verbatim}
41 \def \_allocator #1#2#3{%
42 \_global \_advance \_cs\{#2alloc\}by1
43 \_ifnum \_cs\{#2alloc\}>\_cs\{mai#2\}%
44 \_errmsg\{No room for a new \_ea\_string \_csname #2\_endcsname\}%
45 \_else
46 \_global\#3#1=\_cs\{#2alloc\}%
47 \_wlog\{\_string#1=\_ea\_string \_csname #2\_endcsname\_the\_cs\{#2alloc\}\}%
48 \_fi
49 }

def \newcount \newdimen \newskip \newmuskip \newbox \newtoks \newread \newwrite \newmath ;
\end{verbatim}

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

 alloc.opm

\begin{verbatim}
93 \def \newattribute #1{%
94 \advance \insertalloc by-1
95 \_errmsg\{No room for a new \_ea\_string \_insert\}%
96 \_else
97 \_global \chardef#1=\_insertalloc
98 \_wlog\{\_string#1=\_string \_insert \_the\_insertalloc\}%
99 \_fi
100 }
\end{verbatim}

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

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.

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

Recommendation: it is better to use private variants of \_foreach and \_fornum. When the user writes \input tikz then \foreach macro is redefined! The private variants use \_do separator instead \do separator.
The \texttt{\foreach} and \texttt{\fornum} macros can be nested and arbitrary combined. When they are nested then use \texttt{\#1} for the variable of nested level, \texttt{\###1} for the variable of second nested level etc. Example:

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

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

2.6.3 Is-macros

There are a collection of macros \texttt{\isempty}, \texttt{\istokempty}, \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}

or

\begin{verbatim}
\issomething \langle \params \rangle \iffalse \langle \codeB \rangle \else \langle \codeA \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 \iftrue or \iffalse into unseparated parameter and repeat it because we need to remove an optional space before this command.

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

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

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

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

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

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

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.
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.

2.7.1 Primitive registers

The primitive registers with the same default value as in plain TeX follow:
\_thickmuskip=5mu plus 5mu

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

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

### 2.7.2 Plain TeX registers

Declared registers used in plain TeX

\_newskip\_smallskipamount \_smallskipamount=3pt plus 1pt minus 1pt
\_newskip\_medskipamount \_medskipamount=6pt plus 2pt minus 2pt
\_newskip\_bigskipamount \_bigskipamount=12pt plus 4pt minus 4pt
\_newskip\_normalbaselineskip \_normalbaselineskip=12pt
\_newdimen\_normallineskip \_normallineskip=1pt
\_newdimen\_normallineskiplimit \_normallineskiplimit=0pt
\_newdimen\_jot \_jot=3pt
\_newcount\_interdisplaylinepenalty \_interdisplaylinepenalty=100
\_newcount\_interfootnotelinepenalty \_interfootnotelinepenalty=100
\_def\_normalbaselines{\_lineskip=\_normallineskip
\_baselineskip=\_normalbaselineskip \_lineskiplimit=\_normallineskiplimit}
\_def\_frenchspacing{\_sfcode`\.=1000 \_sfcode`\!=1000 \_sfcode`\:=1000 \_sfcode`\;=1000 \_sfcode`\,=1000 }
\_def\_nonfrenchspacing{\_sfcode`\.=3000 \_sfcode`\!=3000 \_sfcode`\:=1500 \_sfcode`\;=1250 }
\_public \_normalbaselines \_frenchspacing \_nonfrenchspacing
\_smallskipamount \_medskipamount \_bigskipamount \_normalbaselineskip \_normallineskip \_normallineskiplimit \_jot \_interdisplaylinepenalty \_interfootnotelinepenalty ;

### 2.7.3 Different settings than in plain TeX

Default “baseline setting” is for 10 pt fonts (like in plain TeX). But \_typosize and \_typoscale macros re-declare it if another font size is used.

The \_nonfrenchspacing is not set by default because the author of OpTeX is living in the Europe. If you set \_enlang hyphenation patterns then \_nonfrenchspacing is set.

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 1in, 1in coordinates) and set default page dimensions as A4, no letter.

\_emergencystretch=20pt \% we want to use third pass of aparagraph building algorithn
\_clubpenalty=10000 \% after first line of paragraph
\_widowpenalty=10000 \% before last line of paragraph
\_showboxbreadth=150 \% for tracing boxes
\_showboxdepth=7
\_errorcontextlines=15
\_tracinglostchars=2 \% missing chracter warnings on terminal too
\_outputmode=1 \% PDF ouput
\_pdfvorigin=0pt \% orgin is 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
If you insist on plain \TeX values of these parameters then you can call the \plaintexsetting macro.

\begin{verbatim}
\def\plaintexsetting{%
  \emergencystretch=0pt \clubpenalty=150 \widowpenalty=150
  \pdforigin=lin \pdfhorigin=lin \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 \printsec and \printsecc from sections.opm file to your macro file and re-define them.

Notice for OPmac users: there is important difference: all “string-like” parameters are token lists in 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 the same data using protected control sequence. If user re-defines such unprotected control sequence (because he/she does know about it) then nothing bad happens.

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

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

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

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

The \everyintt token list does similar work but acts in the in-line verbatim text processed by a pair of \activettchar characters or by \code\{\text\}. You can set \everyintt={\Red} for example if you want in-line verbatim in red color.
The \ttline is used in \begtt...\endtt environment or in the code printed by \verbinput. If \ttline is positive or zero, then the verbatim code have numbered lines from \ttline+1. The \ttline register is re-set to new value after a code piece is printed, so next code pieces have numbered lines continuously. If \ttline=-1, then \begtt...\endtt lines are without numbers and \verbinput lines show the line numbers of inputted file. If \ttline<0 then no line numbers are printed.

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

The \ttshift gives the amount of shift of all verbatim lines to right. Despite to the \ttindent, it does not shift the line numbers, only the text.

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

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.

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

The default item mark used between \begitems and \enditems is bullet. The \defaultitem tokens list declare this default item mark.

The \everyitem tokens list is applied in vertical mode at the start of each item.

The \everylist tokens list is applied after group is opened by.

The \ilevel keeps the value of current nesting level of the items list.

The \listskipamount gives vertical skip above and below the items list if \ilevel=1.

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

The \begmulti \endmulti pair creates more columns. The parameter \colsep declares the space between columns. If \n columns are specified then we have \n−1 \colseps and \n columns in total \hsize. This gives definite result of columns width.
Each line in the Table of contents is printed in a group. The \everytocline tokens list is processed here before the internal \_tocl:(num) macro which starts printing the line.

The \bibtxhook 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.

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

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

The \mnotesize is horizontal size of the marginal notes.

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

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

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

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

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

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

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

\h kern gives the space between horizontal lines if they are doubled and \v kern gives the space between such vertical lines.

The output routine uses token list \headline and \footline in the same sense as in plain \TeX. If they are non-empty then \hfil or \hspace must be here because they are used inside \vbox 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 beginning of your document. For example  \ontdef\rmfixed\{\rm},  \ontdef\itfixed\{\it\}.

Then use them in headline and footnote:

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

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

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

The \nextpages tokens list can include settings which will be used at next pages. It is processed at the end of output routine with \globaldefs=1 prefix. The \nextpages is reset to empty after processing. Example of usage:

\headline={} \nextpages={\headline={\fixedrm \firstmark \hfil}}

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

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

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

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

The second bundle of OpTEX macros is here.

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

In order to achieve this, we declare \optexcatcodes catcode table and \plaintexcatodes. They save the commonly used catcode tables. Note that \catcodetable is a part of Lua\TeX{} extension. The catcodetable stack is implemented by OpTEX macros. The \setctable{⟨catcode table⟩} pushes current catode table to the stack and activates catodes from the ⟨catcode table⟩. The \restorectable returns to the saved catodes from the catcode table stack. So, the \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 catodes are managed in the \catcodetable0. If the \setctable is used first (or at the outer level of the stack), then the \catcodetable0 is pushed to the stack and the current table is re-set to the given ⟨catcode table⟩. The numbers of these tables are stacked to the \_ctablelist macro. The \restorectable reads the last saved catcode table number from the \_ctablelist and uses it.

When a special macro is defined with different catodes then \normalcatodes can be used at the end of such definition. The normal catodes are restored. The macro reads catodes from \optexcatodes table and sets it to the main catode table 0.

The \load{⟨filename-list⟩} loads files specified in comma separated ⟨filename-list⟩. The first space (after comma) is ignored using the trick #1#2,: first parameter is unseparated. The \load macro saves the information about loaded files by setting \_load:⟨filename⟩ as a defined macro.
The declarator `\optdef\macro #1\{\langle replacement text\rangle\}` defines the \macro with the optional parameter followed by normal parameters declared in `\langle\params\rangle`. The optional parameter must be used as the first parameter in brackets `[...]`. If it isn’t used then \textit{\langle opt default\rangle} is taken into account. The `\langle replacement text\rangle` can use `\textsc{the opt}` because optional parameter is saved to the \texttt{opt} tokens register. Note the difference from \LaTeX{} concept where the optional parameter is in \texttt{#1}. OpTeX uses #1 as the first normal parameter (if declared).

The \texttt{nospaceafter} ignores the following optional space at expand processor level using the negative \texttt{roman numeral} trick.

The declarator `\eoldef\macro #1\{\langle 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\replacement text\rangle`. The catcode of the \texttt{\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 \texttt{\bracedparam\macro\{\langle parameter\rangle\}} can be used here. The \texttt{\bracedparam} is a prefix which re-sets temporarily the \macro to a \macro with normal one parameter.

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

The `\replstring\macro\{\langle textA\rangle\}\{\langle textB\rangle\}` replaces all occurrences of `\langle textA\rangle` by `\langle textB\rangle` in the \macro body. The \texttt{\replstring} must be defined without parameters. The occurrences of `\langle textA\rangle` are not replaced if they are “hidden” in braces, for example …\{…\langle textA\rangle…\}… . 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\texttt{something}\rangle or `\catcode``\texttt{number}` but these characters `^` or `"` can be set as active (typically by `\activettchar` macro). Nothing problematic happens if re-defined \catcode is used in this case.

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

47
The \removespaces \langle text with spaces \rangle \{} expands to \langle text without spaces \rangle.

The \_ea\ignorept\the\langle\dimen\rangle expands to a decimal number \the\langle\dimen\rangle but without pt unit.

The \ignoreit\langle token \rangle just ignores the \langle token \rangle.

You can use expandable \bp\langle\dimen\rangle converter from \TeX\ \langle\dimen\rangle (or from an expression accepted by \dimexpr primitive) to a decimal value in big points (used as natural unit in the PDF format). So, you can write, for example:

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

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

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

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

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

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

The \_magstephalf is defined with \space, (no \relax), in order to be expandable.

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

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

The \magstep and \magstephalf are defined with \space, (no \relax), in order to be expandable.
Plain TeX basic macros and control sequences. \texttt{\endgraf}, \texttt{\endline}. The \texttt{\verbatimtext{\^^L}} is not defined in OpTeX 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{\asmallbreak}, \texttt{\medbreak}, \texttt{\bigbreak}.

\begin{verbatim}
\_def \_magstep#1\{\_ifcase#1 1000\_or 1200\_or 1440\_or 1728\_or 2074\_or 2488\_fi \_space}
\_public \magstephalf \magstep ;
\end{verbatim}
\_def \_smallbreak \{_par\_ifdim\_lastskip<\_smallskipamount\_removelastskip \penalty-50 \_smallskip \fi\}
\_def \_medbreak \{_par\_ifdim\_lastskip<\_medskipamount\_removelastskip \penalty-100 \_medskip \fi\}
\_def \_bigbreak \{_par\_ifdim\_lastskip<\_bigskipamount\_removelastskip \penalty-200 \_bigskip \fi\}
\public \break \nobreak \allowbreak \filbreak \goodbreak \eject \supereject \dosupereject
\removelastskip \smallbreak \medbreak \bigbreak;

Boxes. \line, \leftline, \rightline, \centerline, \rlap, \llap, \underbar.
\_def \_line \{\hbox to\_hsize\}
\_def \_leftline #1\{\_line{#1\_hss}\}
\_def \_rightline #1\{\_hss#1\}
\_def \_centerline #1\{\_hss#1\_hss\}
\_def \_rlap #1\{\_hbox to0pt{#1\_hss}\}
\_def \_llap #1\{\_hbox to0pt{\_hss#1}\}
\_def \_underbar #1\{$\_setbox0=\_hbox{#1}\_dp0=0pt \_math \_underline{\_box0}\}$
\public \line \leftline \rightline \centerline \rlap \llap \underbar;
The \texttt{\_strutbox} is declared as 10pt size dependent (like in plain \TeX), but the macro \_setbaselineskip (from fonts-opmac.opm) redefines it.
\_newbox\_strutbox
\_setbox\_strutbox=\_hbox{\_vrule height8.5pt depth3.5pt width0pt}
\_def \_strut \{_relax\_ifmmode\_copy\_strutbox\_else\_unhcopy\_strutbox\_fi\}
\public \strutbox \strut;

Alignement. \hidewidth \ialign \multispan.
\_def \_hidewidth \{\_hskip\_hideskip\} % for alignment entries that can stick out
\_def \_ialign\{_everycr={}\_tabskip=\_zoskip \_halign\}
\_newcount\_mscount
\_def \_multispan #1\{_omit \_mscount=#1\_relax
\_loop \_ifnum\_mscount>1 \_spanA \_repeat\}
\_def \_spanA \{_span \_omit \_advance\_mscount by-1\}
\public \hidewidth \ialign \multispan;

Tabbing macros are omitted because they are obsolete.
Indentation and others. \textindent, \item, \itemitem, \narrower, \raggedright, \ttraggedright, \leavevmode.
\_def \_hang \{\_hangindent\_parindent\}
\_def \_textindent #1\{_indent\_llap{#1\_enspace}\_ignorespaces\}
\_def \_item \{_par\_hang\_textindent\}
\_def \_itemitem \{_par\_indent \_hangindent2\_parindent \_textindent\}
\_def \_narrower \{_advance\_leftskip\_parindent
\_advance\_rightskip\_parindent\}
\_def \_raggedright \{_rightskip=0pt plus2em
\_spaceskip=.3333em \_xspaceskip=.5em\_relax\}
\_def \_ttraggedright \{_tt \_rightskip=0pt plus2em\_relax\}
\_def \_leavevmode \{_unhbox\_voidbox\}_begins a paragraph, if necessary
\public \_hang \textindent \item \itemitem \narrower \raggedright \ttraggedright \leavevmode;

Few character codes are set for backward compatibility. But old obscurities (from plain \TeX) based on \texttt{\_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.
\%\chardef\%=`\%
\_let\%=\_pcent % more natural, can be used in lua codes.
\%\chardef\&=`\&
\%\chardef\#=`\#
Accents. The macros \ooalign, \d, \b, \c, \dots, are defined for backward compatibility.

The accents commands like \n, \, \H, etc. are not defined. Use the accented characters directly – it is best solution. But you can use the macro \oldaccents which defines accented macros. Much more usable is to define these control sequences to other purposes.
The last part of plain TeX macros. \hrulefill, \dotfill, \rightarrowfill, \leftarrowfill, \magnification, \bye. Math macros are defined in the math-macros.opm file.

2.10 Preloaded fonts for text mode

Format in luatex can download only non-Unicode fonts. Latin Modern EC is loaded here. These fonts are totally unusable in LuTeX when languages with out of ASCII or ISO-8859-1 alphabets are used (for example Czech). We load only few 8bit fonts here especially for simple testing the format. But, if the user needs to do a more serious work, he/she can use \fontfam macro in order to load a selected font family of Unicode fonts.
We have a dilemma: when the Unicode fonts cannot be preloaded in format then basic font set can be loaded by `\everyjob`. But why to load a set of fonts 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 `\tenrm`, `\tenbf`, `\tenit`, `\tenbi`, `\tentt` are declared as `\public` here but only for backward compatibility. We don’t use them in the Font Selection System. But the protected versions of these control sequences are used in the Font Selection System.

\begin{verbatim}
\_codedecl \tenrm {Latin Modern fonts (EC) preloaded <2020-01-23>} % loaded in format
\% Only few text fonts are preloaded:
\_font\_tenrm=ec-lmr10 % roman text
\_font\_tenbf=ec-lmbx10 % boldface extended
\_font\_tenit=ec-lmri10 % text italic
\_font\_tenbi=ec-lmbxi10 % bold italic
\_font\_tentt=ec-lmtt10 % typewriter
\_{\_tenrm}
\_public \tenrm \tenbf \tenit \tenbi \tentt ;
\end{verbatim}

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

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

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

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

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

```
\setfontsize{at12pt}
\resizethefont
```

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

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

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

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

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

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

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

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

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

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

\example:
\fontdef \bigfont \{\setfontsize{at15pt}\bf\}

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

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

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

2.11.2 The \fontlet declarator

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

\fontlet \(\langle newfont\rangle\) = \(\langle fontswitch\rangle \langle sizespec\rangle\)

\example:
\fontlet \bigfont = \_tenbf at15pt

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

Summary: you can declare font switches:

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

2.11.3 Optical sizes

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

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

2.11.4 Implementation notes

The \setfontsize \{\langle sizespec\rangle\} saves the \langle sizespec\rangle to the \_sizespec macro. The \_optsize value is calculated from the \langle sizespec\rangle. If the \langle sizespec\rangle is in the mag(\coefficient) format then the contents of the \_sizespec macro is re-calculated to the at(\dimen) format using previous \_optsize value.
\_newdimen \_optsize \_optsize=10pt
\_newdimen \_defaultoptsize \_defaultoptsize=10pt
\_newdimen \_lastmagsize
\_def \_setfontsize #1{%
\_edef \_sizespec{#1}%
\_ea \_setoptsize \_sizespec \_relax
\_reloading
\}
\_def \_setoptsize {\_isnextchar a{\_setoptsizeA}{\_isnextchar m{\_setoptsizeC}{\_setoptsizeB}}}
\_def \_setoptsizeA at#1 \_relax{\_optsize=#1 \_relax \_lastmagsize=\_optsize} % at<dimen>
\_def \_setoptsizeB scaled#1 \_relax{\_optsize=\_defaultoptsize \_relax} % scaled<scalenum>
\_def \_setoptsizeC mag#1 \_relax{%
\_ifdim \_lastmagsize>0pt \_optsize=\_lastmagsize \_else \_optsize=\_pdffontsize \_font \_fi
\_optsize=#1 \_optsize
\_lastmagsize=\_optsize
\_edef \_sizespec{at \_the \_optsize}%
\_public \_setfontsize \_defaultoptsize ;

\_resizefont \{\_variant-name\}\{\_font switch\}, for example \_resizefont{bf}\_tenbf resizes the font given by the variant. The variant XX have its font switch \_\tenXX. The \_\doresizefont\_fontswitch is used. It works in TFM mode (\_\doresizetfmfont) or OTF mode (\_\doresizeunifont). In both modes, it does

\_font \_\tenXX = \{\_fontname\} \_\sizespec

The \_\fontname is generated by the \_\fontname \TeX primitive where \_rfontskipat removes the at<dimen> part of the \_\fontname output. The \_\fontname is generated differently in OTF mode, see \_\doresizeunifont macro.

The \_\whatresize is defined as \{\_variant-name\}.

\_fontdef \{\_font switch\}\{\_modifiers\}\{\_variant selector\}\} opens group, runs \{\_modifiers\}\{\_variant selector\}\ (i.e. it runs #2 parameter). The font switch #1 saved in the \_\fontselector macro is re-declared because the variant selector runs the \_\resizefont. Now, we need to keep the current meaning of the font switch #1 but we must leave the opened group. This is done by the \_\keepmeaning macro.

\_fontlet \{\_font switch A\} \{\_font switch B\} \{\_size spec\} does

\_font \_\fontswitch A = \{\_fontname\} \_\size spec

The \_\fontname is extracted using the primitive command \_\\fontname \_\\fontswitch B.
\newcurrfontsize \langle size spec \rangle sets current font size to the \langle size spec \rangle. It is implemented by \fontlet. The font switch of the current font is extracted by \_the\_font because its original meaning is set to “inaccessible” by \TeX\ when \font primitive is started. \resizethefont is implemented by \newcurrfontsize using data from the \_sizespec macro.

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

The implementation detail: The \langle font id \rangle:reg is defined as the \langle optical size data \rangle and all control sequences \_\langle fontname \rangle:reg from this data line has the same meaning because of the \_reversetfm macro. The \_wichtfm expands this data line and apply \_dowichtfm. This macro select the right result from the data line by testing with the current \_optsize value.
Optical sizes data for preloaded 8bit Latin Modern fonts:

```
\_def\_dowhichtfm #1 #2 {%
\_ifdim\_optsize<#2pt \_ea\_ignoretfm\_else \_ea\_dowhichtfm
\_fi
}\_def\_ignoretfm #1\_relax{}
```

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 \curvar and more variant selectors can be declared by the \famvardef macro.
- **font modifiers** (for example \cond, \caps, \setfontsize{⟨size spec⟩}), they are in two types: bulid in (like \setfontsize) or declared modifiers (by by the \moddef macro).
- **family selectors** (for example \Termes, \LMfonts), they are declared typically in the font family files.

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

- The **font switchers** select fonts independent on the font context.
- The **variant selectors** select the font depending on the font context and on the specified variant.
- The **font modifiers** create a change in the font context but they don’t select the font itself.
- The **family selectors** set a family in the font context and resets all font modifiers. They 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 \texttt{\cond} and \texttt{\light}. They can be arbitrary combined (in arbitrary order) and if the font family disposes with all such sub-variants then the desired font is selected (after variant selector is used). On the other hand there are font modifiers which negates the previous font modifier, for example \texttt{\cond}, \texttt{\extend}. You can reset all modifiers to their initial value by the \texttt{\resetmod} command.

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

\begin{verbatim}
⟨family selector⟩ ⟨font modifiers⟩ ⟨variant selector⟩
\end{verbatim}

For example:

\begin{verbatim}
\fontfam [Heros]  % Heros family is active here, default \rm variant.
\fontfam [Termes] % Termes family is active here, default \rm variant.
\end{verbatim}

\begin{verbatim}
{\Heros\caps\cond\it The caps+condensed italics in Heros family is here.}
\end{verbatim}

\begin{verbatim}
The Termes roman is here.
\end{verbatim}

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

You can use the \texttt{\setfontsize \{\textit{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:

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

Much more comfortable way to resize fonts is using OPmac-like command \texttt{\typosize}, \texttt{\typoscale}. These commands prepare the right sizes for math fonts too and re-calculates many internal parameters like \texttt{\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 \texttt{\noloadmath} before you load a first font family.

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

2.12.4 Declaring font commands

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

\begin{verbatim}
\fontdef\langle font switch \rangle \{\langle family selector \rangle \langle font modifiers \rangle \langle variant selector \rangle\}
\end{verbatim}

Such font switches should be used in \texttt{\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: \texttt{\rm}, \texttt{\it}, \texttt{\bf}, etc. It means that the smaller font for notices should be initialized by \texttt{\setfontsize\{at9pt\}\rm} for example. If you want a “notices font selector” then you can do \texttt{\def\noticefont\{\setfontsize\{at9pt\}\rm}. This font selector does not change the \texttt{\baselineskip}. If you want to do this then put different \texttt{\baselineskip} setting to your definition. But you must not forget that the end of group before \texttt{\par} is a typical mistake of \TeX users: the last paragraph is in smaller font but in normal baselineskip, because \texttt{\baselineskip} setting is taken into account when \texttt{\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 \texttt{\{\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:

\begin{verbatim}
\def\titlefont\{\setfontsize\{at14pt\}\bf \let\it\bi\}
\end{verbatim}

\begin{verbatim}
...\{\titlefont \ here \ we \ have \ bold \ 14pt \ font \ and \ \{\it \ here\} \ was \ bold \ 14pt \ italics\}
\end{verbatim}

You can declare a new variant selector by the \texttt{\famvardef} macro. This macro has similar syntax as \texttt{\fontdef}.
\famvardef \langle new variant selector \rangle \{\langle font modifiers \rangle \ \langle variant selector \rangle\}

The \langle new variant selector \rangle 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 \langle new variant selector \rangle is used in normal text then it does following steps: pushes current font context to a stack, modifies font context by declared \langle font modifiers \rangle, runs following \langle variant selector \rangle. It selects a font. Then pops the stack. The font context have its original values but new font is selected.

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

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

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

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

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

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

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

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

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

2.12.5 Modifying font features

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

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

\famvardef \fraclig {\setff{+frac}\currvar}

Compare 1/2 or 1/10 \fraclig to 1/2 or 1/10.

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

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

2.12.6 Special font modifiers

Despite the font modifiers declared in the font family file (and dependent on the font family), we have following font modifiers (independent of font family):
\setfontsize{\{sizespec\}} \% sets the font size
\setff{\{font feature\}} \% adds the font feature
\setfontcolor{\{color\}} \% sets font color
\setletterspace{\{number\}} \% sets letter spacing
\setwordspace{\{scaling\}} \% modifies word spacing

The \setfontsize command is described in the section 2.11. The \setff command was described in previous subsection.
\setfontcolor{\{color\}} specifies the color and the opacity of the text. The \{color\} parameter should be in hexadecimal format of four bytes \{red\}\{green\}\{blue\}\{opacity\}, for example FF0080FF means full red, zero green, half blue and full opacity. You can use names \texttt{red}, \texttt{green}, \texttt{blue}, \texttt{yellow}, \texttt{cyan}, \texttt{magenta}, \texttt{white}, \texttt{grey}, \texttt{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:

```latex
\famdecl [Heros] \Heros {TeX Gyre Heros fonts based on Helvetica}
\{\caps \cond \rm \it \bf \bi \{FiraMath\}
\{[texgyreheros-regular]\}}
\def\_fnamegen{[texgyreheros\_condV-\_currV]:\_capsV\_fontfeatures}
\wlog{\_detokenize{\{}
\caps ...... caps & small caps````J
\cond ...... condensed variants````J
\}}
\moddef \resetmod {\_fsetV caps={},cond={}} \_fvars regular bold italic bolditalic }
\moddef \caps {\_fsetV caps=smcp;onum; }
\moddef \nocsaps {\_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 \_famdecl macro must expand (at expand processor level only) to a file name of loaded font (or to its font name) and to optional font features appended. The Font Selection System uses this macro at primitive level in the following sense:

\_font \{\{selector\}\} {\{\_fontnamegen\}\} \{\_sizespec\}

Note that the extended \font syntax \{font\{selector\} {(font name):{font features}} (size spec.) or \font\{selector\} {\{(font file name):{font features}} (size spec.) is expected here.

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

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

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

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

The following auxiliary macros are used here:

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

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

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

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

is processed.

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

\_fsetV {word-a}={value-a},{word-b}={value-b} ...etc. terminated by a space

with obvious result \def \{word-a\}V {value-a}\ def \{word-b\}V {value-b} etc.
Example: if both font modifiers \caps and \cond were applied from the Heros family, then \def\capsV{+smcp;+onum} and \def\condV{cn} were processed by these font modifiers. If user needs the \bf variant at 11pt now then the
\font {texgyreheroscn-bold:+smcp;+onum;+tlig;} at11pt
is processed. We assume that a font file texgyreheroscn-bold.otf is present in your TeX system.

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

The \moddef macro does more things than simple \_def:
- The modifier macros are defined as \_protected.
- The modifier macros are defined as family-dependent.

The \famvardef macro has the same features.

The \langle Familyselector \rangle is defined by the \_famdecl macro as:
\protected\def\langle Familyselector \rangle {%
  \_def\_currfamily {\langle Familyselector \rangle} %
  \_def\_fontnamegen {⟨font name or font file name generated⟩} %
  \resetmod
}

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

The \_initfontfamily must be run after modifiers decaration. It sets \_let\_resetmod=\resetmod and runs the \langle Familyselector \rangle. Finally, it runs \_rm, so first font from new family is loaded and it is ready to use it.

Name conventions. Create font modifiers, new variants and the \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 \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 \_wordM. You can be sure that such name does not influence the private name space used by OpTEX.

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

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

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

2.12.8 How to write the font family file with optical sizes

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

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

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

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

We are using a special macro \_LMregfont in f-lmfonts.opm. It sets the file names to lowercase and enables to use a shortcasts 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 otical sizes are provided by a font family. Suppose that font file names are much more chaotic (because artists are very creative people), so you need to declare more systematic \{internal-templates\} and do an alias from each \{internal-template\} to \{real-font-name\}. For example, you can do it as follows:

\def\fontalias #1 #2 {\_regoptsizes #1 ?#2 {} <*>}
% alias name real font name
\fontalias crea-a-regular {Creative Font}
\fontalias crea-a-bold {Creative FontBold}
\fontalias crea-a-italic {Creative oblique}
\fontalias crea-a-bolditalic {Creative Bold plus italic}
\fontalias crea-b-regular {Creative Regular subfam}
\fontalias crea-b-bold {Creative subfam bold}
\fontalias crea-b-italic {Creative-subfam Oblique}
\fontalias crea-b-bolditalic {Creative Bold subfam Oblique}

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

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

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

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

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

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

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

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

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

### 2.12.10 Implementation of the Font Selection System

The \_famdecl \[⟨Family Name⟩\] \{⟨Famselector⟩\} \{⟨comment⟩\} \{⟨variants⟩\} \{⟨math⟩\} \{⟨font for testing⟩\} \{def\_fontnamegen{⟨data⟩}\} runs \_initunifonts, then checks if \{⟨Famselector⟩\} is defined. If it is true, then closes the file by \endinput. Else it defines \{⟨Famselector⟩\} and saves it to 64... etc.

The \_faminfo commad has the syntax:

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

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

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

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

The \_initunifonts initializes extended \font primitive (to be able to load Unicode fonts). Unfortunately, this part of OpTeX depends on Btex lua codes \texttt{ltxuatem.lua} and \texttt{luaotfload-main.lua}. And this code need to be declared a control sequence \texttt{\e@alloc\_attribute\_count} by \texttt{\countdef} primitive.

Moreover, the \_initunifont switches with the \_doresizefont macro to OTF mode which is represented by the macro \_doresizeunifont. This mode includes a fallback to TFM mode if \_fontnamegen is not defined. Finally, the \_initunifont sets itself to relax because we need not to do this work twice.

The \_famdecl \[⟨Family Name⟩\] \{⟨Famselector⟩\} \{⟨comment⟩\} \{⟨variants⟩\} \{⟨math⟩\} \{⟨font for testing⟩\} \{def\_fontnamegen{⟨data⟩}\} runs \_initunifonts, then checks if \{⟨Famselector⟩\} is defined. If it is true, then closes the file by \endinput. Else it defines \{⟨Famselector⟩\} and saves it to 64... etc.

The \_faminfo command has the syntax:

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

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

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

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

### 2.12.10 Implementation of the Font Selection System

The \_famdecl \[⟨Family Name⟩\] \{⟨Famselector⟩\} \{⟨comment⟩\} \{⟨variants⟩\} \{⟨math⟩\} \{⟨font for testing⟩\} \{def\_fontnamegen{⟨data⟩}\} runs \_initunifonts, then checks if \{⟨Famselector⟩\} is defined. If it is true, then closes the file by \endinput. Else it defines \{⟨Famselector⟩\} and saves it to 64... etc.

The \_faminfo command has the syntax:

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

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

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

The \_famtext writes a line to the terminal and to the log file when all families are listed.
the \_mainfamcommand macro because the \_initfontfamily needs it. The \_currenfamily is set to the \Famselector because the following \moddef commands need to be in the right font family context. The \currenfamily is set to the \Famselector by the \Famselector too, because \Famselector must set the right family context. The font family context is given by the current \currenfamily value and by the actual meaning of the \fontnamegen macro.

\def\famdecl [#1]#2#3#4#5#6#7#8{% 
\initunifonts \uniaccents 
\ifx #2\undefined 
\isfont{#7}\iffalse 
\protectededef{\if\noexpand\\famdecl\moddef\currenfamily{\csstring #2}\endinput}{\else 
\edef\currenfamily {\csstring #2}%= 
\def\mainfamcommand{#2} \def\mathfaminfo{#6}%= 
\protectededef#2{\def\noexpand\currenfamily{\csstring #2}\unexpanded{#8\resetmod}}%= 
\wterm {FONT: [#1] -- \string #2 \detokenize{(#3)\textquotesingle mods:{#4} vars:{#5} math:{#6}}}%= 
\fi 
\else \ea #2\ea \endinput \fi 
\else \ea \ea \ea \endinput \fi 
\def\initfontfamily{% 
\mainfamcommand \reloading \rm 
}\保护ed{\protect{\noexpand\famdecl\moddef\currenfamily{\csstring #2}\endinput}{\else 
\edef\currenfamily {\csstring #2}%= 
\def\mainfamcommand{#2} \def\mathfaminfo{#6}%= 
\protectededef#2{\def\noexpand\currenfamily{\csstring #2}\unexpanded{#8\resetmod}}%= 
\wterm {FONT: [#1] -- \string #2 \detokenize{(#3)\textquotesingle mods:{#4} vars:{#5} math:{#6}}}%= 
\fi 
\else \ea #2\ea \endinput \fi 
}\def\optname #1{\ifcsname \oz:#1\endcsname 
\ea \ea \ea \optnameA \csname \oz:#1\ea \endcsname 
\else \failedoptname{#1}\fi 
}\def\failedoptname #1{optname-fails:(#1)} 
\def\optnameA #1?#2 #3 <#4 {\ifx*#4\optnameC \else 
\ifdim \optsize<#4pt \optnameC \else \afterfifi \optnameA #1?#2 \fi \fi \fi 
}\def\optnameC #1* {\fi \fi} \def\afterfifi #1\fi\fi{\fi \fi #1} 
\def\optfontalias #1#2{\slet{\oz:#1}{\oz:#2}}

\fvars (rm-template) \bf-template \it-template \bi-template saves data for usage by the \currV macro. If a template is only dot then previous template is used (it can be used if the font family doesn’t dispose with all standard variants). \currV expands to a template declared by \fvars depending on the \variant name. Usable only of standard four variants. Next variants can be declared by the \famvardef macro. \protect \onlyif \key=\value\protect ,...\protect ,\key=\value\protect expands to \protect\def\\key\protect V{\value\protect} in the loop. 

\def\fvars #1 #2 #3 #4 {% 
\sdef{\fvar:rm}{#1}\% 
\sdef{\fvar:bf}{#2}\% 
\sdef{\fvar:it}{#3}\% 
\sdef{\fvar:bi}{#4}\% 
\fset{\fvar:bf}{\fvar:rm}\% 
\fset{\fvar:bi}{\fvar:it}\% 
\fset{\fvar:it}{\fvar:bf}\% 
\fset{\fvar:rm}{\fvar:it}\% 
} 
\def\currV{\cs{\fvar: whatresize}} 
\def\V{ } 
\def \fsetV #1 \{ \fsetVa #1=,\} 
\def \fsetVa #1=#2,\{\isempty{#1}\iffalse 
\ifx #1\else \sdef{\oz:#1}{\fvar:rm}\ea \ea \fsetVa\ea\fsetVa\fi\fi 
} 
\def \onlyif #1=#2:#3{\}
The \texttt{\moddef \{\texttt{modifier}\} \{\texttt{data}\}} simply speaking does \texttt{\def \{\texttt{modifier}\} \{\texttt{data}\}}, but we need to respect the family context. In fact, \texttt{\protected \def \f:\{\texttt{current family}\}:\{\texttt{modifier}\} \{\texttt{data}\}} is performed and the \texttt{\{\texttt{modifier}\}} is defined as \texttt{\_famdepend \{\texttt{modifier}\} \_f:\_currfamily \{\texttt{modifier}\}}. It expands to \texttt{\_f: \_currfamily \{\texttt{modifier}\}} value if it is defined or it prints warning. When the \texttt{\_currfamily} value is changed then we can declare the same \texttt{\{\texttt{modifier}\}} with different meaning.

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

\begin{verbatim}
\_edef\_act\{\noexpand\_isinlist\{,#2\}\{\cs\{#1\V\}\}\noop\_act
\_iftrue #3\_fi
\}
\_edef\_act\{\noexpand\_isinlist\{,#2\}\{\cs\{#1\V\}\}\noop\_act
\_iftrue #3\_fi
\end{verbatim}

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

- \texttt{\protected \def \f: \_currfamily \{XX\}},
- \texttt{\def \f: \{current family\} \{XX\}} keeps family dependent definition,
- \texttt{\_tryload\{XX\} \{\texttt{data}\}} loads actually the font \texttt{\_ten\{XX\}},
- \texttt{\_currvar: \_ten\{XX\} \{\texttt{data}\}} in order to the \texttt{\currvar} macro work correctly.

\begin{verbatim}
\_edef\_act\{\noexpand\_isinlist\{,#2\}\{\cs\{#1\V\}\}\noop\_act
\_iftrue #3\_fi
\}
\_edef\_act\{\noexpand\_isinlist\{,#2\}\{\cs\{#1\V\}\}\noop\_act
\_iftrue #3\_fi
\end{verbatim}
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.

```latex
\begin{Verbatim}
\def\_famf:\langle\textit{familyname}\rangle\{\langle\textit{file-name}\rangle\}
\end{Verbatim}
```

When the `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.

 When the `fams-ini.opm` or `fams-loca.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.

```latex
\begin{Verbatim}
\def\_faminfo\{\langle\textit{family-name}\rangle\{\langle\textit{mod-plus-vars}\rangle\}\{\langle\textit{file-name}\rangle\}
\end{Verbatim}
```

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

```latex
\begin{Verbatim}
\newtoks\_catalogsample
\end{Verbatim}
```
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 \_setff #1{\ifx^#1^\let \_fontfeatures=\_defaultfontfeatures\else \edef \_fontfeatures{\_fontfeatures #1;}\fi \reloading}
\setff {} % default font features: +tlig;
\def \_removefeature #1{\if\isinlist \_fontfeatures{#1}\true\def \_tmp ##1#1##2##3\relax{\def \_fontfeatures{##1##3}}\else\edef \_fontfeatures{\_fontfeatures letterspace=##1;}\fi}
\public \setff ;
```

The \setfontcolor and \setletterspace are macros based on the special font features provided by Lua\TeX{} (and by X\TeX{} 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 \texttt{+Ws} too in order the font is reloded by the \texttt{font} primitive (with independent \fontdimen registers).

```
\def \_savedfontcolor{} \def \_savedletterspace{} \def \_savedwsp{} \def \_setfontcolor #1{\_removefeature{color=}\edef \_tmp{\_calculatefontcolor{#1}}\ifx \_tmp \empty \else \edef \_fontfeatures{color=\_tmp;}\fi \reloading}
\def \_setletterspace #1{\_removefeature{letterspace=}\if^#1^\else \edef \_fontfeatures{letterspace=#1;}\fi \reloading}
\def \_setwordspace #1{\if^#1^\def \_setwsp##1{}\_removefeature{+Ws}\else \def \_setwsp\{\_setwspA{#1}\_setff{+Ws}\fi \reloading}

\def \_setwsp #1{} \def \_setwspA #1#2{\fontdimen2#2=#1 \fontdimen2#2 \fontdimen3#2=#1 \fontdimen4#2=#1}

\def \_calculatefontcolor#1{\trycs\{_fc:#1\}{#1}} % you can define more smart macro ...
\sdef\{_fc:red\}{FF0000FF} \sdef\{_fc:green\}{00FF00FF} \sdef\{_fc:blue\}{0000FFFF}
\sdef\{_fc:yellow\}{FFFF00FF} \sdef\{_fc:cyan\}{00FFFFFF} \sdef\{_fc:magenta\}{FF00FFFF}
\sdef\{_fc:white\}{FFFFFFFF} \sdef\{_fc:grey\}{00000080} \sdef\{_fc:lgrey\}{00000025}
\sdef\{_fc:black\}{} % ... you can declare more colors...
\public \_setfontcolor \_setletterspace \_setwordspace ;
```

2.13 Preloaded fonts for math mode

The Computer Modern and AMS fonts are preloaded here in classical math-fam concept, where each math family includes three fonts with max 256 characters (typically 128 characters).
On the other hand, when \fontfam macro is used in the document then text font family and appropriate math family is loaded with Unicoded fonts, i.e. Unicoded-math is used. It re-defines all settings given here.

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

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 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.

\protected\def \_bbchar {\_fam5 } % double stroked letters
\protected\def \_frak {\_fam7 } % fraktur
\protected\def \_script {\_fam6 } % more extensive script than \cal

\public \rm \bf \it \bi \tt \mit \cal \bbchar \frak \script ;

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

\regtfm cmmi 0 cmmi5 5.5 cmmi6 6.5 cmmi7 7.5 cmmi8 8.5 cmmi9 9.5
cmm10 11.1 cmmi12 *
cmmib 0 cmmib5 5.5 cmmib6 6.5 cmmib7 7.5 cmmib8 8.5 cmmib9 9.5 cmmib10 *
cmbsy 0 cmbsy5 5.5 cmbsy6 6.5 cmbsy7 7.5 cmbsy8 8.5 cmbsy9 9.5 cmbsy10 *
cmex 0 cmex7 7.5 cmex8 8.5 cmex9 9.5 cmex10 *
cmexb 0 cmexb10 *
cmr 0 cmr5 5.5 cmr6 6.5 cmr7 7.5 cmr8 8.5 cmr9 9.5
cmr10 11.1 cmr12 15 cmr17 *
cmbx 0 cmbx5 5.5 cmbx6 6.5 cmbx7 7.5 cmbx8 8.5 cmbx9 9.5
cmbx10 11.1 cmbx12 *
cmti 0 cmti7 7.5 cmti8 8.5 cmti9 9.5 cmti10 11.1 cmti12 *
cmtt 0 cmtt10 11.1 cmtt12 *

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

\regtfm eufm 0 eufm5 5.5 eufm6 6.5 eufm7 7.5 eufm8 8.5 eufm9 9.5 eufm10 *
eufb 0 eufb5 5.5 eufb6 6.5 eufb7 7.5 eufb8 8.5 eufb9 9.5 eufb10 *
rsfs 0 rsfs5 6 rsfs7 8.5 rsfs10 *

\loadmathfamily ⟨number⟩ ⟨font⟩ loads one math family, i.e. the triple of fonts in the text size, script size and script-script size. The ⟨font⟩ is ⟨font-id⟩ used in the \regtfm parameter or the real TFM name. The family is saved as \fam⟨number⟩.

\setmathfamily ⟨number⟩ ⟨font-switch⟩ loads one math family like \loadmathfamily does it. But the second parameter is a ⟨font-switch⟩ declared previously by the \font primitive. The font family is loaded at \sizemtext, \sizemscript and \sizemsscript sizes. These sizes are set by \setmathsizes [⟨text-size⟩/⟨script-size⟩/⟨scriptscript-size⟩] macro. These parameters are given in the \ptmunit unit, it is set to 1 \ptunit and it is set to 1 pt by default.

\corrmsizes should be used in the \normalmath and \boldmath macros if you need a size correction when a selected math family is loaded. It is similar as ex-height correction but for math fonts.

\def\corrmsizes{⟨\ptmunit=1\ptunit⟩\relax} % for corrections of sizes in different fonts
\def\loadmathfamily #1 #2 {⟨\chardef \tmp#1\corrmsizes\edef\optsizesave{⟨\the\optsizes⟩}\def\optsize\sizemtext{⟨\font\mF=⟨\whichtfm(#2)⟩ at \optsize \textfont#1\optsizesave}⟨\the\optsize⟩\def\optsize\sizemscript{⟨\font\mF=⟨\whichtfm(#2)⟩ at \optsize \scriptfont#1\optsizesave}⟨\the\optsize⟩\def\optsize\sizemsscript{⟨\font\mF=⟨\whichtfm(#2)⟩ at \optsize \scriptscriptfont#1\optsizesave}⟨\the\optsize⟩\relax}{
The \_setmathdimens macro is used in \texttt{\textbackslash normalmath or \textbackslash boldmath} macros. It makes math dimensions dependent on the font size (plain TeX sets them only for 10 pt typesetting). The \texttt{\textbackslash 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 \texttt{\textbackslash 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 mathcode of it is \textquote{8000}. It means that it is active character in math mode. It is defined as subscript prefix.

There is a problem: The \texttt{x\_n} is tokenized as \texttt{x}, _, n and it works without problem. But \texttt{\textbackslash int\_a\_b} is tokenized as \texttt{\textbackslash int\_a}, _, b. The control sequence \texttt{\textbackslash int\_a} isn't defined. We must write \texttt{\textbackslash int\_a\_b}.

The lua code presented here solves this problem. But you cannot set our own control sequence in the form \texttt{\textbackslash \langle word \rangle\_\langle one-letter \rangle} (where \texttt{\langle word \rangle} is sequence of letters) because such control sequences are unaccessible: proprocessor rewrites it.

The \texttt{\textbackslash mathsb\_on} macro activates the rewriting rule \texttt{\langle word \rangle\_\langle nonletter \rangle} to \texttt{\langle word \rangle\_\langle nonletter \rangle} and \texttt{\langle word \rangle\_\langle letter \rangle\langle nonletter \rangle} to \texttt{\langle word \rangle\_\langle letter \rangle\langle nonletter \rangle} at input processor level. The \texttt{\textbackslash mathsb\_off} deactivates it. You can ask by \texttt{\_ifmathsb} if this feature is activated or deactivated. By default, is is activated in the \texttt{\everyjob}. 
function (str)
    return string.gsub(str.. " ", " (\_nbb[a-zA-Z]+)_(\[a-zA-Z]*[^_a-zA-Z])", " _pcent 1 _pcent 2")
end
\global \mathsbtrue
\def \mathsoff {
    \directlua{ callback.register("process_input_buffer", nil) }
    \global \mathsbfalse
}
\public \mathsoff \mathsbon ;

All mathcodes are set to equal values as in plain\TeX. But all encoding-dependend declarations (like these) will be set to different values when Unicode-math font is used.

\mathcode\]^@="2201 % \cdot
\mathcode\]^A="3223 % \downarrow
\mathcode\]^B="010B % \alpha
\mathcode\]^C="010C % \beta
\mathcode\]^D="225E % \land
\mathcode\]^E="023A % \lnot
\mathcode\]^F="3232 % \in
\mathcode\]^G="0119 % \pi
\mathcode\]^H="0115 % \lambda
\mathcode\]^I="010D % \gamma
\mathcode\]^J="010E % \delta
\mathcode\]^K="3222 % \uparrow
\mathcode\]^L="2206 % \pm
\mathcode\]^M="2208 % \oplus
\mathcode\]^N="0231 % \infty
\mathcode\]^O="0140 % \partial
\mathcode\]^P="321A % \subset
\mathcode\]^Q="321B % \supset
\mathcode\]^R="225C % \cap
\mathcode\]^S="225B % \cup
\mathcode\]^T="0238 % \forall
\mathcode\]^U="0239 % \exists
\mathcode\]^V="220A % \otimes
\mathcode\]^W="3224 % \leftrightarrow
\mathcode\]^X="3220 % \leftrightarrow
\mathcode\]^Y="3221 % \rightarrow
\mathcode\]^Z="8000 % \ne
\mathcode\]^\="2205 % \diamond
\mathcode\]^\="3214 % \le
\mathcode\]^\="3215 % \ge
\mathcode\]^\="3211 % \equiv
\mathcode\]^\="225F % \lor
\mathcode\]^\="8000 % \space
\mathcode\]^\="5021
\mathcode\]^\="8000 % \prime
\mathcode\]^\="4028
\mathcode\]^\="5029
\mathcode\]^\="2203 % \not
\mathcode\]^\="202B
\mathcode\]^\="613B
\mathcode\]^\="2200
\mathcode\]^\="013A
\mathcode\]^\="013D
\mathcode\]^\="303A
\mathcode\]^\="603B
\mathcode\]^\="313C
\mathcode\]^\="303D
\mathcode\]^\="313E
\mathcode\]^\="503F
\mathcode\]^\="405B
\mathcode\]^\="026E % \backslash
\mathcode\]^\="505D
\mathcode\]^\="8000 % \math-active subscript
\mathcode\]^\="4266
\mathcode\]^\="026A
All control sequences declared by \texttt{\mathchardef} are supposed (by default) only for public usage. It means that they are declared without \_ prefix. If such sequences are used in internal \texttt{OpTeX} macro then their internal prefixed form is declared using \texttt{\_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\texttt{TeX}, but they are \texttt{\protected} in \texttt{OpTeX}.
These macros are defined similarly as in plain\TeX. Only internal macro names from plain\TeX with @ character are re-written in more readable form. \sp is alternative for ^ . The \sb alternative for _ was defined at the line 27 of the file math-macros.opm.

Active \texttt{prime} character is defined here.

Math relations defined by the \texttt{jointrel} plain \TeX macro:
\textbf{Math accents (encoding dependent declarations).} \end{math-macros}
\protected\def\bmod{\nonscript\_\medmuskip\mkern5mu\mathbin{\mathrm{mod}}\penalty900\mkern5mu\nonscript\_\medmuskip}
\protected\def\pmod#1{\allowbreak\mkern18mu({\mathrm{mod}\thinspace#1})}
\public\bmod\pmod;
\cases, \matrix, \pmatrix and \bordermatrix macros from plain \TeX
\openup, \eqalign, \displaylines and \eqalignno macros from plain \TeX.
\def\amsafam{4}\def\amsbfam{5} \mathchardef\boxdot\textquotesingle\amsafam\text{00}\mathchardef\boxplus\textquotesingle\amsafam\text{01}\mathchardef\boxtimes\textquotesingle\amsafam\text{02}\mathchardef\square\textquotesingle\amsafam\text{03}\mathchardef\blacksquare\textquotesingle\amsafam\text{04}

These macros are inspired from ams-math.tex file.
The \not macro is re-defined to be more intelligent 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;
only otherwise, \not\XXX is done in the usual way.

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

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

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

The \\mathbox{⟨text⟩} macro is copied from OPmac trick 078. It behaves like \hbox{⟨text⟩} but the ⟨text⟩ is scaled to smaller size if it is used in scriptstye or scriptscript style.
2.15 Unicode-math fonts

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

\loadboldmath \textit{bold-font} \to \textit{normal-font} loads bold variant only if \textit{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 \url{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\{[texgypagella-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

```latex
\loadmath \{ (Unicode Math fonts <2020-02-25>) \} % preloaded in format
```

\loadmath \{ (Unicode Math fonts) \} loads given font. It does:

- define \unimathfont as \textit{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.

\normalunimath
\normalunimath{\loadumathfamily 1 {\unimathfont}{} % Base font
\loadmathfamily 4 rsfs % script
\setunimathdimens
}%
\boldunimath
\ifx\unimathboldfont\undefined
\loadumathfamily 1 {\unimathfont}{embolden=1.7;} % Base faked bold
\else
\loadumathfamily 1 {\unimathboldfont}{} % Base real bold font
\fi
\loadmathfamily 4 rsfs % script
\setunimathdimens
%
\setunimathdimens{X PlainTeX sets these dimens for 10pt size only:
\delimitershortfall=0.5\fontdimen6\textfont3
\nulldelimiterspace=0.12\fontdimen6\textfont3
\scriptspace=0.05\fontdimen6\textfont3
}%

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

\umathrange{\from\to}{\first} sets \texttt{\Umathcodes} of the characters in the interval \texttt{\from}--\texttt{\to} to \texttt{\first}, \texttt{\first+1}, \texttt{\first+2} etc., but \texttt{\Umathcharholes} are skipped (\texttt{\umathcharholes} are parts of
the Unicode table not designed for math alphabets but they causes that the math alphabets are not continuously spread out in the table; I mean that the designers were under the influence of drugs when they created this part of the Unicode table. The ⟨from⟩–⟨to⟩ clause includes normal letters like A–Z. ⟨from⟩–⟨to⟩ is the same as ⟨first⟩ ⟨first⟩. ⟨from⟩–⟨to⟩ clause includes normal letters like A–Z.

\_umahrangegreek \langle first \rangle is the same as \_umathrange \{⟨alpha⟩–⟨omega⟩\} \langle first \rangle. \_umahrangegreek \langle first \rangle is the same as \_umathrange \{⟨Alpha⟩–⟨Omega⟩\} \langle first \rangle.

\_umathnumB \_umathnumB+1 \_umathnumB+2 etc. It is used for redefining the contol sequences for math Greek \alpha, \beta, \gamma etc.

2.15.2 Macros and codes set when \loadmatfont is processed

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

The control sequences for \texttt{\alpha}, \texttt{\beta} etc are redefined here. The \texttt{\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.

\texttt{unimath-codes.opm}

The math alphabets are declared here using the \_umathrange macro.
The \texttt{\cal}, \texttt{\bbchar}, \texttt{\frak}, \texttt{\script} and the \texttt{\rm}, \texttt{\bf}, \texttt{\it}, \texttt{\bi}, \texttt{\tt} are defined here. Their “8bit definitions” from the file \texttt{math-preload.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\bfdigits}\}
\public \bf \bi ;
\inmath \{\cmds\} applies \cmds only in math mode.

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:

We have to read this information and convert it to the \Umathcodes.
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{''00397}{\mupEta }{\mathalpha}{capital eta, greek}%
\UnicodeMathSymbol{''00398}{\mupTheta }{\mathalpha}{capital theta, greek}%
\UnicodeMathSymbol{''00399}{\mupIota }{\mathalpha}{capital iota, greek}%
\UnicodeMathSymbol{''0039A}{\mupKappa }{\mathalpha}{capital kappa, greek}%
\UnicodeMathSymbol{''0039B}{\mupLambda }{\mathalpha}{capital lambda, greek}%
\UnicodeMathSymbol{''0039C}{\mupMu }{\mathalpha}{capital mu, greek}%
\UnicodeMathSymbol{''0039D}{\mupNu }{\mathalpha}{capital nu, greek}%
\UnicodeMathSymbol{''0039E}{\mupXi }{\mathalpha}{capital xi, greek}%
\UnicodeMathSymbol{''0039F}{\mupOmicron }{\mathalpha}{capital omicron, greek}%
\UnicodeMathSymbol{''003A0}{\mupPi }{\mathalpha}{capital pi, greek}%
\UnicodeMathSymbol{''003A1}{\mupRho }{\mathalpha}{capital rho, greek}%
\UnicodeMathSymbol{''003A3}{\mupSigma }{\mathalpha}{capital sigma, greek}%
\UnicodeMathSymbol{''003A4}{\mupTau }{\mathalpha}{capital tau, greek}%
\UnicodeMathSymbol{''003A6}{\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...

```
\protected\def \sqrt {\\Radical 1 "0221A}
\protected\def \cuberoot {\\Radical 1 "0221B}
\protected\def \fourthroot {\\Radical 1 "0221C}
```

```
\public \sqrt \cuberoot \fourthroot;
```

```
\def\intwithnolimits#1#2 {\\ifx#1\relax \else
\ea\let\csnamestring#1op\endcsname=#1\fi
\ifx#3\mathaccent \gdef#2{\Umathaccent fixed 7 1 #1}\fi}
\intwithnolimits \int \iint \iiint \oint \oiint \oiiint \intclockwise \varointclockwise \ointctrclockwise
\sumint \iiiint \intbar \intBar \fint \pointint \sqint \intlarhk \intx \intcap \intcup \upint \lowint
```

```
\protected\def \vert {\Udelimiter 0 1 "07C}
\protected\def \Vert {\Udelimiter 0 1 "02016}
\protected\def \Vvert {\Udelimiter 0 1 "02980}
```

```
\protected\def \\Udelimiter 0 1 "02980 \paunch 0 1 "07C}
\protected\def \\Udelimiter 0 1 "02016}
\protected\def \\Udelimiter 0 1 "02980}
```
Aliases are declared here. They are names not mentioned in the `unimath-table.opm` file but commonly used in TeX.
The \not macro is redefined here.

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.

2.15.3 A few observations

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

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

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

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

2.16 Scaling fonts in document (high-level macros)

These macros are documented in section 1.3.2 from user point of view.
\texttt{\textbackslash typosize \{[\texttt{font-size]/\texttt{baselineskip}]\}} sets given parameters. It sets text font size by the \texttt{\textbackslash setfontsize} macro and math font sizes by setting internal macros \texttt{\_sizemtext}, \texttt{\_sizemscript} and \texttt{\_sizemsscript}. It uses common concept font sizes: 100\%, 70\% and 50\%. The \texttt{\_setmainvalues} sets the parameters as main values when the \texttt{\_typosize} is called first.

\begin{verbatim}
\protecteddef \_typosize [#1/#2]{%
  \textfontsize{#1}\_mathfontsize{#1}\_setbaselineskip{#2}%
  \setmainvalues \ignorespaces
}
\def \_textfontsize #1{
  \if$#1$\else \setfontsize{at#1\_ptunit}\fi}
\def \_mathfontsize #1{
  \if$#1$\else
    \tmpdim=#1\_ptunit
    \edef \_sizemtext{\ea\_ignorept \the\tmpdim \ptmunit}\
    \tmpdim=0.7\tmpdim
    \edef \_sizemscript{\ea\_ignorept \the\tmpdim \ptmunit}\
    \tmpdim=#1\_ptunit \tmpdim=0.5\tmpdim
    \edef \_sizemsscript{\ea\_ignorept \the\tmpdim \ptmunit}\
  \fi}
\public \_typosize ;
\end{verbatim}

\texttt{\textbackslash typoscale \{[\texttt{font-factor]/\texttt{baseline-factor}]\}} scales font size and baselineskip by given factors in respect to current values. It calculates the \texttt{\_typosize} parameters and runs the \texttt{\_typosize}.

\begin{verbatim}
\protecteddef \_typoscale [#1/#2]{%
  \ifx$#1$\def \_tmp{[/}\else
    \settmpdim{#1}\optsize
    \edef \_tmp{[\_ea\_ignorept \the\tmpdim]/}\fi
  \ifx$#2$\edef \_tmp{\_tmp]\}\else
    \settmpdim{#2}\baselineskip
    \edef \_tmp{\_tmp \_ea\_ignorept \the\tmpdim]\}\fi
  \ea\_typosize\_tmp
}
\def \_settmpdim#1#2{%
  \tmpdim=#1pt \divide\tmpdim by1000
  \tmpdim=\ea\_ignorept \the#2\tmpdim
}\public \_typoscale ;
\end{verbatim}

\texttt{\_setbaselineskip \{\texttt{baselineskip}\}} sets new \texttt{\_mainbaselineskip} and more values of registers which are dependent on the \texttt{\_baselineskip} including the \texttt{\_strutbox}.

\begin{verbatim}
\def \_setbaselineskip #1{\if$#1$\else
  \tmpdim=#1\_ptunit
  \_mainbaselineskip\_tmpdim \_relax
  \_bigskipamount\_tmpdim plus.33333\_tmpdim minus.33333\_tmpdim
  \_medskipamount=.5\tmpdim plus.16666\tmpdim minus.16666\_tmpdim
  \_smallskipamount=.25\tmpdim plus.08333\tmpdim minus.08333\_tmpdim
  \_normalbaselineskip\_tmpdim
  \_jot=.25\_tmpdim
  \_maxdepth=.33333\_tmpdim
  \setbox\_strutbox=\hbox{\_vrule height.709\_tmpdim depth.291\_tmpdim width0pt}\%
  \fi}
\end{verbatim}

\texttt{\_setmainvalues} sets the current font size and \texttt{\_baselineskip} values to the \texttt{\_mainfosize} and \texttt{\_mainbaselineskip} registers. It redefines itself in order to set the main values only first. \texttt{\_scalemain} returns to these values if they were set. Else they are set to 10/12 pt.
\normalmath % load fonts if \typosize is running first
\let \setmainvalues = \setmainvaluesL
\def \setmainvaluesL {
  \ifmmode \normalmath \else
  \rm \everymath={\normalmath} \everydisplay={\normalmath} \fi}
\def \scalemain {%
  \ifdim \mainfosize=0pt
    \mainfosize=10pt \mainbaselineskip=12pt
  \let \setmainvalues = \setmainvaluesL
  \fi
  \optsize=\mainfosize \baselineskip=\mainbaselineskip
}
\public \scalemain \mainfosize \mainbaselineskip ;

\thefontsize [(size)] and \thefontscale [(factor)] do modification of the size of the current font. They are implemented by the \newcurrfontsize macro.

\em keeps the weight of the current variant and switches roman ↔ italic. It adds the italic correction by the \additcorr and \afteritcorr macros. The second does not add italic correction if the next character is dot or comma.

The \boldify macro does \let \it \bi and \let \normalmath = \boldmath.

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

The output routine \_optexoutput is similar as in plain \TeX{}. It does:

- \_begoutput which does:
  - increments \gpageno,
  - prints \_Xpage{{\pageno}} to the .ref file (if \openref is active),
  - calculates \hoffset,
  - sets local meaning of macros used in headlines/footlines (see \regmacro).
- \_shipout\_completepage, which is \vbox of –
  - background box, if \pgbackground is non-empty,
  - headline box by \_makeheadline, if the \headline is nonempty,
  - \vbox to\vsize of \_pagecontents which consists of –
    - \pagedest, the page destination pg:{\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 nonempty (from \fnote, \footnote),
    - \pgbottomskip (default is 0pt).
  - footline box by \_makefootline, if the \footline is nonempty
- \_endoutput which does:
  - increments \pageno using \advancepageno
  - runs output routine repeatedly if \dosupereject is activated.

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

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

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

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

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

The \_ensureblack sets the typesetting of its parameter locally to \Black color. We needn’t do this if colors are never used in the document. So, default value of the \_ensureblack macro is empty. But first usage of color macros in the document re-defines \_ensureblack. See the section 2.19 for more details.
\_makeheadline creates \vbox to0pt with its contents (the \headline) shifted by \headlinedist up.

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

The \_pagecontents is similar as in plain \TeX. The only difference is that the \pagedest is inserted at the top of \_pagecontents and \_ensureblack is applied to the \topins and \footins material. The \_footnoterule is defined here.

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

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.
2.18 Margins

The \margins macro is documented in the section 1.2.1.

\margins is a macro that sets the page origin at the top left corner of the paper, no at the obscure position 1in, 1in. It is much more comfortable for macro writers.
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.

\_def\trueunit{}\_def\magscale[#1]{\_mag=#1\_def\trueunit{true}%=\_mag=⟨factor⟩ and recalculates page dimensions to their true values.

By default, OpTeX sets \localcolor. It means that the typesetting returns back to a previous color at the end of current group, so you cannot write \Black explicitly. This is implemented using
The implementation of colors is based on colorstack, so the current color can follow across more pages. It is not so obvious because PDF viewer (or PDF interpreter) manipulates with colors locally at each PDF page and it initializes each PDF page with black on white color.

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

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

You can move a shared amount of CMY components (i.e. their minimum) to the $\mathit{K}$ 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:

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

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

\colordef \Color {\Brown-\Black}

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

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

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

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

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

CMYK to RGB:

\[
\]

RGB to CMYK:

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

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

\colordef{Colors <2020-03-18>} % loaded in format

We declare internal boolean value \_iflocalcolor ad do \localcolor as default.
The basic colors in CMYK \Blue \Red \Brown \Green \Yellow \Cyan \Magenta \Grey \LightGrey \White and \Black are declared here.

By default, the _setcmykcolor \setrgbcolor and _setgreycolor macros with \{\pdf-primitive\} parameter expand to _setcolor{⟨pdf-primitive⟩} using _formatcmyk or _formatrgb or _formatgrey expandable macros. For example _setcmykcolor{1 0 0 0} expands to _setcolor{1 0 0 rg 1 0 0 RG}. We set both types of colors (for lines (K or RG or G) and for fills (r or rg or g) together in the ⟨pdf-primitive⟩ command. This is the reason why the \fillstroke uses both its parameters. If only fills are needed you can do \def\fillstroke#1#2{#1}. If only strokes are needed you can do \def\fillstroke#1#2{#2}.

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 headres 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 _setcmykcolor\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 everyjob because the initialization is not saved to the format.

\%\_mathchardef\_fnotestack=\pdfcolorstackinit page {0 g 0 G} \% must be in everyjob

\_def \_openfnotestackA {\_pdfcolorstack\_fnotestack current}

We use lua codes for RGB to CMYK or CMYK to RGB conversions and for addition color components in the \colordef macro. The \_rgbtocmyk \langle R \rangle \langle G \rangle \langle B \rangle ; expands to \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle and the \_cmyktorgb \langle C \rangle \langle M \rangle \langle Y \rangle \langle K \rangle ; expands to \langle R \rangle \langle G \rangle \langle B \rangle. The \_colorcrop, \_colordefFin and \_douseK are auxiliary macros used in the \colordef. The \_colorcrop rescales color components in order to they are in \[0, 1\] interval. The \_colordefFin expands to the values accumulated in Lua code \_color_C, \_color_M, \_color_Y and \_color_K. The \_douseK applies \_useK to CMYK components.

\def\_rgbtocmyk #1 #2 #3 ;{%\_ea \_stripzeros \_detokenize \_ea{\_directlua{\local kr = \math.max(#1,#2,#3)
if (kr==0) then
\tex.print(’0. 0. 0. 1 ;’)
else
\tex.print(string.format(’\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;’, (kr-#1)/kr, (kr-#2)/kr, (kr-#3)/kr, 1-kr))
\end}
\}}

\def\_cmyktorgb #1 #2 #3 #4 ;{%\_ea \_stripzeros \_detokenize \_ea{\_directlua{\local kr = 1-#4
\tex.print(string.format(’\_pcent.3f \_pcent.3f \_pcent.3f ;’, (1-#1)*kr, (1-#2)*kr, (1-#3)*kr))
\}}}

\def\_colorcrop{\_directlua{\local m=\math.max(\_color_C, \_color_M, \_color_Y, \_color_K)
if (m>1) then
\_color_C=\_color_C/m \_color_M=\_color_M/m \_color_Y=\_color_Y/m \_color_K=\_color_K/m
\end}

\def\_colordefFin{\_colorcrop \_ea \_stripzeros \_detokenize \_ea{\_directlua{\tex.print(string.format(’\_pcent.3f \_pcent.3f \_pcent.3f \_pcent.3f ;’, \_color_C, \_color_M, \_color_Y, \_color_K))
\}}}

\def\_douseK{\_colorcrop \_directlua{\local kr=\math.min(\_color_C, \_color_M, \_color_Y)
if (kr==1) then
\_color_C=0 \_color_M=0 \_color_Y=0 \_color_K=1
else
\_color_C=(\_color_C-kr)/(1-kr) \_color_M=(\_color_M-kr)/(1-kr) \_color_Y=(\_color_Y-kr)/(1-kr) \_color_K=\math.min(\_color_K+kr,1)
\end}

\}}

We have a problem with the %.3f directive in Lua code. It prints trailed zeros: (0.300 instead desired 0.3) but we want to save PDF file space. The macro \_stripzeros removes these trailing zeros at expand processor level. So \_stripzeros 0.300 0.400 0.560 ; expands to 0.3 0.4 0.56.

\def\_stripzeros #1.#2 #3{\_ifx0#1\_else#1\_fi.\_stripzeroA #2 0 ;%\_ifx;#3\_else \_space \_ea\_stripzeros\_ea#3\_fi}

\def\_stripzeroA #10 #2{\_ifx^#2^\_stripzeroC#1:\_else #1\_fi}

\def\_stripzeroB #10 #2{\_ifx^#2^\_stripzeroC#1:\_else \_stripzeroB#1\_fi}

\def\_stripzeroC #1 #2{#1}

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

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

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

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

\begin{verbatim}
217 \_def \_rgbcolordef {\_commoncolordef \_setrgbcolor \krg}
218 \_def \_cmykcolordef {\_commoncolordef \_setcmykcolor \KRG}
219 \_def \_commoncolordef#1#2#3#4#5#6{\%
220 \_begingroup
221 \_directlua{color_C=0 color_M=0 color_Y=0 color_K=0}
222 \_def \_setcmykcolor##1{!=#2 ##1 }
223 \_def \_setrgbcolor ##1{!=#3 ##1 }
224 \_def \_setgreycolor##1{!=#4 ##1 }
225 \_let \_useK=_\relax
226 \_edef \_tmpb{+#6}
227 \_replstring \_tmpb{+ }{+} \_replstring \_tmpb{- }{-} \_replstring \_tmpb{+}{\_addcolor} \_replstring \_tmpb{-}{\_addcolor-}
228 \_ifx K#2 \_let \_useK=\_douseK \_fi
229 \_tmpb
230 \_edef \_tmpb{\_colordefFin}
231 \_ifx k#2 \_edef \_tmpb{\_ea\_stripK \_tmpb;}
232 \_fi
233 \_ea\_endgroup
234 \_ea\_def \_ea#5\_ea{\_ea#1\_ea{\_tmpb}}
235 \_es\_endgroup
236 \_es\_def \_es#5\_es{\_es#1\_es{\_es#6\_es}(\_es#7\_es)}
237 \_es\_def \_addcolor#1\#2\#3\#4\#5 {\cs{addcolor:A #1 #2 #3 #4 #5}}
238 \_es\_def \_addcolorA #1 #2 #3 #4 #5 {%
239 \_def \_tmpa{#1}\_ifx \_tmpa\_empty \_else \_edef \_tmpa{\_tmpa*}\_fi
240 \_directlua{color_C=math.max(color_C+\_tmpa#2,0)
241 color_M=math.max(color_M+\_tmpa#3,0)
242 color_Y=math.max(color_Y+\_tmpa#4,0)
243 color_K=math.max(color_K+\_tmpa#5,0)}
244 \_sdef{addcolor:=K}#1 #2 #3 #4 #5 {\_addcolorA #1 #2 #3 #4 #5 }
245 \_sdef{addcolor:^K}#1 #2 #3 #4 #5 {\_addcolorA #1 (1-#2) (1-#3) (1-#4) #5 }
246 \_sdef{addcolor:^G}#1 #2 {\_addcolorA #1 (1-#2) (1-#2) (1-#2) #2 }
247 \_sdef{addcolor:=G}#1 #2 {\_addcolorA #1 #2 (1-#2) (1-#2) #2 }
248 \_sdef{addcolor:=R}#1 #2 #3 #4 {\_edef \_tmpa{\noexpand\_addcolorA #1 \_rgbtocmyk #2 #3 #4 ; }\_tmpa}
249 \_sdef{addcolor:^R}#1 #2 #3 #4 {\cs{addcolor:=R}#1 (1-#2) (1-#3) (1-#4) #4 }
250 \_sdef{addcolor:=k}#1 #2 #3 #4 #5 {\cs{addcolor:A #1 \_cmytorgb #2 #3 #4 #5 ; }\_tmpa}
251 \_sdef{addcolor:^k}#1 #2 #3 #4 #5 {\cs{addcolor:A #1 #2 #3 #4 #5 ; }\_tmpa}
252 \_sdef{addcolor:^g}#1 #2 {\cs{addcolor:A #1 #2 #2 #2 #2 ; }\_tmpa}
253 \_sdef{addcolor:=g}#1 #2 {\cs{addcolor:A #1 #2 #2 #2 #2 ; }\_tmpa}
254 \_sdef{addcolor:^r}#1 #2 #3 #4 {\_addcolorA #1 #2 #3 #4 #0 }
255 \_sdef{addcolor:=r}#1 #2 #3 #4 {\_addcolorA #1 #2 #3 #4 #0 }
\end{verbatim}
Public versions of \colordef and \useK macros are declared using \_def, because the internal versions \_colordef and \_useK are changed during processing.

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

The \_showcolor macro should be re-defined for color catalog printing. For example:

\def\vr{\vrule height10pt depth2pt width20pt}
\def\_showcolor{\hbox{\tt\bslash \_tmpb: \csname \_tmpb\endcsname \vr}\space\space}
\begmulti 4 \typosize[11/14]
\morecolors
\endmulti

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. Opt\TeX reads this file at the beginning of the document (using \everyjob) if such file exists. The .ref file looks like:

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

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

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

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

If a macro needs to create and to use `.ref` file then such macro must use `\openref`. When the file is created (using internal `\_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\langle foo\rangle`. 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{\langle version\rangle}`

We can check the version compatibility by this macro. Because OPmac does not understand `\Xrefversion` we use `\Xrefversion` (with different number of `\langle version\rangle` 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\{\langle definitions of your ref macros\rangle\}`. This command sends to `.ref` file your `\langle definitions of your ref macros\rangle` immediately. Next lines in `.ref` file should include our macros. Example from CTUstyle2:

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

We must read `\langle definition of your ref macros\rangle` 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.

\ref\{References <2020-03-03}\} % preloaded in format
\Xpage\{(gpageno)\}{(pageno)} saves the parameter pair into \currpage. Resets _lfnotenum; it is used if footnotes are numbered from one at each page.
\Xlabel\{(label)\}{(text)} saves the \(\langle\text\rangle\) to _lab:\{(label)\} and saves \{pg:\{gpageno\}\}{(pageno)} to \pgref:\{(label)\}.
\label\[\{(label)\}\] saves the declared label to _lastlabel and \wlabel\{(text)\} uses the _lastlabel and activates \wref\Xlabel\{(label)\}{(text)}.

Default _printlabel is empty macro (labels are not printed). The _showlabels redefines it as box with zero dimensions and with left lapped [\{(label)\}] in blue 10pt \tt font shifted up by 1.7ex.
2.22 Hyperlinks

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

- **ref**:\{label\} – the destination is created when \label\{\{label\}\} is used, see also the section 2.21.
- **toc**:\{tocrefnum\} – the destination is created at chap/sec/secc titles, see also the section 2.23.
- **pg**:\{gpageno\} – the destination is created at beginning of each page, see also the section 2.17.
- **cite**:\{bibnum\} – the destination is created in bibliography reference, see also the section 2.31.1.
- **url**:\{url\} – used by \url\{\url\} or \url\{\url\}, see also the end of this section.

The \{tocrefnum\}, \{gpageno\} and \{bibnum\} are numbers starting from one and globally incremented by one in whole document. The registers \{tocrefnum\}, \{gpageno\} and \{bibnum\} are used for these numbers.

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

\{\dest\{type\}\\{spec\}\\} creates a destination of internal links. The destination is declared in the format \{type\}\\{spec\}. If the \hyperlinks command in not used, then \dest does nothing else it is set to \destactive. The \destactive is implemented by \pdfdest primitive. It creates a box in which the destination is shifted by \destheight. The reason is that the destination is exactly at top border of the PDF viewer but we want to see the line where destination is. The destination box is positioned by sety.\destactive creates a destination of internal links. The destination is created when \dest\{\{label\}\} is used, see also the section 2.21.

\{\link\{type\}\\{spec\}\\}\\{\color\}\\{\text\} creates an internal link to \dest with the same \{type\}\\{spec\}. You can have more links with the same \{type\}\\{spec\} but only one \dest in the document. If \hyperlinks command is not used, then \link only prints \text else it is set to \linkactive. The \linkactive is implemented by \pdfdestlink...\pdfdestlink primitives.

\{\url\}_{\text} is equivalent to \link but the \{color\} is used from \hyperlinks declaration.

\{\ulink\}_{\text} creates external link. It prints only te \text by default but the \hyperlinks declaration defines it as \urlactive\{url\}_{\text}. The external link is created by the \pdfdestlink...\pdfdestlink primitives. The \{url\} is detokenized with \escapechar=-1 before it is used, so \%, \# etc. can be used in the \{url\}.
The \_pdfstartlink primitive uses \_pdfborder{} in its parameter (see \_linkactive or \_urlactive macros). The \_pdfborder{} expands to attr{/C[? ? ?] /Border[0 0 .6]} if the \{type\}border (i.e. \refborder, \citeborder, \tocborder, \pgborder, \urlborder, \ntfborder or \fnfborder) is defined. User can define it in order to create colored frames around active links. For example \def\tocborder{1 0 0} causes red frames in TOC (not printed, only visible in PDF viewers).

\def\pdfborder#1{\ifcsname #1border\endcsname
\attr{/C{\csname #1border\endcsname} /Border[0 0 .6]}\else \attr{/Border[0 0 0]}\fi}

\_def\hyperlinks{\ilink_color}\{\ulink_color\} activates \dest, \link, \ilink, \ulink in order they create links. These macros are redefined here to their “active” version.

\def\hyperlinks#1#2{\let\dest=\destactive \let\link=\linkactive
\def\ilink[#1]{\link[#1]{\localcolor#1}{#2}}\def\ulink[#1]{\urlactive[url:#1]{\localcolor#2}{#2}}
\public \dest \ilink \ulink
\public \hyperlinks
\url{\langle url \rangle} does approximately the same as \ulink{\langle url \rangle}{\langle url \rangle}, 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.

\def\url#1{{%\def\tmpa{#1}\replstring\tmpa {\|}{}%\edef\tmpa{\detokenize\tmpa}\def\tmpb{#1}\replstring\tmpb {\|}{\urlbskip}\replstring\tmpb {\//} {{\urlskip\urlbskip\urlbskip}}\replstring\tmpb {\//} {{\urlskip\urlbskip}}\replstring\tmpb {\//} {{\urlskip\urlbskip}}\replstring\tmpb {\?} {{\urlskip\urlbskip}}\replstring\tmpb {=} {{\urlskip\urlbskip}}\ea\replstring\ea\tmpb\ea\string \& {{\urlbskip\char`\& \urlskip}}\ea\bslash\ea\string \& {{\penalty0}}\ea\url{\langle url \rangle}\ea\urlfont\null\ea}}
\def\urlfont{\tt}
\def\urlskip{\null\nobreak\hskip0pt plus0.05em\relax}
\def\urlskip100 {\null\nobreak\hskip0pt plus0.05em\relax}
\def\urlslasheslash{/\urlskip}/

2.23 Making table of contents

\_codedecl \maketoc {Macros for maketoc <2020-03-12>} % preloaded in format
\_Xtoc{\langle level\rangle}{\langle type\rangle}{\langle number\rangle}{\langle title\rangle} (in .ref file) reads the specified data and appends them to the \_toclist as \_tocline{\langle level\rangle}{\langle type\rangle}{\langle number\rangle}{\langle title\rangle}{\langle gpageno\rangle}{\langle pageno\rangle} where:

- \langle level\rangle: 0 reserved, 1: chapter, 2: section, 3: subsection
- \langle type\rangle: the type of the level, i.e. chap, sec, secc
- \langle number\rangle: the number of the chapter/section/subsection in the format 1.2.3
- \langle title\rangle: the title text
- \langle gpageno\rangle: the page number numbered from 1 independently of pagination
- \langle pageno\rangle: the page number used in the pagination
The last two parameters are restored from previous \_Xpage\{(pageno)\}\{\(gpage\)\}, data were saved in the \_currpage macro.

\_tocline\{(level)\}\{(type)\}\{(number)\}\{(title)\}\{(gpage)\}\{\(pageno\)\} prints the record to the table of contents. It opens group, reduces \_leftskip, \_rightskip, runs the \everytocline (user can customise the design of TOC here) and runs \_tocl: \{(level)\}\{(number)\}\{(title)\}\{(pageno)\} macro. This macro starts with vertical mode, inserts one record with given \(level\) and it should end by \_tocpar which returns to horizontal mode. The \_tocpar appends \_nobreak \_hskip-2\_iindent\_null \_par. This causes that the last line of the record is shifted outside the margin given by \_rightskip. A typical record (with long \(title\)) looks like:

\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 .................... <pageno>

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

You can re-define default macros for each level of tocline if you want. Parameters are \{(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.
- \tocpar finalizes one TOC recors with rlapped \_pageno. \_pageno\{\(pageno\)\} 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 \_rectoc, \_regmark and \_regoul. These token lists are used in \maketoc, \begoutput and \pdfunidef.

\maketoc.opm

\newtoks \_regtoc \newtoks \_regmark \newtoks \_regoul
\def \_regmacro #1#2#3{\_toksapp \_regtoc {#1} \_toksapp \_regmark {#2} \_toksapp \_regoul {#3}}
\public \maketoc \_regmacro ;

2.24 PDF outlines

2.24.1 Nesting PDF outlines

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

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

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

\advance \count2 by 1
% increases the mapping pointer of the type
% \_ol:\_count0: \_count1: \_count2 of this section
\advance \_ol: \_count0: \_count1 by 1
% increases the number of descendants connected
% to the parent of this section.

When second step is processed, then we only read the stored data about the number of 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.

\insertoutline {⟨text⟩} inserts one outline with zero direct descendants. It creates link destination of the type \_ol:⟨num⟩ into the document (where \insertoutline is used) and the link itself is created too in the outline.

\outlines.opm

3 \_codedecl \outlines {PDF outlines <2020-03-12>} % preloaded in format
4 \def \outlines#1{\_pdfcatalog{PageMode/UseOutlines}\_openref
5 \_ifx \_toclist \_empty \_opwarning{\_noexpand\outlines -- data unavailable. TeX me again}\_openref
6 \_else \_endgroup \fi
7 \_opwarning{\_noexpand\outlines -- data unavailable. TeX me again}\_openref
8 \{ \_let \_tocline= \_outlinesA
9 \_count0=0 \_count1=0 \_count2=0 \_count3=0 \_toclist \% calculate numbers o childs
10 \_ifx \_dest \_destactive \_else
11 \_opwarning{\_noexpand\outlines doesn't work when \_noexpand\hyperlinks isn't declared}\_fi
12 \{ \_let \_tocline= \_outlinesA
13 \_count0=0 \_count1=0 \_count2=0 \_count3=0 \_toclist \% create outlines
14 \_toclist\}% create outlines
15 \_fi
16 \fi
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-discomfortable, because it looks like

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

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

The command \texttt{\pdfunidef\macro{string}} is implemented here using \texttt{\directlua}. The input string is preprocessed: detokenized, converted \texttt{\word /} to \texttt{\word/} (used in logos) preprocessed spaces using \texttt{\replstring} and then the \texttt{\_pdfunidefB} is repeated on each character. It calls the \texttt{\directlua} chunk to print octal numbers in the macro \_octalprint.

The \texttt{\regmacro} is used in order to sed the values of macros \texttt{\em}, \texttt{\rm}, \texttt{\bf}, \texttt{\it}, \texttt{\bi}, \texttt{\tt} and \texttt{~} to values usable in PDF outlines.
2.25 Chapters, sections, subsections

We are using scaled fonts for titles \_titfont, \_chapfont, \_secfont and \_seccfont. They are scaled from main fonts size of the document, which is declared by first \texttt{\typosize[⟨fo-size⟩/⟨b-size⟩]} command.

The \texttt{\_titfont} macro is defined by \texttt{\_eoldef}, it means that the parameter is separated by end of line. The macros \_chap, \_sec and \_secc use \_eoldef too.

You can re-define \_printchap, \_printsec or \_printsecc macros if another design of section titles is needed. These macros gets the \texttt{⟨title⟩} text in its parameter. The common recommendations for these macros are:

- Use \_abovetitle{⟨penaltyA⟩}{⟨skipA⟩} and \_belovtitle{⟨skipB⟩} for inserting vertical material above and below the section title. The arguments of these macros are normally used, i.e. \_abovetitle inserts \texttt{⟨penaltyA⟩}{⟨skipA⟩} and \_belovtitle inserts \texttt{⟨skipB⟩}. But there is an exception: if \_belovtitle{⟨skipB⟩} is immediately followed by \_abovetitle{⟨penaltyA⟩}{⟨skipA⟩}

(for example section title is immediately followed by subsection title), then only \( \langle \text{skipA} \rangle \) is generated, i.e. \( \langle \text{skipA} \rangle \langle \text{penaltyA} \rangle \langle \text{skipA} \rangle \) is reduced only to \( \langle \text{skipA} \rangle \). The reason of such behavior: we don’t want to duplicate vertical skip and we don’t want to use negative penalty in such cases. Moreover, \( \langle \text{belowtitle} \rangle \langle \text{penaltyA} \rangle \langle \text{skipA} \rangle \) takes previous whatever vertical skip (other than from \( \langle \text{belowtitle} \rangle \)) and generates only greater from this pair of skips. It means that \( \langle \text{whatever-skip} \rangle \langle \text{penaltyA} \rangle \langle \text{skipA} \rangle \) is transformed to \( \langle \text{penaltyA} \rangle \langle \text{whatever-skip} \rangle \langle \text{skipA} \rangle \). The reason of such behavior: we don’t want to duplicate vertical skips (from \( \langle \text{belowlistskip} \rangle \)) above the title.

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

- Use \( \_\text{nbpar} \) for closing the paragraph for printing title. This command inserts \( \_\text{nobreak} \) between each line of such paragraph, so the title cannot be broken to more pages.

- You can use \( \_\text{firstnoindent} \) in order to the first paragraph after the title is not indented.

The \( \_\text{sectionlevel} \) is the level of the printed section:

- \( \_\text{sectionlevel}=0 \) – reserved for parts of the book (unused by default)
- \( \_\text{sectionlevel}=1 \) – chapters (used in \( \_\text{chap} \))
- \( \_\text{sectionlevel}=2 \) – sections (used in \( \_\text{sec} \))
- \( \_\text{sectionlevel}=3 \) – subsections (used in \( \_\text{secc} \))
- \( \_\text{sectionlevel}=4 \) – subsubsections (unused by default)

The \( \_\text{chapx} \) initializes counters used in chapters, the \( \_\text{secx} \) initializes counters in sections and \( \_\text{seccx} \) initializes counters in subsections. If you have more types of numbered objects in your document then you can declare appropriate counters and do \( \_\text{addto\text{\_chapx}} \langle \text{yourcounter} \rangle \) 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 \( \_\text{globaldefs} \langle \_\text{chapx} \rangle \).

Default concept: Tables, figures and display maths are numbered from one in each section – subsections don’t reset these counters. Footnotes declared by \( \_\text{footnumchapters} \) are numbered in each chapter from one.

The \( \_\text{the*} \) macros \( \_\text{thechapnum} \), \( \_\text{theseccnum} \), \( \_\text{theseccnum} \), \( \_\text{thetnum} \), \( \_\text{thefnum} \) and \( \_\text{thednum} \) include the format of numbers used when the object is printing. If chapter is never used in the document then \( \_\text{chapnum}=0 \) and \( \_\text{othe\text{\_chapnum}} \) expands to empty. Sections have numbers \( \langle \text{num} \rangle \) and subsections have numbers \( \langle \text{num} \rangle \langle \text{num} \rangle \langle \text{num} \rangle \). On the other hand, if chapter is used in the document then \( \_\text{chapnum}>0 \) and sections have numbers \( \langle \text{num} \rangle \langle \text{num} \rangle \langle \text{num} \rangle \langle \text{num} \rangle \langle \text{num} \rangle \).
The \notoc and \nonum prefixes are implemented by internal \_ifnotoc and \_ifnonum. They are reset after each chapter/section/subsection by the \_resetnonumnotoc macro.

The \_printrefnum macro is used in \_print* macros.

Note that the \_wtotoc macro expands its parameters and does \_wref.

The \_printrefnum macro is used in \_print* macros.

The \_wtotoc macro expands its parameters and does \_wref. The problem of “fragile macros” from old \LaTeX{} never occurs.
The \aboveitle{\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 \aboveitle reads \lastpenalty and if it has this special value then it removes the skip used before and don’t use the parameter. The \aboveitle 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.

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

\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.

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 \insmrk{\title text} saves \mark in the format \{\title-num\} {\title-text}, so it can be printed “as is” in \headline (see the space between them), or you can define a formating macro with two parameters for processing these data, if you need it.

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{\ifempty{#1}{\false #1: \false #2}\fi} \headline = \% \ifodd \pageno \hfil \ea{\formathead{firstmark}{}\% \else \hfil \ea{\formathead{firstmark}{}\% \fi}
```

108
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.

\begin{verbatim}
sections.opm
306 \_def\_caption/#1{\_def\_tmpa{#1}\_nospaceafter \_capA}
307 \_optdef\_capA []{{\trylabel \_incaption}
308 \_def\_incaption {}{\_bgroup
309 \_ifcsname _\_tmpa num\_endcsname \_ea\_incr \_csname _\tmpa num\_endcsname
310 \_else \_opwarning{Unknown caption /_\_tmpa}\_fi
311 \_edef\_thecapnum {\_csname _the\_tmpa num\_endcsname}%
312 \_edef\_thecaptitle{\_mtext{\_tmpa}}%
313 \_ifcsname _everycaption\_tmpa\_endcsname
314 \_ea\_the \_csname _everycaption\_tmpa\_endcsname
315 \_def\_par{\_nbpar\_egroup}\_let\par=\_par
316 \_cs{\_printcaption\_tmpa}%
317 }
318 \_def \_cskip {\_par\nobreak\medskip} % space between caption and the object
319 \_public \_caption \cskip ;
\end{verbatim}

The \_printcaption and \_printcaptionf macros start in vertical mode. They switch to horizontal mode and use \_wlabel\_thecapnum (in order to make reference and hyperlink destination) a they can use:

- \_thecaptitle ... expands to the word Table or Figure (depending on the current language).
- \_thecapnum ... expands to \_thecapnum (caption number).

\begin{verbatim}
sections.opm
333 \_def \_printcaption {}{%
334 \_noindent \_wlabel\_thecapnum {\_bf\_thecaptitle~\_thecapnum}\_enspace
335 \_narrowlastlinecentered\_indent
336 }% start
337 \_let \_printcaptionf = \_printcaption % caption of figures = caption of tables
\end{verbatim}

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.

\begin{verbatim}
sections.opm
345 \_def\_narrowlastlinecentered{%
346 \_leftskip=#1plus1fil
347 \_rightskip=#1plus-1fil
348 \_parfillskip=0pt plus2fil
349 }
\end{verbatim}

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

\begin{verbatim}
sections.opm
356 \_optdef\_eqmark []{{\trylabel \_ineqmark}
357 \_def\_ineqmark{\_incr\_dnum
358 \_ifinner\_else\eqno \_fi
359 \_wlabel\_thednum \_thednum}
360 }% end
361 \_public \_eqmark ;
\end{verbatim}

The \_numberedpar ⟨letter⟩{⟨name⟩}) is implemented here.

\begin{verbatim}
sections.opm
367 \_newcount\_counterA \_newcount\_counterB \_newcount\_counterC
368 \_newcount\_counterD \_newcount\_counterE
369 \_def\_resetABCDE{\_counterA=0 \_counterB=0 \_counterC=0 \_counterD=0 \_counterE=0 }
370 \_def\_theAnum{\_othe\_chapnum.\_othe\_secnum.\_the\_counterA}
371 \_def\_theBnum{\_othe\_chapnum.\_othe\_secnum.\_the\_counterB}
372 \_def\_theCnum{\_othe\_chapnum.\_othe\_secnum.\_the\_counterC}
373 \_def\_theDnum{\_othe\_chapnum.\_othe\_secnum.\_the\_counterD}
374 \_def\_theEnum{\_othe\_chapnum.\_othe\_secnum.\_the\_counterE}
375 \_def\_numberedpar\_#1\#2{\_ea\_incr \_csname _counter\#1\_endcsname
376 \_def\_tmpa(#1)\_def\_tmpb(#2)\_numberedparparam}
\end{verbatim}

109
The \_printnumberedpar \theXnum \langle{name}\rangle opens numbered paragraph and prints it. The optional parameter is in \_the\_opt. You can re-define it if you need another design. \_printnumberedpar needs not to be re-defined if you only want to print Theorems in italic and to insert vertical skips (for example). You can do this by the following code:

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

2.26 Lists, items

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 indentes by new \indent from left. Default item mark is \_printitem. The \begitems runs \_aboveliskip only if we are not near below a title, where a vertical skip is placed already and where the \penalty 11333 is. It activates * and defines it as \_startitem.

The \enditems runs \_isnextchar\_par{}\{\_noindent\} thus the next paragraph is without indentation if there is no empty line between the list and this paragraph (it is similar behavior as after display math).

\def\begitems\{\par
\bgroup
\_advance \_ilevel by1
\_setlistskip
\ifnum \lastpenalty<10000 \_aboveliskip \_fi
\_itemnum=0 \_adef*{\_startitem}
\_advance \_leftskip by\_iindent
\_printitem=\_defaultitem
\the\_everylist \_relax
\}
Various item marks are saved in \item:<letter> macros. You can re-define then or define more such macros. The \style<letter> does \printitem={\item:<letter>}. More exactly: \begitems does \printitem=\defaultitem first, then \style<letter> does \printitem={\item:<letter>} when it is used and finally, \startitem alias * uses \printitem.

\athe<(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

The internal parameters \ttskip, \ttpenalty, \viline, \vifile and \ttfont for verbatim macros are set.

\code{(text)} expands to \detokenize{(text)} when \escapechar=-1. In order to do it more robust when it is used in \write then it expands as noexpanded \code{space} (followed by space in its csname). This macro does the real work.

The \printinverbatim{(text)} macro is used for \code{(text)} printing and for `<(text)` printing. It is defined as \ hbox{(followed by space in its csname). This macro does the real work.

When \code occurs in PDF outlines then it does the same as \detokenize. The macro for preparing outlines sets \escapechar to –1 and uses \regoul token list before \edef.
The \code is not \protected because we want it expands to \unexpanded\{\code\space\}{\text}. This protect the expansions of the \code parameter (like \\text{,} \text{^} etc.).

\verbatim.opm

36 \def\_code#1{\_unexpanded\_ea{\csname _code \_endcsname{#1}}}
37 \protected\_edef{\_code}{#1{\_escapename=-1 \_ttfont \_the\everyintt \_relax
38 \_ea\_printinverbatim\_ea{\_detokenize(#1)}}}
39 \_def\_printinverbatim\_ea{\_detokenize\_ea\_detokenize\_let\_code=\_detokenize}
40 \public \_code \;
41

The \_setverb macro sets all catcodes to “verbatim mode”. It should be used only in a group, so we prepare a new catcode table with “verbatim” catcodes and we define it as \_catcodetable\_verbatimcatcodes. After the group is finished then original catcode table is restored.

\verbatim.opm

51 \_newcatcodetable \_verbatimcatcodes
52 \_def\_setverb{\_begingroup
53 \_def\do##1{\_catcode`##1=12}
54 \_dospecials
55 \_savecatcodetable\_verbatimcatcodes % all characters are normal
56 \_endgroup
57 }
58 \_setverb
59 \_def\_setverb{\_catcodetable\_verbatimcatcodes }% \activettchar

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

\verbatim.opm

\_def *\{\_begingroup \_setverb ... \_readverb\}
\_def \_readverb #1*\{\_printinverbatim{#1}\_endgroup\}

Note that the second occurrence of * is not active because \_setverb deactivates it.

\verbatim.opm

\begtt is defined only as public. We don’t need private \begtt variant. This macro is defined by \eoldef, so user can put a parameter at the same line where \begtt is. This \#1 parameter is used after \_everytt parameters settings, so user can change them locally.

The \begtt macro opens group, does \_setverb and another preprocessing, sets \_endlinechar to ^J and reads the following text in verbatim mode until \endtt occurs. This scanning is done by \_startverb macro which is defined as:

\verbatim.opm

\_def \_startverb #1\endtt #2\_endgroup{...}
\_def \_startverb #1\endtt #2\_endgroup{...}

We must to ensure that the backslash in \endtt has category 12 (this is a reason of the \_ea chain in real code). The \#2 is something between \endtt and end of the same line and it is simply ignored.

The \_startverb puts the scanned data to \_prepareverdata. 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 \_prepareverdata finishes then \_startverb runs \_printverb loop over each line of the data and does a final work: last skip plus \_noindent in the next paragraph.

The \_printverb macro calls \_printverbline{\line} to each scanned line of verbatim text. This macro expect that it 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 `_viscanminus`. Finally the `_doverbinput` is run. At beginning of `_doverbinput`, we have `_viline` = number of lines already read using previous `\verbinput`, `_vinolines` = the number of lines we need to skip and `_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 `\prepareverbmacro` loop.
The `\textbackslash visiblesp` sets spaces as visible characters \textbackslash␣. It redefines \textbackslash␣ primitive, so it is useful for verbatim modes only.

\begin{verbatim}

def \textbackslash visiblesp{\ifx \initunifonts \relax \def \{	extbackslash textbackslash char\textbackslash#251} \else \def \{\textbackslash char\textbackslash32} \fi

\def \textbackslash visiblesp{\ifx \initunifonts \relax \def \{\textbackslash char\textbackslash261} \else \def \{\textbackslash char\textbackslash32} \fi

2.27.2 Listings with syntax highlighting

The user can write

\begtt \hisynax{C}

114
and the code is colorized by C syntax. The user can write `\everytt={\hisyntax{C}}` and all verbatim listings are colorized.

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

```latex
\hisyntax-c.opm
```

OpTeX provides `hisyntax-{c,python, tex, html}.opm` files. You can take inspiration from these files and declare more languages.

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

Another way to set non-default colors is to declare `\newtoks\hicolors` (without the `_`) prefix and set the colors palette here. It has precedence before `\_hicolors` (with the `_`) prefix declared in the `hicolors-⟨name⟩.opm` file. This is useful when there are more hi-syntax languages used in one document.

Notes for hi-syntax macro writers

The file `hisyntax-⟨name⟩.opm` is read only once in the TeX group. If there are definitions then they must be declared as global.

The `hisyntax-⟨name⟩.opm` file must (globally) declare `\hisyntax{⟨name⟩}` tokens string where the action over verbatim text is declared typically by `\replfromto` or `\replthis` macros.
The verbatim text is prepared by **pre-processing phase**, then the \_hisyntax\{name\} is applied and then **post-processing phase** does final corrections. Finally, the verbatim text is printed line by line.

The pre-processing phase does:

- Each space is replaced by \n\n\n, so \n\{word\}\n should be a pattern to finding whole words (no subwords). The \n control sequence is removed in the post-processing phase.
- Each end of line is represented by \n\n\n.
- The \_start control sequence is added before the verbatim text and \_end control sequence is appended to the end of the verbatim text. These control sequences are removed in post-processing phase.

There are special macros working only in a group when processing the verbatim text.
- \n means noting but it should be used as a boundary of words as mentioned above.
- \t means a tabulator. It is prepared as \n\t\n because it can be at the boundary of a word.
- \x \{letter\}\{\{text\}\} can be used as replacing text. Suppose the example

\replfromto{/*}{*/}{\x C{/*#1*/}}

This replaces all C comments /\*...\*/ by \x C{:\*...:\*}. But the C comments may span more lines, i.e. the \-J should be inside it. The macro \x \{letter\}\{\{text\}\} is replaced by one or more \z \{letter\}\{\{text\}\} in post-processing phase where each parameter \{text\} of \z keeps inside one line. Inside-line parameters are represented by \x C\{\{text\}\} and they are replaced to \z C\{\{text\}\} without any change. But:

\x C\{\{text1\}\}--\text3\} is replaced by
\z C\{\{text1\}\}--\text3\}\z C\{\{text3\}\}

The \z \{letter\}\{\{text\}\} is expanded to \_z:\{letter\}\{\{text\}\} and if \zcolor \{letter\} \{color\} is declared then \_z:\{letter\}\{\{text\}\} expands to \{color\}\{\{text\}\}. So, required color is activated at all lines (separately) where C comment spans.
- \y \{\{text\}\} is replaced by \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 \replfromto and \replthis manipulate with the verbatim text which has been read already and stored in the \_tmpl macro.

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

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

First two parameters are used before usage of \replfromto. You can use \csstring\% or something else here.

The \replthis \{\{pattern\}\}\{\{what\}\}\} replaces each \{pattern\} by \{what\}. Both parameters of \replthis are expanded first.

```latex
3 \_codedcl \hisyntax {Syntax higlighting of verbatim listings <2020-04-04> \% preloaded in format

hi-syntax.opm}

The following macros \replfromto and \replthis manipulate with the verbatim text which has been read already and stored in the \_tmpl macro.

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

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

First two parameters are used before usage of \replfromto. You can use \csstring\% or something else here.

```
The patterns \langle from \rangle, \langle to \rangle and \langle pattern \rangle are not found when they are hidden in braces \{...\}. Example:

\replfromto{/*}{*/}{\x C{/*#1/*}}

replaces all C comments by \x C{...}. The patterns inside \{...\} are not used by next usage of \replfromto or \replthis macros.

The \_xscan macro does replacing \x by \z in the post-processing phase. The \x \langle letter \rangle \langle text \rangle expands to \xscan \langle letter \rangle \langle text \rangle ^^J. If \#3 is \_end then it signals that something wrong happens, the \langle from \rangle was not terminated by legal \langle to \rangle when \replfromto did work. We must to fix it by the \_xscanR macro.

The \hicolor \langle letter \rangle \langle color \rangle defines \_z \langle letter \rangle \langle text \rangle as \{\langle color\rangle\langle text\rangle\}. It should be used in the context of \langle letter \rangle \langle text \rangle macros.

The \hisyntax\langle name \rangle re-defines default \_prepareverbdata\langle macro \rangle \langle verbtext \rangle in order to it does more things: It saves \langle verbtext \rangle to \_tmpb, appends \_n around spaces and ^^J characters in pre-processing phase, it opens \hisyntax\langle name \rangle. opm file if \_hisyntax\langle name \rangle is not defined. Then \_the\_isyntax\langle name \rangle is processed. Finally, the post-processing phase is realized by setting appropriate values to \_x and \_y macros and doing \_edef\_tmpb{\_tmpb}.

Aliases 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

\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 \pdffile 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.

```
\def\inspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inspicB\_inspicA}
\def\_inspicA #1 {\_inspicB {#1}}
\def\_inspicB #1{% 
  \pdffile \_ifdim\picwidth=0pt \_else width \_picwidth\_fi 
  \_ifdim\picheight=0pt \_else height \_picheight\_fi 
  \_picparams {\_the\_picdir#1}%
  \pdffileximage \_pdflastximage\_egroup}
\def\_picparams{}
\public \inspic ;
```

Inkscape is able to save a picture to *.pdf file and labels for the picture to *.pdf_tex file. The second file is in \LaTeX format (unfortunately) and it is intended to read immediately it after *.pdf in included in order to place labels of this picture in the same font as document is printed. We need to read this \LaTeX macros when \inkinspic is used. These macros are stored in the \_inkdefs toknes list and it is used locally in the group. The solution is borrowed from OPmac trick 0032.

```
\def\inkinspic{\_hbox\_bgroup\_isnextchar\_bgroup\_inkinspicB\_inkinspicA}
\def\_inkinspicA #1 {\_inkinspicB {#1}}
\def\_inkinspicB #1{% 
  \_ifdim\picwidth=0pt \_setbox0=\_hbox{\_inspic{#1}}\_picwidth=\_wd0 \_fi
  \_the\_inkdefs
  \opinput {\_the\_picdir #1_tex}% file with labels
  \_egroup}
\newtoks\_inkdefs \_inkdefs={% 
  \def\makeatletter#1\makeatother{}%
  \def\includegraphics[#1]{\_inkscanpage#1,page=,\_end \_inspic{#2}\_hss}%
  \def\_inkscanpage#1page=#2,#3\_end{\_ifx,#2,,\_else\_def\_picparams{page#2}\_fi}%
  \def\put(#1,#2)#3{\_nointerlineskip\_vbox to0pt{\_vss\_hbox to0pt{\_kern#1\_picwidth\_pdfsave\_hbox to0pt{#3}\_pdfrestore\_hss}\_kern#2\_picwidth}}%
  \def\begin#1{\_csname _begin#1\_endcsname}
  \def\_beginpicture(#1,#2){\_vbox\_bgroup
    \hbox to\_picwidth\_def\_end##1{\_egroup}}%
  \def\_begintabular[#1]{#2#3\_end}{\_vtop{\_def\_cr{\_cr}\_table{#2}{#3}}}%
  \def\_color[#1]{\_scancolor #2,}%
  \def\_scancolor#1,#2,#3,\_pdfliteral{\#1 \#2 \#3 rg}]%
  \def\_makebox(#1)[#2]#3{\_hbox to0pt{\_csname _mbx:#2\_endcsname{#3}}}%
  \def\_sdef{\_mbx:lb}#1{#1\_hss}\_sdef{\_mbx:rb}#1{\_hss#1}\_sdef{\_mbx:t}{#1\_hss}\_sdef{\_mbx:lt}{#1\_hss}
  \def\_rotatebox#1#2{\_pdfrotate{#1}#2}
  \def\_lineheight#1{}%
  \def\_setlength#1#2{}%
  }
\public \inkinspic ;
```

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

```
\def\_pdfscale#1#2{\_pdfsetmatrix{#1 0 0 #2}}
\def\_gonfunc#1#2{%
  \_directlua{\text{print(string.format(\_pcent.4f,math.\_#1(3.14159265*(\_#2)/180)))}}%
}\def\_sin{\_gonfunc(sin)}
```
\_def\_cos\{\_gonfunc{\cos}\}
\_def\_pdfrotate#1\{_pdfsetmatrix\_cos{#1} \_sin{#1} \_sin{(#1)-180} \_cos{#1})\}
\_public \pdfscale \pdfrotate ;

The \transformbox{\langle transformation \rangle} \{\langle text \rangle\} is copied from OPmac trick 0046. The \rotbox{\langle degrees \rangle} \{\langle text \rangle\} is a combination of \rotsimple from OPmac trick 0101 and the \transformbox. Note, that \rotbox{-90} puts the rotated text to the height of the outer box (depth is zero) because code from \rotsimple is processed. But \rotbox{90.0} puts the rotated text to the depth of the outer box (height is zero) because \transformbox is processed.

\def\multiplyMxV #1 #2 #3 #4 {% matrix \_tmpdim = #1\_vvalX \_advance\_tmpdim by #3\_vvalY
\_vvalY = #4\_vvalY \_advance\_vvalY by #2\_vvalX
\_vvalX = \_tmpdim
}
\def\multiplyMxM #1 #2 #3 #4 {% currmatrix := currmatrix * matrix
\_vvalX=#1pt \_vvalY=#2pt \_ea\_multiplyMxV \_currmatrix
\_edef\_tmpb{\_ea\_ignorept\_the\_vvalX\_space \_ea\_ignorept\_the\_vvalY}\%
\_vvalX=#3pt \_vvalY=#4pt \_ea\_multiplyMxV \_currmatrix
\_edef\_currmatrix{\_tmpb\_space}
}
\_def\transformbox#1#2{
\_hbox{\_setbox0=\_hbox{{#2}}%
\_dimendef\vvalX 11 \_dimendef\vvalY 12 % we use these variables
\_dimendef\newHt 13 \_dimendef\newDp 14 % only in this group
\_dimendef\newLt 15 \_dimendef\newRt 16
\_preptransform{#1}%
\_kern-\_newLt \_vrule height\_newHt \_depth\_newDp \_width0pt
\_setbox0=\_hbox{\_box0}\_ht0=0pt \_dp0=0pt
\_pdfsave\_pdfrotate\_box0\_pdfrestore \_kern\_newRt%
}
\_def\_preptransform #1{
\_def\currmatrix{1 0 0 1 }%
\_def\pdfsetmatrix##1{\_edef\_tmpb{##1 },\_ea\_multiplyMxM \_tmpb,\_unskip}%
\_let\pdfsetmatrix=\_pdfsetmatrix #1%
\_setnewHtDp 0pt \_ht0 \_setnewHtDp 0pt -\_dp0
\_setnewHtDp \_wd0 \_ht0 \_setnewHtDp \_wd0 -\_dp0
\_protected\_def \_pdfsetmatrix{\_pdfextension setmatrix}%
\_let\pdfsetmatrix=\_pdfsetmatrix
}
\_def\_setnewHtDp #1 #2 {%
\_vvalX=#1\_relax \_vvalY=#2\_relax \_ea\_multiplyMxV \_currmatrix
\_ifdim\_vvalX<\_newLt \_newLt=\_vvalX \_fi
\_ifdim\_vvalX>\_newRt \_newRt=\_vvalX \_fi
\_ifdim\_vvalY>\_newHt \_newHt=\_vvalY \_fi
\_ifdim-\_vvalY>\_newDp \_newDp=-\_vvalY \_fi
}
\_def\rotbox#1#2{% 
_isequal{90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_ht0 \_TMPDIM=-\_dp0}{\vfill}{#2}
_else \_isequal{-90}{#1}\_iftrue \_rotboxA{#1}{\_kern\_dp0 \_TMPDIM=\_ht0}{\#2}
_else \_transformbox{\_pdfrotate{#1}}{\#2}
_\_fi
}
\_def\rotboxA#1#2{\_hbox{\_setbox0=\_hbox{\_box0}}\_ht0=0pt \_dp0=0pt
\_pdfsave\_pdfrotate{\_box0}\_pdfrestore\_vfill%
\_\_kern\_TMPDIM
}
\_public \transformbox \rotbox ;

\scantwodimens scans two objects with the syntactic rule \langle dimen \rangle and returns \{\langle number \rangle\} \{\langle number \rangle\} in sp unit.
\puttext \langle right \rangle \langle up \rangle \{\langle text \rangle\} puts the \langle text \rangle to desired place: From current point moves \langle down \rangle and \langle right \rangle, puts the \langle text \rangle and returns back. The current point is unchanged after this macro ends.
\putpic \langle right \rangle \langle up \rangle \langle width \rangle \langle height \rangle \{\langle image-file \rangle\} does \puttext with the image scaled to desired \langle width \rangle and \langle height \rangle. If \langle width \rangle or \langle height \rangle is zero, natural dimension is used. The \nospec is a shortcut to such natural dimension.
\backgroundpic{⟨image-file⟩} puts the image to the background of each page. It it used in the slides style, for example.

\_def\_scantwodimens{%
\_directlua{tex.print(string.format('{\_pcent d}{\_pcent d}',
token.scan_dimen(),token.scan_dimen()))}%
}

defputtext{%
\_ifdim\_prevdepth>0pt \_vskip\_prevdepth \_relax \_fi
\_nointerlineskip
\wd0=0pt \_ht0=0pt \_dp0=0pt
\_vbox to0pt{\_kern\_dimen2 \_hbox to0pt{\_kern\_dimen1 \_box0\_hss}\_vss}}

defputpic{%
\_ifvmode
\_ifdim\_prevdepth>0pt \_vskip\_prevdepth \_relax \_fi
\_nointerlineskip
\wd0=0pt \_ht0=0pt \_dp0=0pt
\_vbox to0pt{\_kern\_dimen2 \_hbox to0pt{\_kern\_dimen1 \_box0\_hss}\_vss}}

def\_circle{⟨\text⟩}{y} creates an ellipse with ⟨x⟩ axis and ⟨y⟩ axix. The origin is in the center. \_oval{⟨\text⟩}{x}{y}{roundness} creates an oval with ⟨x⟩, ⟨y⟩ size and with given ⟨roundness⟩. The real size is bigger by 2⟨roundness⟩. The origin is at the left bottom corner. \_mv{⟨\text⟩}{x}{y}{curve} moves current point to ⟨x⟩, ⟨y⟩, crates the ⟨curve⟩ and retuns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.

\_circle{⟨\text⟩}{y} creates an ellipse with ⟨x⟩ axis and ⟨y⟩ axix. The origin is in the center.
\_oval{⟨\text⟩}{x}{y}{roundness} creates an oval with ⟨x⟩, ⟨y⟩ size and with given ⟨roundness⟩. The real size is bigger by 2⟨roundness⟩. The origin is at the left bottom corner.
\_mv{⟨\text⟩}{x}{y}{curve} moves current point to ⟨x⟩, ⟨y⟩, crates the ⟨curve⟩ and returns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.

\_circle(⟨\text⟩){y} creates an ellipse with ⟨x⟩ axis and ⟨y⟩ axix. The origin is in the center.
\_oval(⟨\text⟩){x}{y}{roundness} creates an oval with ⟨x⟩, ⟨y⟩ size and with given ⟨roundness⟩. The real size is bigger by 2⟨roundness⟩. The origin is at the left bottom corner.
\_mv(⟨\text⟩){x}{y}{curve} moves current point to ⟨x⟩, ⟨y⟩, crates the ⟨curve⟩ and returns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.

\_circle(⟨\text⟩){y} creates an ellipse with ⟨x⟩ axis and ⟨y⟩ axix. The origin is in the center.
\_oval(⟨\text⟩){x}{y}{roundness} creates an oval with ⟨x⟩, ⟨y⟩ size and with given ⟨roundness⟩. The real size is bigger by 2⟨roundness⟩. The origin is at the left bottom corner.
\_mv(⟨\text⟩){x}{y}{curve} moves current point to ⟨x⟩, ⟨y⟩, crates the ⟨curve⟩ and returns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.

The \incircle{⟨text⟩} is an example of \_oval usage.
The \incircle{⟨text⟩} is an example of \_circle usage.
The \ratio, \_width, \fcolor, \lcolor, \shadow and \overlapmargins are parameters, they can be set by user in optional brackets [...]. 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.

\_circle(⟨\text⟩){y} creates an ellipse with ⟨x⟩ axis and ⟨y⟩ axix. The origin is in the center.
\_oval(⟨\text⟩){x}{y}{roundness} creates an oval with ⟨x⟩, ⟨y⟩ size and with given ⟨roundness⟩. The real size is bigger by 2⟨roundness⟩. The origin is at the left bottom corner.
\_mv(⟨\text⟩){x}{y}{curve} moves current point to ⟨x⟩, ⟨y⟩, crates the ⟨curve⟩ and returns back the current point. All these macros are fully expandable and they can be used in the \pdfliteral argument.
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.

A generic macro \_clipinpath{\langle x \rangle \langle y \rangle \langle curve \rangle \langle 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 width \rangle \langle height \rangle \langle text \rangle} and \_clipincircle{\langle x \rangle \langle y \rangle \langle width \rangle \langle height \rangle \langle text \rangle} are defined here. These macros read normal \TeX\ dimensions in their parameters.
2.29 The \table macro

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.

Categories (for example of \| character) have to be normal when reading \table parameters.

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 \scantabdata converts \table’s ⟨declaration⟩ to \halign⟨declaration⟩. The result is stored into \tabdata tokens list. For example, the following result is generated when ⟨declaration⟩=|cr||cl|. 123
The default “declaration letters” c, l, r and p are declared. by \def\tabdeclare{letter}{...} for a
non-parametric letter and by \def\paramtabdeclare{letter}{...} for a letter with a parameter. The
double hash ## must be in the definition, it is replaced by a real table item data. All items are put in
group because of \aftergroup can be used (from \localcolors for example). You can declare more
such “declaration letters” if you want.

The second result in the \ddlinedata macro is a template of one row of the table used by \crli macro.

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 \unskip 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 \multispan{⟨declaration⟩}{⟨text⟩} macro generates similar \omit\span\omit\span sequence as plain \TeX\ macro \multispan. Moreover, it uses _\scantabdata_ to convert \langle declaration⟩ from \table syntax to \halign syntax.

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⟩} 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.
2.30 Balanced multi-columns

This code is documented in detail in the “\texttt{TeX}book naruby”, pages 244-246, free available, \url{http://petr.olsak.net/tbn.html}, but in Czech. Roughly speaking, macros complete all material between \texttt{\begmulti\{num-columns\}} and \texttt{\endmulti} into one \texttt{\vbox}. Then the macro measures the amount of free space at the current page using \texttt{\pagegoal} and \texttt{\pagetotal} and does \texttt{\vsplit} of \texttt{\vbox} to columns with height of such free space. This is done only if we have enough amount of material in \texttt{\vbox} to fill full page by columns. This is repeated in loop until we have less amount of material in \texttt{\vbox}. Then we run \texttt{\_balancecolumns} which balances the last part of columns. Each part of printed material is distributed to main vertical list as \texttt{\vbox\{columns\}} and we need not do any change in the output routine.

If you have paragraphs in \texttt{\begmulti... \endmulti} environment then you may say \texttt{\raggedright} inside this environment and you can re-assign \texttt{\widowpenalty} and \texttt{\clubpenalty} (they are set to 10000 in \texttt{Op\TeX}).

Splitting columns...

\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\penalty0
  \hsize := column width = (\hsize+\colsep) / \Ncols - \colsep
  \divide\hsize by\Ncols \divide\hsize by\colsep
  \mullines=0
  \def\par{\ifhmode\endgraf\global\advance\mullines by\prevgraf\fi}%
  \def\endmulti{\vskip-\prevdepth\vfil
    \ea\egroup\ea\baselineskip\relax
    \dimen0=.8\maxdimen \tmpnum=\dimen0 \divide\tmpnum by\baselineskip
    \splittopskip=\baselineskip
    \setbox1=\vsplit6 to0pt
    \dimen1 := the free space on the page
    \ifdim\pagegoal=\maxdimen \dimen1=\vsize \corrsize{\dimen1}\fi
    \else \dimen1=\pagegoal \advance\dimen1 by-\pagetotal \fi
    \ifdim\dimen1<2\baselineskip
      \vfil\break \dimen1=\vsize \corrsize{\dimen1}\fi
    \ifnum\mullines<\tmpnum
      \dimen0=\ht6 \else \dimen0=.8\maxdimen \fi
    \divide\dimen0 by\Ncols \relax
    \ifx\balancecolumns\flushcolumns \advance\dimen0 by-.5\vsize \fi
    \multiskip\egroup
  }%}
\def\endmulti{\vskip-\prevdepth\vfil
  \ea\egroup\ea\baselineskip\relax
  \dimen0=.8\maxdimen \tmpnum=\dimen0 \divide\tmpnum by\baselineskip
  \splittopskip=\baselineskip
  \setbox1=\vsplit6 to\dimen1
  \repeat
  \hbox{}\nobreak\vskip\splittopskip \nointerlineskip
  \line{\unhbox\unskip}
  \dimen0=\dimen1 \divide\dimen0 by\baselineskip \multiply\dimen0 by\Ncols
  \global\advance\mullines by-\dimen0
  \egroup
}\def\splitpart{\%
  \makecolumns\egroup % 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{\unhbox\unskip}
  \dimen0=\dimen1 \divide\dimen0 by\baselineskip \multiply\dimen0 by\Ncols
  \global\advance\mullines by-\dimen0
  \egroup
}\def\splitpart{\%
  \makecolumns % full page
  \vskip Opt plus 1fil minus\baselineskip \break
  \ifnum\mullines<\tmppnum
    \dimen0=.8\maxdimen \else \dimen0=.8\maxdimen \fi
  \divide\dimen0 by\Ncols \relax
  \ifx\balancecolumns\flushcolumns \divide\dimen0 by-.5\vsize \fi
}
Final balancing of the columns.

\def\balancecolumns{\bgroup \setbox7=\copy6 % destination height: \dimen0
  \ifdim\dimen0>\baselineskip \else \dimen0=\baselineskip \fi
  \vbadness=20000
  \def\tmp{\setbox1=\hbox{} \tmpnum=0
    \loop \ifnum\Ncols>\tmpnum
      \advance\tmpnum by1
      \setbox1=\hbox{\unhbox1 \ifvoid6 \hbox to \wd6{\hss} \else \vsplit6 to \dimen0 \fi \hss}
    \repeat
    \ifvoid6 \else
      \advance\dimen0 by.2\baselineskip
    \fi
    \setbox6=\copy7 \ia \tmp \fi}
  \hbox{} \nobreak \vskip-\splittopskip \nointerlineskip
  \hbox to \hsize{\unhbox1 \unskip} \egroup}
\def\corrsize #1{\ifdim#1<0 \advance#1 by \baselineskip \fi}
\def\splitpart{\ifvoid6 \hbox to \wd6{\hss} \else \vsplit6 to \dimen0 \fi \hss}
\def\balancecolumns % last balancing

2.31 Citations, bibliography

2.31.1 Macros for citations and bibliography preloaded in the format cite-bib.opm

\codedecl \cite {Cite, Bibliography <2020-03-09>} % loaded in format

Registers used by \cite, \bib macros are declared here. The \bibnum counts the bibliography items from one. The \bibmark is used when \nonumcitations is set.

\newcount\bibnum % the bibitem counter
\newtoks\bibmark % the bibmark used if \nonumcitations
\newcount\lastcitenum % lastcitenum=0 % for \shortcitations
\newcommand\bibnum\bibmark ;
\public\begmulti \endmulti ;
\def\cite[#1]{\citeA[#1,,\[\printsavedcites];}
\def\nocite[#1]{\citeA[#1,,,]}\def\rcite[#1]{\citeA[#1,,\[\]};\def\ecite[#1]{\bgroup\citeA[#1,,\[\tailedcite\]};\def\savedcites{}% % this list is set by \ecite inside group and it is used by \printsavedcites\bibnum\bibmark ;
\public\cite\nocite\rcite\ecite ;
\public\cite\nocite\rcite\ecite ;
\bib-marks may be numbers or a special text related to cited bib-entry. It depends on \nonmcitations and on used \bib-style. The mapping from \label to \bib-mark is done when \bib or \usebib is processed. These macros store the information to \Xbib{(label)}{\{number\}}{\{nonumber\}} where \{number\} and \{nonumber\} are two variants of \bib-mark \{numbered or text-like\}. This information is read from .ref file and is saved to macros \bib{\{label\}} and \bibm{\{number\}}. First one includes number and second one includes \{nonumber\}. The \lastbibnum macro includes last number of bib-entry used in the document. A designer can use it to set appropriate indentation when printing the list of all bib-entries.

\citeA \label, processes one label from list of labels given in the parameter of \cite, \nocite, \rcite or \ecite macros. It adds the \{label\} to global list \citelist which will be used by \usebib (it must to know what \{labels\} are used in the document in order to pick-up only relevant bib-entries from the database. Because we want to save space and not to save the same \{label\} to \citelist twice, we distinguish four cases:

- \{label\} was not declared by \Xbib and it is first such \{label\} in the document: Then \bib{\{label\}} is undefined and we save label using \addcitelist, write warning on the terminal and define \bib{\{label\}} as empty.
- \{label\} was not declared by \Xbib but it was used previously in the document: Then \bib{\{label\}} is empty and we do nothing (only data to \savedcites are saved).
- \{label\} was declared by \Xbib and it is first such \{label\} in the document: Then \bib{\{label\}} includes \bibm{\{number\}}\& and we test this case by \if \& \bibm{\{number\}}\&. This is true when \bibm{\{number\}} expands to empty. The \{label\} is saved by \addcitelist and \bib{\{label\}} is re-defined directly as \{number\}.
- \{label\} was declared by \Xbib and it was used previously in the document. Then we do nothing (only data to \savedcites are saved).

The \citeA macro runs repeatedly over whole list of \{labels\}.

The \bib-marks \{in numeric or text form\} are saved in \savedcites macro separated by commas. The \printsavedcites prints them by normal order or sorted if \sortcitations is specified or condensed if \shorcitations 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 \sortcitations sets simply \lastcitenum=1. The macros for \bib-marks printing follows (sorry, without detail documentation). They are documented in opmac-d.pdf (but only in Czech).
The \bib{⟨label⟩} \{⟨optional bib-mark⟩\} prints one bib-entry without reading any database. The bib-entry follows after this command. This command counts the used \bibs from one by \bibnum counter and saves \_Xbib{⟨label⟩}{⟨bib-mark⟩} into .ref file immediately using \_wbib. This is the core of creation of mapping from ⟨labels⟩ to ⟨bib-marks⟩.
The \_printbib prints the bib-entry itself. You can re-define it if you want different design. The \_printbib starts in horizontal mode after noindent and after the eventual hyperlink destination is inserted. By default, the \_printbib sets the indentation by \hangindent and prints numeric \bib-marks by llap{[\the\bibnum]} If nonumcitations then the \_citelinkA is not empty and \bib-marks (\the\bibnum nor \the\bibmark) are not printed. The text of bib-entry follows. User can create this text manually using \bib command or it is generated automatically from a .bib database by \usebib command.

The vertical space between bib-entries is controlled by \bibskip macro.

The \usebib command is implemented in usebib.om file which is loaded when the \usebib command is firstly used. The usebib.om file loads the librarian.tex for scaning the .bib files. See the section 2.31.2, where the file usebib.om is documented.

The macros above works if all \cite (or similar) commands are used before the \usebib command is used because \usebib prints only such bib-entries their \labels are saved in the \citelist. But if some \cite is used after \usebib, then \usebib sets \addcitelist to \_writeXcite, so such \cite saves the information to the .ref file in the format \_Xcite{\label}. Such information are copied to \citelistB during reading .ref file and \usebib concats two lists of \labels from \citelist and \citelistB and uses this concatenated list.

2.31.2 The \usebib command

The file usebib.om implements the command \usebib\sorttype \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). \bibfiles is the part of the name bib-\sorttype. om of the style file and \bibfiles are one or more .bib file names without suffix separated by comma without space. Example:

\usebib/s (simple) mybase,yourbase

This command reads the \bibfiles directly and creates the list of bibliographics references (only those declared by \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 then only the first wins. So, you can set an exceptions in your local.bib file for your document.

Notes for style writers

The bib-\sorttype. om file must define the commands:

- \_authorname ... formatting of one name in the authors list. The macro can use the following data: \NameCount (the number of the currently processed author name in the list), \_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.
- \prints\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\} \{\ifnot\defined\}. The part \if\defined is executed if \fieldname is declared in .bib file for the entry which is currently processed. Else the part
\texttt{\{if not defined\}} is processed. The part \texttt{\{if defined\}} can include the \texttt{*} parameter which is replaced by the value of the \texttt{\{fieldname\}}. The part \texttt{\{if not defined\}} can include the \texttt{\_bibwarning} command if the \texttt{\{fieldname\}} is mandatory.

- \texttt{\_bprintb \{\{fieldname\}\} \{\{if defined\}\} \{\{if not defined\}\}}. The same as \texttt{\_bprinta}, but the \texttt{##1} parameter is used instead \texttt{*}. Differences: \texttt{##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 \texttt{\_bprintb} commands are nested (\texttt{\_bprintb} in \texttt{\_bprintb}), then you need to write \texttt{####1} parameter for internal \texttt{\_bprintb}. But if \texttt{\_bprintb} commands are nested then the parameter is not duplicated.

- \texttt{\_pbbprintc \macro \{\{if non-empty\}\}}. The \{\{if non-empty\}\} part is executed if \texttt{\macro} is non-empty. The parameter can be used, it is replaced by the \texttt{\macro}.

- \texttt{\_bprintv \{\{field\}, \{\{field\}, \ldots\} \{\{if defined\}\} \{\{if not defined\}\}}. The part \{\{if defined\}\} is executed if \texttt{\{field\}} or \texttt{\{\{field\}\}} or ... is defined, else the second part \{\{if not defined\}\} is executed. There is one filed name or the list field names separated by commas. The parts cannot include any parameter.

There are two special fieldnames: \texttt{!author} and \texttt{!editor}. The processed list of authors or editors (by repeatedly calling \texttt{\_authorname} or \texttt{\_editorname}) are used here instead of raw data.

You can define \texttt{\_print:BEGIN} and/or \texttt{\_print:END} which is executed at the begin or end of each \texttt{\{entrytype\}}. The formatting does not solve the numbering and paragraph indentation of the entry. This is processed by \texttt{\_printbib} macro used in Op\TeX (and may be redefined by the author or document designer).

You can declare \texttt{\_bimark={something}} in the \texttt{\_print:END} macro. This bibmark is saved to the .ref file (created by Op\TeX) 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 \texttt{\usebib} command is executed before typesetting the reference list. Each definition or setting is local here.

If you are using non-standard fieldnames in .bib database and bib. style, you has to declare them in the file by \texttt{\_CreateField \{\{fieldname\}\}}.

You can declare \texttt{\_SortingOrder} in the manner documented by librarian package.

If your style adds some words or abbreviations you can make them multilingual by saying \texttt{\_mtext\{\{label\}\}} instead such word and \texttt{\_mtdef\{\{label\}\} \{\{English\}\} \{\{Czech\}\} \{\{Slovak\}\}} 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 \texttt{\_bibmark} tokens register have to be prepared in the style file (in \texttt{\_print:BEGIN}, \texttt{\_print:END}, in \texttt{\_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:

\texttt{\_CreateField \{sortedby\}}
\texttt{\_SpecialSort \{sortedby\}}

Suppose that the .bib file includes:

\begin{verbatim}
... author = "Jan Chadima",
        sortedby = "Hzzadima 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{\_authorname = "@"}, because this character is sorted before \texttt{A}.

### 2.31.3 The \texttt{\usebib} \texttt{\_codedec1} \texttt{\MakeReference} \{Reading bib databases <2020-03-13>\} % loaded on demand by \texttt{\usebib}

Loading the \texttt{\librarian.tex} macro package. See \texttt{\texdoc librarian} for more information about it.
The `usebib` command.

```latex
\def\usebib/#1 (#2) #3 {%
  \ifx\citelist\empty
    \opwarning{No cited items. \noexpand\usebib ignored}%
  \else
    \bgroup \par
      \emergencystretch=.3\hsize
      \ifx\bibpart\undefined \def\bibpart{none}\fi
      \def\optexbibstyle{#2}%
      \setctable\optexcatcodes
      \input bib-#2.opm
      \the\bibtexhook
      \let\citeI=\relax \xdef\citelist{\citelist\citelistB}%
      \global\let\addcitelist=\writeXcite
      \def\tmp##1[*]##2\relax\def\tmp{##2}\expandafter\tmp\citelist[*]\relax
      \ifx\tmp\empty\else % there was \nocite[*] used.
        \setbox0=\vbox{\hsize=\maxdimen \def\citelist{}\adef@{\readbibentry}\
          \input #3.bib
          \expandafter}
      \fi
    \endinput \fi
}\def\readbibentry#1#{\readbibentryA}
\def\readbibentryA#1\relax!.\{
  \addto\citelist{\citeI[#1]}
}\def\readbibentryB#1#2#3\relax!.\{
  \addto\citelist{\citeI[#1][#2]}
}\def\usebib{\noexpand\usebib}
```

Corrections in librarian macros.

```latex
\def\lb@checkmissingentries#1,{}{\opmacwarning{\usebib: entry [#1] isn't found in .bib file(s)}}
\def\lb@checkmissingentries#1,\{#2\}{\opmacwarning{\usebib: entry [#1] isn't found in .bib file(s)}}
\def\lb@checkmissingentries#1,\{}{\opmacwarning{\usebib: entry [#1] isn't found in .bib file(s)}}
```

132
Main action per every entry.

The \bprinta, \bprintb, \bprintc, \bprintv commands used in the style files:
usebib.opm

2.31.4 Usage of the bib-iso690 style

This is the iso690 bibliographic style used by OpTEX.

See op-example.bib for an example of the .bib input. You can try it by:

\fontfam[LMfonts]
\nocite[*]
\usebib/s (iso690) op-example
\end

Common rules in .bib files

There are entries of type @FOO{...} in the .bib file. Each entry consists of fields in the form name=n,"value", or name=n, {value}. No matter which form is used. If the value is pure numeric then you can say simply name=n,value. Warning: the comma after each field value is mandatory! If it is missing then the next field is ignored or bad interpreted.

The entry names and field names are case insensitive. If there exist a data field no mentioned here then it is simply ignored. You can use it to store more information (abstract, for example).

There are “standard fields” used in ancient bibTEX (author, title, editor, edition, etc., see http://en.wikipedia.org/wiki/BibTeX). The iso690 style introduces several “non-standard” fields: ednote, numbering, isbn, issn, doi, url, citedate, key, bibmark. They are documented here.

Moreover, there are two optional special fields:

- lang = language of the entry. The hyphenation plus autogenerated phrases and abbreviations will be typeset by this language.
- option = options by which you can control special printing of various fields.
There can be only one option field per each entry with (may be) more options separated by spaces. You can declare the global option(s) in your document applied for each entry by \biboptions={...}.

**The author field**

All names in the author list have to be separated by “ and ”. Each author can be written by various formats (the von part is typically missing):

- **Firstname(s) von Lastname**
- **von Lastname, Firstname(s)**
- **von Lastname, After, Firstname(s)**

Only the Lastname part is mandatory. Examples:

- Petr Olšák
- Olšák, Petr
- Leonardo Piero da Vinci
- da Vinci, Leonardo Piero
- 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 \_rm \_Lastname.

You can specify the option \aumax:{⟨number⟩}. The ⟨number⟩ denotes the maximum authors to be printed. The rest of authors are ignored and the et\-al. is appended to the list of printed authors. This text is printed only if the \aumax value is less than the real number of authors. If you have the same number of authors in the .bib file as you need to print but you want to append et\-al. then you can use \auetal option.

There is an \aumin:{⟨number⟩} option which denotes the definitive number of printed authors if the author list is not fully printed due to \aumax. If \aumin is unused then \aumax authors is printed in such case.

All authors are printed if \aumax:{⟨number⟩} option isn't given. There is no internal limit. But you can set the global options in your document by setting the \biboptions tokens list. For example:

\biboptions={aumax:7 aumin:1}
% if there is 8 or more authors then only first author is printed.
\entdd

**Examples:**

\begtt
author = "John Green and Bob Brown and Alice Black",
output: GREEN, John, Bob BROWN, and Alice BLACK.

author = "John Green and Bob Brown and Alice Black",
option = "aumax:1",
output: GREEN, John et al.

author = "John Green and Bob Brown and Alice Black",
option = "aumax:2",
output: GREEN, John, Bob BROWN et al.

author = "John Green and Bob Brown and Alice Black",
option = "aumax:3",
output: GREEN, John, Bob BROWN, and Alice BLACK.

author = "John Green and Bob Brown and Alice Black",
option = "auetal",
output: GREEN, John, Bob BROWN, Alice BLACK et al.

If you need to add a text before or after authors list, you can use the auprint:{⟨value⟩} option. The ⟨value⟩ will be printed instead of the authors list. The ⟨value⟩ can include \AU macro which expands to the authors list. Example:

author = "Robert Calbraith",
option = "auprint:{$AU}\space [pseudonym of J. K. Rowling]\"",
output: CALBRAITH Robert [pseudonym of J. K. Rowling].

You can use the autrim:(⟨number⟩) option. All Firstnames of all authors are trimmed (i.e. reduced to initials) iff the number of authors in the author field is greater than or equal to ⟨number⟩. There is an exception: autrim:0 means that no Firstnames are trimmed. This is default behavior. Another example: autrim:1 means that all Firstnames are trimmed.

author = "John Green and Bob Brown and Alice Black",
option = "auetal autrim:1",
output: GREEN, J., B. BROWN, A. BLACK et al.

If you need to write a team name or institution instead authors, replace all spaces by \␣ in this name. Such text is interpreted as Lastname. You can add the secondary name (interpreted as Firstname) after comma. Example:

author = "Czech Technical University in Prague, Faculty of Electrical 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 “and”, the Firstnames, Lastnames etc. are interpreted and you can use the options edmax:(⟨number⟩), edmin:(⟨number⟩), edetal, edtrim:(⟨number⟩) and edprint:{⟨value⟩} (with \ED macro). Example:

eeditor = "Jan Tomek and Petr Karas",
option = "edprint:{\ED, editors.} edtrim:1",
Output: J. TOMEK and P. KARAS, editors.

If edprint option is not set then {\ED,eds.} or {\ED,ed.} is used depending on the entry language and on the singular or plural of the editor(s).

The ednote field
The ednote field is used as the secondary authors and more editorial info. The value is read as raw data without any interpretation of Lastname, Firstname etc.

ednote = "Illustrations by Robert Agarwal, edited by Tom Nowak",
output: Illustrations by Robert AGARWAL, edited by Tom NOWAK.

The \upper command have to be used for Lastnames in ednote field.

The title field
This is the title of the work. It will be printed (in common entry types) by italics. The ISO 690 norm declares, that the title plus optional subtitle are in italics and they are separated by colon. Next, the optional secondary title have to be printed in upright font. This can be added by titlepost:{⟨value⟩}. Example:

title = "The Simple Title of The Work",
or
title = "Main Title: Subtitle",
or
title = "Main Title: Subtitle",
option = "titlepost:{Secondary title}"
The output of the last example: Main Title: Subtitle. Secondary title.

The edition field
This field is used only for second or more edition of cited work. Write only the number without the word "edition". The shortcut "ed." (or something else depending on current language) is added automatically. Examples:

```latex
\begin{verbatim}
\texttt{edition = "Second",}
\texttt{edition = "2nd",}
\texttt{edition = "2\$\{\text{rm nd}\}\$",}
\texttt{edition = "2. ",}
\end{verbatim}
```

Output of the last example: 2. ed.

```latex
\begin{verbatim}
\texttt{edition = "2."}
\texttt{lang = "cs",}
\end{verbatim}
```

Output: 2. vyd.

Note, that the example \texttt{edition = "Second"} may cause problems. If you are using language "cs" then the output is bad: Second vyd. But you can use \texttt{editionprint:\{\langle value\rangle\}} option. The the \texttt{\langle value\rangle} is printed instead of edition field and shortcut. The edition field must be set. Example:

```latex
\begin{verbatim}
\texttt{edition = "whatever",}
\texttt{option = "editionprint:\{Second full revised edition\}"},
\end{verbatim}
```


You can use \texttt{\\EDN} macro in editionprint value. This macro is expanded to the edition value. Example:

```latex
\begin{verbatim}
\texttt{edition = "Second",}
\texttt{option = "editionprint:\{\\EDN space full revised edition\}"},
or
\texttt{edition = "Second full revised edition",}
\texttt{option = "editionprint:\{\\EDN\}"},
\end{verbatim}
```

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.

```latex
\begin{verbatim}
\texttt{address = "Berlin",}
\texttt{publisher = "Springer Verlag",}
\texttt{year = 2012,}
\end{verbatim}
```


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 \texttt{noautoletters} option.

You can use "\texttt{yearprint:\{\langle value\rangle\}} " option. If it is set then the \texttt{\langle value\rangle} 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:

```latex
\begin{verbatim}
\texttt{year = 2000,}
\texttt{option = "yearprint:\{© 2000\}"},
\end{verbatim}
```


```latex
\begin{verbatim}
\texttt{year = "2012a",}
\texttt{option = "yearprint:\{2012\}"},
\end{verbatim}
```


The address, publisher and year are typically mandatory fields. If they are missing then the warning occurs. But you can set \texttt{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:

```latex
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:

```latex
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:

```latex
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:

```latex
volume = 31,
number = 3,
pages = "37--42",
```


```latex
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:

```latex
volume = 31,
number = 3,
pages = "37--42"
numbering = "Issue~\VOL/\NO, pages~\PP",
```

Output: Issue 31/3, pages 37–42

Note: The volume, numbers and pages fields are printed without numbering filed only in the @ARTICLE entry. It means, that if you need to visible them in the @INBOOK, @INPROCEEDINGS etc. entries, then you must to use numbering field.

Common notes about entries
The order of the fields in the entry is irrelevant. We use the printed order in this manual. The exclamation mark (!) denotes the mandatory field. If such field is missing then the warning occurs during processing.
If the **unpublished** option is set then the fields address, publisher, year, isbn and pages are not mandatory. If the **nowarn** option is set then no warnings about missing mandatory fields occurs.

If the field is used but not mentioned in the entry documentation below then it is silently ignored.

- **The @BOOK entry**  
  This is used for book-like entries.  
  Fields: author(!), title(!), howpublished, edition, ednote, address(!), publisher(!), year(!), citedate, series, isbn(!), doi, url, note.  
  The ednote field here means the secondary authors (illustrator, cover design etc.).

- **The @ARTICLE entry**  
  This is used for articles published in a journal.  
  Fields: author(!), title(!), journal(!), howpublished, address, publisher, month, year, [numbering or volume, number, pages(!)], citedate, issn, doi, url, note.  
  If the numbering is used then it is used instead volume, number, pages.

- **The @INBOOK entry**  
  This is used for the part of a book.  
  Fields: author(!), title(!), booktitle(!), howpublished, edition, ednote, address(!), publisher(!), year(!), numbering, citedate, series, isbn or issn, doi, url, note.  
  The author field is used for author(s) of the part, the editor field includes author(s) or editor(s) of whole document. The pages field specifies the page range of the part. The series field can include more information about the part (chapter numbers etc.).  
  The @INPROCEEDINGS and @CONFERENCE entries are equivalent to @INBOOK entry.

- **The @THESIS entry**  
  This is used for student’s thesis.  
  Fields: author(!), title(!), howpublished, address(!), school(!), month, year(!), citedate, type(!), ednote, doi, url, note.  
  The type field must include the text “Master’s Thesis” or something similar (depending on the language of the outer document).  
  There are nearly equivalent entries: @BACHELORSTHESIS, @MASTERSTHESIS and @PHDTHESIS. These entries set the type field to an appropriate value automatically. The type field is optional in such case. If it is used then it has a precedence before default setting.

- **The @MISC entry**  
  It is intended for various usage.  
  Fields: author, title, howpublished, ednote, citedate, doi, url, note.  
  You can use \AU, \ED, \EDN, \VOL, \NO, \PP, \ADDR, \PUBL, \YEAR macros in ednote field. These macros print authors list, editors list, edition, volume, number, pages, address, publisher and year field values respectively.

  The reason of this entry is to give to you the possibility to set the format of entry by your own decision. The most of data are concentrated in ednote field.

- **The @BOOKLET, @INCOLLECTION, @MANUAL, @PROCEEDINGS, @TECHREPORT, @UNPUBLISHED entries**  
  These entries are equivalent to @MICS entry because we need to save the simplicity. They are implemented only for (almost) backward compatibility with the ancient Bib\TeX. 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 orienets 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](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

139
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 (i.e. not yearprint value) is also used in the text oriented bib-marks when `\nonumcitations` is set.

If you have some problems with name sorting, you can use the hidden field `key`, which is used for sorting instead of the “Lastname Firstname(s)” of authors. If the `key` field is unset then the “Lastname Firstname(s)” is used for sorting normally. Example:

```latex
author = "Světla Čmejrková",
key = "Czzmejrkova Svetla",
```

This entry is now sorted between C and D.

The norm recommends to place the autocitations to the top of the list of references. You can do this by setting `key␣=␣"@"`, to each entry with your name because the @ character is sorted before A.

Languages

There is the language of the outer document and the languages of each entry. The ISO 690 norm recommends that the technical notes (the prefix before URL, the media type, the “and” conjunction between semifinal and final author) may be printed in the language of the outer document. The data of the entry have to be printed in the entry language (edition ed./vyd., Vol./ročník, No./č. etc.). Finally there are the phrases independent on the language (for example In:). Unfortunately, the 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:

```latex
howpublished = "\Mtext{blue-ray}"
```

Now, you can set the variants of blue-ray into your macros:

```latex
\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:

```latex
\def\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{}:

- `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
Summary of options

- **aumax:<number>** ... maximum number of printed authors
- **aumin:<number>** ... number of printed authors if aumax exceeds
- **autrim:<number>** ... full Firstnames iff number of authors are less than this
- **auprint:<value>** ... text instead authors list (\AU macro may be used)
- **edmax, edmin, edtrim** ... similar as above for editors list
- **edprint:<value>** ... text instead editors list (\ED macro may be used)
- **titlepost:<value>** ... text after title
- **yearprint:<value>** ... text instead real year (\YEAR macro may be used)
- **editionprint:<value>** ... text instead real edition (\EDN macro may be used)
- **urlalso** ... the "available also from" is used instead "available from"
- **unpublished** ... the publisher etc. fields are not mandatory
- **nowarn** ... no mandatory fields

Another options in the option field are silently ignored.

### 2.31.5 Implementation of the bib-iso690 style

```latex
\_maybetod (alias \ in the style file group) does not put second dot.
```

### Option field.

```latex
\_CreateField {option}
```

### Formating of Author/Editor lists.

```latex
\_firstauthorformat(%
```

```latex
\_otherauthorformat(%
```

```latex
\_commonname(%
```

```latex
\_firstnamecount=1
```

```latex
\_firstauthorformat
```
Preparing bib-mark (used when \nonumcitations is set).
\def\setbibmark{
  \ifx\dobibmark\undefined
    \def\dobibmark{\Lastname}\fi
  \else
    \ifnum\namecount=\NameCount
      \ifx\maybeetal\empty \bibconjunctionand \else , \fi
    \else
      \fi
    \otherauthorformat
  \fi
}
Non-standard fieldnames.

Sorting.

Supporting macros.
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 string is equal in primary pass then secondary pass is started. It distinguishes between variously accented letters. Czech rules, for
example says: not accented before dieresis before acute before circumflex before ring. At less priority: lowercase letters must be before uppercase letters.

The \_sortingdata{iso-code} implements these rules for the language (iso-code). The groups between commas are not distinguished in the first pass. The second pass distinguishes all characters mentioned in the \_sortingdata{iso-code} (commas are ignored). The order of letters in the \_sortingdata{iso-code} macro is significant for sorting algorithm. The Czech rules (cs) are implemented here:

```latex
\begin{verbatim}
def \_sortingdatacs {
% 
./, (),.-,a,2,\% 
 a\d\a\d\a\d\a,\% 
 bB,\% 
 cc,\% 
 cc,\% 
dd,\% 
eEae\eE,\% 
 fF,\% 
gG,\% 
hH,\% 
"T"V,\% ch CH 
 iII,\% 
jJ,\% 
kK,\% 
lIIIII,\% 
mM,\% 
nNhN,\% 
oo\o\o\o\o\o,\% 
pP,\% 
qQ,\% 
rř,\% 
sS,\% 
ťť,\% 
uUúÜúÚůŮ,\% 
vV,\% 
wW,\% 
xX,\% 
yYyY,\% 
zZ,\% 
\% 
0,1,2,3,4,5,6,7,8,9,\% }
\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 \texttt{65} are sorted first if they are not mentioned in the \_sortingdata{iso-code} macro. Others not mentioned characters have undefined behavior during sorting.

```latex
\begin{verbatim}
def \_ignoredcharscs {.,;?!:'"|()[\]<>=+/}
def \_compoundcharscs {ch:"T Ch:"U CH:"\% DZ etc. are sorted normally
```

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 \texttt{^^I} or \texttt{^^M} because these characters have very specific category code. And use space to separate more mappings, like in \_compoundcharscs above.

```latex
\begin{verbatim}
let \_sortingdataSk = \_sortingdata
```

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 \_\texttt{\{string\}} control sequences (in order to save \TeX\ memory). The \_\texttt{\{string\}} converts \_\texttt{\{string\}} to \_\texttt{\{tmpb\}} with respect to the data initialized in \_\texttt{\{primary\} or \_\texttt{\{secondary\}.}

The compound characters are converted to single characters by the \_\texttt{\{dcompound\} macro.

Macro \_\texttt{\{isAleB\}} \_\texttt{\{string1\}} \_\texttt{\{string2\}} returns the result of comparison of given two strings to \_\texttt{\{ifAleB\} control sequence. Usage: \_\texttt{\{isAleB \_\texttt{\{string1\} \_\texttt{\{string2\} \_\texttt{\{ifAleB \_\texttt{\{else\} \_\texttt{\{fi\}}

The converted strings (in respect of the data prepared for first pass) must be saved as values of \_\texttt{\{string1\} and \_\texttt{\{string2\} macros. The reason is speed: we don’t want to convert them repeatedly in each comparison. The macro \_\texttt{\{testAleB\} \_\texttt{\{converted string1\}} \_\texttt{\{converted-string2\}} \_\texttt{\{relax\}} \_\texttt{\{string1\}} \_\texttt{\{string2\}} does the real work. It reads first character from both converted strings, compares them and if it is equal then calls itself recursively else gives result.
Merge sort is very effectively implemented by \TeX macros. The following code is created by my son Miroslav. The \texttt{mergesort} macro expects that all items in \texttt{iilist} are separated by comma when it starts. It ends with sorted items in \texttt{iilist} without commas. So \texttt{dosorting} macro must prepare commas between items.

The \texttt{dosorting} \texttt{list} macro redefines \texttt{list} as sorted \texttt{list}. The \texttt{list} have to include control sequences in the form \texttt{\langle c \rangle \langle string \rangle}. These control sequences will be sorted in respect to \texttt{\langle strings \rangle} without change of meanings of these control sequences. Their meanings are irrelevant when sorting. The first character \texttt{\langle c \rangle} in \texttt{\langle c \rangle \langle string \rangle} should be whatever. It does not influence the sorting. Op\TeX uses comma at this place for sorting indexes: \texttt{\langle word1 \rangle , \langle word2 \rangle , \langle word3 \rangle \ldots}.

The actual language (chosen for hyphenation patterns) is used for sorting data. If the \texttt{sortinglang} macro is defined as \texttt{\langle iso-code \rangle} (for example \texttt{\def \sortinglang {de}}) then this has precedence and actual language is not used. Moreover, if you specify \texttt{asisortingtrue} then ASCII sorting will be processed and all language sorting data will be ignored.
The \makeindex prints the index. First, it sorts the \_iilist, second, it prints the sorted \_iilist, each item is printed using \_printindexitem.

The \_printindexitem \langle word \rangle prints one item to the index. If \langle word \rangle is defined then this is used instead real \langle word \rangle (this exception is declared by \iis macro). Else \langle word \rangle is printed by \_printii. Finally, \_printiipages prints the value of \langle word \rangle, i.e. the list of pages.

\_printii \langle word \rangle & does more intelligent work because we are working with words in the form \langle main-word \rangle/\langle sub-word \rangle/\langle sub-sub-word \rangle. The \_everyii tokens register is applied before \_noindent. User can declare something special here.

The \newiiletter{\langle letter \rangle} macro is empty by default. It is invoked if first letter of index entries is changed. You can declare a design between index entries here. You can try, for example:

\def\newiiletter#1#2{% 
  \bigskip \hbox{\setfontsize{at15pt}\bf\uppercase{#1}}\medskip
}
\_printiipages (pglist) \& gets (pglist) in the form \{pg\} \{(type)\}, \{pg\} \{(type)\}, \ldots \{pg\} \{(type)\} and it converts them to \{pg\}, \{pg\}, \{from\} \{to\}, \{pg\} \ldots The same pages must be printed only once and continuous consequences of pages must be compressed to the form \{from\} \{to\}. Moreover, the consequence is continuous only if all pages have the same \{(type)\}. Empty \{(type)\} is most common, pages with \b{(type)\} must be printed as bold and with \i{(type)\} as italics. Moreover, the \{pg\} mentioned here are \{pggeno\}, but we have to print \{pageno\}. The following macros solves these tasks.

You can re-define \_pgprint \{pggeno\}:{\{iitype\}} if you need to implement more \{iitype\}.

The \index{\{word\}} puts one \{word\} to the index. It writes \_Index{\{word\}}{\{iitype\}} to the .ref file. All other variants of indexing macros expands internally to \_iindex. The \_Index{\{word\}}{\{iitype\}} stores \{word\} to the \_iilist if there is first occurrence of the \{word\}. The list of pages where \{word\} occurs, is the value of the macro \_\{word\}, so the \{pggeno\}:{\{iitype\}} is appended to this list. Moreover, we need a mapping from \{pggeno\} to \{pageno\}, because we print \{pageno\} in the index, but hyperlinks are implemented by \{pggeno\}. So, the macro \_pgi{\{pggeno\}} is defined as \{pageno\}. The following macros solves these tasks.
The implementation of macros \ii, \iid, \iis follows. Note that \ii works in horizontal mode on order to the \write whatsis is not broken from the following word. If you need to keep vertical mode, use \index{⟨word⟩} directly.

The \iitype{⟨type⟩} saves the ⟨type⟩ to the \iitypesaved macro. It is used in the \iiindex macro.

\public \ii \iid \iis \iitype

2.33 Footnotes and marginal notes

\_printfnotemark prints the footnote mark. You can re-define this macro if you want another design of footnotes. For example

\fnotenumpages
\def \_printfnotemark {
\ifcase 0\fnotenum\or *\or**\or***\or$\textsuperscript{†}$\or$\textsuperscript{‡}$\or$\textsuperscript{††}$\fi}
This code gives footnotes* and ** and*** and† etc. and it supposes that there are no more than 6 footnotes at one page.

If you want to distinguish between footnote marks in the text and in the front of footnote itself, then you can define \printfnotemarkA and \printfnotemarkB.

The \fnotelinks\langle colorA\rangle\langle colorB\rangle implements the hyperlinked footnotes (from text to footnote and backward).

Each footnote saves the \_Xfnote (without parameter) to the .ref file (if \openref). We can create the mapping from \_gfnotenum to \_pgfnotenum in the macro \_fn:\_fnotenum. Each \_Xpage macro sets the \_lfnotenum to zero.

The \fnote {\langle text\rangle} macro is simple, \fnotemark and \fnotetext does the real work.

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 outer box of marginal note has zero width and zero depth and it is appended after current line using \vadjust primitive or it is inverted to vertical mode as a box with \vskip-\baselineskip followed.

The \mnoteA macro does the real work. The \_lrmnote\{⟨left⟩\}⟨⟨right⟩\} uses only first or only second parameter depending on the left or right marginal note.

We don’t want to process \fnote, \fnotemark, \mnote in TOC, headlines nor outlines.

OpTeX provides three styles: \report, \letter and \slides. Their behavior is documented in user part of the manual in the section 1.7.2 and \slides style (for presentations) is documented in op-slides.pdf which is an example of the presentation.

2.34 Styles

OpTeX provides three styles: \report, \letter and \slides. Their behavior is documented in user part of the manual in the section 1.7.2 and \slides style (for presentations) is documented in op-slides.pdf which is an example of the presentation.

2.34.1 \report and \letter styles

We define auxiliary macro first (used by the \address macro)

The \boxlines returns the outer vertical mode a box with \hsize, next box with \hsize etc. Each box has its natural width. This reason why we cannot use paragraph mode where each resulting box has the width \hsize. The \col is set active and \everypar starts \hbox and acive \col closes this \hbox by \.

The \report and \letter style initialization macros are defined here. The \letter defines \address and \subject macros.
The \texttt{\textbackslash slides} macro reads macro file \texttt{slides.opm}, see the section \texttt{\textbackslash slidesopm \texttt{\textbackslash public \textbackslash \textbackslash \textbackslash slides ;}} 

\subsection{\texttt{\textbackslash slides} style for presentations}

\input{\texttt{\textbackslash slidesopm}}

The bottom margin is set to 3mm. If we use 1mm, then baseline of \texttt{\textbackslash footline} is 2mm from the bottom page. This is depth of the \texttt{\textbackslash Grey} rectangle used for page numbers. It is r-lapped to \texttt{\textbackslash hoffset} width because left margin = \texttt{\textbackslash hoffset} = right margin. It is 14mm for narrow pages or 16mm for wide pages. 

The \texttt{\textbackslash subtit} is defined analogically like \texttt{\textbackslash tit}. 

154
The \texttt{\pshow\(\text{num}\)} prints the text in invisible (transparent) font when \texttt{\layernum<\text{num}}. The transparency is set by the \texttt{pdfpagersrocyres} primitive.

\begin{verbatim}
pdfpagersrocyres{/ExtGState << /Invisible << /Type /ExtGState /ca 0 /CA 0 >> /Visible << /Type /ExtGState /ca 1 /CA 1 >> >>}
\addto\morepgresources{/Invisible << /Type /ExtGState /ca 0 /CA 0 >> /Visible << /Type /ExtGState /ca 1 /CA 1 >>}
\def\Invisible {\pdfliteral{/Invisible gs}}
\def\Visible {\pdfliteral{/Visible gs}}
\def\Transparent {\Invisible \aftergroup \Visible}
\def\_use#1#2{\ifnum\layernum#1\relax#2\fi}
\def\_pshow#1{\_use{=#1}\Red \_use{<#1}\Transparent \_ignorespaces}
\end{verbatim}

The main level list of items is activated here. The \texttt{\_item:X} and \texttt{\_item:x} are used and are re-defined here. If we are in nested level of items and \texttt{\pg+} is used then \texttt{egroups} macro expands to the right number of \texttt{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 \texttt{\pg+}.

\begin{verbatim}
\newcount\gilevel
\def\*{*}
\edef*{\startitem}
\sdef{\_item:X}{\Blue\raise.2ex\fullrectangle{.8ex}\kern.5em}
\sdef{\_item:x}{\Blue\raise.3ex\fullrectangle{.6ex}\kern.4em}
\_style X
\def\egroups{\par\global\gilevel=\ilevel \egroup}
\everylist={\novspaces \ifcase\ilevel \or \_style x \else \_style - \fi \addto\egroups{\egroup}}
\end{verbatim}

The default values of \texttt{\pg}, i.e. \texttt{\pg;}, \texttt{\pg+} and \texttt{\pg.} are very simple. They are used when \texttt{\showslides} is not specified.

\begin{verbatim}
\def\_pg#1{\cs{\_spg:#1}}
\edef{\_spg:;}{\vfil\break \lfnotenumreset}
\edef{\_spg:.}{\end}
\edef{\_spg:+}{\par}
\end{verbatim}

We need no numbers and no table of contents when using slides. The \texttt{\_printsec} macro is redefined in order the title is centered and typeset in \texttt{\Blue}.

\begin{verbatim}
\def\titfont{\\typo\size[42/60]\bf \Blue}
\def\subtitfont{\\typo\size[20/30]\bf}
\def\secfont{\\typo\size[26/30]\bf \Blue}
\def\_nonum \_notoc \_let\resetnotumnotoc=\relax
\def\printsec\#1{\par
  \_abovetitle{\_penalty-400}\_bigskip
  \_secfont \noindent \_leftskip=0pt plusfill \_rightskip=\_leftskip
  \_printrefnum[@\_quad\#1\nbpar]\insertmark[\#1]%
  \_nobreak \_belovetitle{\_medskip}%
}
\end{verbatim}

When \texttt{\slideshow} is active then each page is opened by \texttt{\setbox\slidepage=vbox\bgroup} (roughly speaking) and closed by \texttt{\egroup}. The material is \texttt{\unvbox}ed and saved for the usage in the next usage if \texttt{\pg+} is in process. The \texttt{\slidelayer} is incremented instead \texttt{\pageno} if \texttt{\pg+}. This counter is equal to \texttt{\count1}, so it is printed to the terminal and log file next to \texttt{\pageno}.

The code is somewhat more complicated when \texttt{\layers} is used. Then \texttt{\layered-text} is saved to the \texttt{\_layertext} macro, the material before it is in \texttt{\_slidepage} box and the material after it is in \texttt{\_slidepageB} box. The pages are completed in the \texttt{\loop} which increments the \texttt{\layernum} register.
\def\slideshowactive{%
  \sdef{\spg:;}\{\closepage \global\slidlayer=1 \resetpage \openslide\}
  \sdef{\spg:+}\{\closepage \incr\slidlayer \decr\pageno \openslide\}
  \sdef\bye \{\closepage \end\}
  \let\layers=\layersactive
  \def\destbox[#1=#2]{\iffalse \destboxori[#1=#2]\fi}%
}\def\openslide{%
  \ifvoid\slidepage \else \unvbox\slidepage \nointerlineskip \lastbox \fi}
\def\setilevel{\loop \decr\gilevel \ifnum\gilevel<0 \else \begitems \repeat}
\def\closepage{\egroups
  \ifnum\maxlayers=0 \unvcopy\slidepage \vfil \break
  \else \begingroup \setwarnslides \layernum=0
  \loop
    \ifnum\layernum<\maxlayers \advance\layernum by1
      \printlayers \vfil \break
    \else \advance\layernum \fi
  \repeat
  \global\maxlayers=0
  \incr\layernum \global\setbox\slidepageB=\vbox{\printlayers}\
  \endgroup
}\def\resetpage{%
  \global\setbox\slidepage=\box\voidbox
  \global\setbox\slidepageB=\box\voidbox
  \lfnotenumreset\let\destboxori=\destbox
  \newcount\layernum \newcount\maxlayers
  \maxlayers=0
  \loop\long\def\layersactive #1 \#2\endlayers{%
    \par\egroup
    \edef\layertext#2\global\maxlayers=#1
    \global\setbox\slidepageB=\vbox{\printlayers}\
  }\public \subtit \slideshow \pg \wideformat \use \pshow ;
\def\layers{\par\layernum=\numexpr\layernum+1\relax
  \global\maxlayers=0
  \long\def\layersactive #1 \#2\endlayers{%
    \par\egroup
    \edef\layertext#2\global\maxlayers=#1
    \global\setbox\slidepageB=\vbox{\printlayers}\
  }
\def\fnotenumpages{%\def\fnotenum{\the\lfnotenum}\pgfnotefalse
  \def\lfnotenumreset{\global\lfnotenum=0 }}\let\lfnotenumreset=\relax
\public \subtit \slideshow \pg \wideformat \use \pshow ;

Default $\texttt{\layers (num)}$ macro (when $\texttt{\slideshow}$ is not activated) is simple. It prints the $\texttt{\layertext}$ with $\texttt{\layernum=(num)+1}$ because we need the result after last layer is processed.

We must to redefine $\texttt{\fnotenumpages}$ because the data from .ref file are less usable for implementing such feature: the footnote should be in more layers repeatedly. But we can suppose that each page starts by $\texttt{\pg;}$ macro, so we can reset the footnote counter by this macro.
2.35 Logos

Despite plain \TeX each macro for logos ends by \ignoreslash. This macro ignores next slash if it is present. You can use \TeX/ like this for protecting the space following the logo. This is visually more comfortable. The macros \TeX, \OpTeX, \LuaTeX, \XeTeX are defined.

The \_slantcorr macro expands to slant-correction of current font. It is used to shifting A if the \LaTeX logo is in italic.

The expandable versions of logos used in Outlines needs the expandable \ignslash (instead of the \ignoreslash).

2.36 Multilingual support

2.36.1 Lovercase, uppercase codes

All codes in unicode table keep information about pairs lowecase-uppercase letters or single letter. We need to read such information and set appropriate \lccode and \uccode. The \catcode above the code 127 is not set, i.e. the \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.
\texttt{_wterm{Setting lccodes and uccodes for Unicode characters}}

\begin{verbatim}
\_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 \texttt{uni-lcuc.opm})
\end{verbatim}

\section*{2.36.2 Hyphenations}

\texttt{\_codedecl \langlist (Initialization of hyphenation patterns <2020-03-10>) \% preloaded in format}

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{\_lan:\{number\}} expands to \texttt{⟨iso-code⟩} of the language. The number is internal number of languages used as a value of \texttt{language} register.
- \texttt{\_ulan:\{long-lang\}} expands to \texttt{⟨iso-code⟩} too. This is transformation from long name of language (lowercase letters) to \texttt{⟨iso-code⟩}.
- \texttt{\langle iso-code\rangle Patt} (for example \texttt{\_csPatt}) is the language \texttt{⟨number⟩} declared by \texttt{\chardef}.  
- \texttt{\langle iso-code\rangle lang} (for example \texttt{\enlang, \cslang, \sklang, \delang, \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 patters are preloaded first:

\begin{verbatim}
\_chardef\_enPatt=0
\_def\_pattlist{\_enPatt=0}
\_def\_langlist{en(USenglish)}
\_sdef{\_lan:0}{en}
\_sdef{\ulan:usenglish}{en}
\_def\_enlang{\useenglish}{en}
\_def\_enlang{\enPatt23} % \lefthyphen=2 \righthyphen=3
\_def\_enlang{\enang}
\_def\_langspecific:en{\\_nonfrenchspacing}
\_lefthyphenmin=2 \_righthyphenmin=3 \% disallow x- or -xx breaks
\_input hyphen \% en(USenglish) patterns from TeX82
\end{verbatim}

\texttt{\preplang \langle iso-code\rangle \langle long-lang\rangle \langle number\rangle \langle number\rangle \langle pre-hyph\rangle\langle post-hyph\rangle} prepares the \texttt{\langle iso-code\rangle lang} to its initialization state. Roughly speaking, it does:
\chardef\_<iso-code>Patt = ⟨number⟩
\def\_lan:{⟨number⟩} {⟨iso-code⟩}
\def\_ulan:{⟨long-lang⟩} {⟨iso-code⟩}
\def\_<iso-code>lang {%
  \loadpatts ⟨long-lang⟩ ⟨number⟩ % loads patterns using Lua code
  \gdef\_<iso-code>lang {\_uselang{⟨iso-code⟩} \_<iso-code>Patt ⟨pre-hyph⟩ ⟨post-hyph⟩}
  ⟨iso-code⟩ lang % runs itself in processing state
}%
\def\_<iso-code>lang {\_<iso-code>lang} % public version \_<iso-code>lang

You can see that \_<iso-code>lang runs \loadpatts ⟨long-lang⟩ ⟨iso-code⟩ in initialization state and \_uselang in processing state.

\_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).

\_def\_uselang#1#2#3#4\%{
  \_language=#2\_lefthyphenmin=#3\_righthyphenmin=#4\relax\frenchspacing % \nonfrenchspacing can be set in \cs{\_langspecific:lan}
  \cs{\_langspecific:#1} %
}\%

The \_langspecific \{⟨long-lang⟩\} is defined here (for compatibility with e-plain users).

\_def\_uselanguage#1\%{\_lowercase{\cs{\_uselang:#1}}} \public \_uselanguage ;

The numbers for languages are declared as fixed constants (no auto-generated). This concept is inspired from CSplain. There are typical numbers of languages in CSplain: 5=Czech in IL2, 15=Czech in T1 and 115=Czech in Unicode. We keep these constants but we load only Unicode patterns (greater than 100), of course.
\_preplang fi Finnish \_fiPatt 129 22
\_preplang hy Hungarian \_huPatt 130 22
\_preplang tr Turkish \_trPatt 131 22
\_preplang en Estonian \_etPatt 132 23
\_preplang eu Basque \_euPatt 133 22
\_preplang ga Irish \_gaPatt 134 23
\_preplang nb Bokmal \_nbPatt 135 22
\_preplang nn Nynorsk \_nnPatt 136 22
\_preplang nl Dutch \_nlPatt 137 22
\_preplang pt Portuguese \_ptPatt 138 23
\_preplang ro Romanian \_roPatt 139 22
\_preplang hr Croatian \_hrPatt 140 22
\_preplang zh Pinyin \_zhPatt 141 11
\_preplang is Icelandic \_isPatt 142 22
\_preplang hsb Uppersorbian \_hsbPatt 143 22
\_preplang af Afrikaans \_afPatt 144 12
\_preplang gl Galician \_glPatt 145 22
\_preplang kmr Kurmanji \_kmrPatt 146 22
\_preplang tk Turkmen \_tkPatt 147 22
\_preplang la Latin \_laPatt 148 22
\_preplang lac classicLatin \_lacPatt 149 22
\_preplang lal liturgicalLatin \_lalPatt 150 22
\_preplang elm monoGreek \_elmPatt 201 11
\_preplang elp Greek \_elpPatt 202 11
\_preplang grc ancientGreek \_grcPatt 203 11
\_preplang ca Catalan \_caPatt 204 22
\_preplang cop Coptic \_copPatt 205 11
\_preplang mm Mongolian \_mnPatt 206 22
\_preplang sa Sanskrit \_saPatt 207 13
\_preplang ru Russian \_ruPatt 208 22
\_preplang uk Ukrainian \_ukPatt 209 22
\_preplang hy Armenian \_hyPatt 210 12
\_preplang as Assamese \_asPatt 211 11
\_preplang hi Hindi \_hiPatt 212 11
\_preplang kn Kannada \_knPatt 213 11
\_preplang lv Latvian \_lvPatt 215 22
\_preplang lt Lithuanian \_ltPatt 216 22
\_preplang ml Malayalam \_mlPatt 217 11
\_preplang mr Marathi \_mrPatt 218 11
\_preplang or Oriya \_orPatt 219 11
\_preplang pa Panjabi \_paPatt 220 11
\_preplang ta Tamil \_taPatt 221 11
\_preplang te Telugu \_tePatt 222 11

The \langlist includes names of all languages which are ready to load and use their hyphenation patterns. This list is printed to terminal and to log at ini\TeX state here. It can be used when processing document too.

\message{Language hyph.\patterns ready to load: \langlist.}
\string\_langlist\string\_space for example}

\public \langlist ;

Maybe, you need to do more language specific actions than only to switch hyphenation patterns. For example you need to load a specific font with a specific script used in selected language, you can define a macros for quotation marks depending on the language etc.

The example shows how to declare such language specific things.

\def\langset #1 #2{\sdef{{\langspecific:#1}{#2}}}

\langset fr {\_langset\string\_space for example}
\langset de {\_langset\string\_space for example}
\langset gr {\_langset\string\_space for example}

\langset #1 #2{\sdef{{\langspecific:#1}{#2}}}

\langset fr {\_langset\string\_space for example}
\langset de {\_langset\string\_space for example}
\langset gr {\_langset\string\_space for example}

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 \texttt{language} register, if you use \texttt{\_mtext{⟨phrase-id⟩}}, specially \texttt{\_mtext{chap}}, \texttt{\_mtext{t}}, \texttt{\_mtext{f}} or \texttt{\_mtext{subj}}. If your macros generate more words then you can define such words by \texttt{\sdef{\_mt:⟨phrase-id⟩:⟨lang⟩}} where \texttt{⟨phrase-id⟩} is a label for declared word and \texttt{⟨lang⟩} is language shortcut (iso code).

Using \texttt{\langw ⟨lang⟩ ⟨chapter⟩ ⟨table⟩ ⟨figure⟩ ⟨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 Subjekt
\langw es Capítulo Tabla Figura Sujeto
\langw fr Chaptire Tableau Figure Matière
\langw it Capitolo Tabella Fig. Soggetto
\langw pl Rozdział Tabela Ilustracja Temat
\end{verbatim}

You can add more words as you wish. For example \texttt{\today} macro:

\begin{verbatim}
\def\today{}\public \today;
\end{verbatim}

Quotes should be tagged by \texttt{"⟨text⟩"} and \texttt{‘⟨text⟩’} if \texttt{⟨iso-code⟩} quotes is declared at beginning of the document (for example \texttt{\enquotes}). If not, then the control sequences \texttt{"} and \texttt{‘} are undefined. Remember, that they are used in another meaning when \texttt{\oldaccents} command is used. The macros \texttt{"} and \texttt{‘} are not defined as \texttt{\protected} because we need their expansion when \texttt{\outlines} are created. User can declare quotes by \texttt{\quoteschars⟨clqq⟩⟨crqq⟩⟨clq⟩⟨crq⟩}, where \texttt{⟨clqq⟩}...\texttt{⟨crqq⟩} are normal quotes and \texttt{⟨clq⟩}...\texttt{⟨crq⟩} are alternative quotes. or use \texttt{\altquotes} to swap between meaning of these two types.
of quotes. \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
\end{verbatim}

Sometimes should be usable to leave the markup "such" or 'such' i.e. without the first backslash. Then you can make the characters " and ' active by the \activequotes macro and leave quotes without first backslash. First, declare ⟨iso-code⟩quotes, then \altquotes (if needed) and finally \activequotes.

\begin{verbatim}
\_def \_activequotes {\_ea \_activequotesA"""" \_ea \_activequotesA'}
\_def \_activequotesA #1 #2 #3 {\_bgroup \_lccode `~={#3} \lowercase {\_egroup \_adef #3 ##1 ~={#1 ##1 #2}}}
\end{verbatim}

\_public \quoteschars \activequotes \enquotes \csquotes \skquotes \frquotes \plquotes \esquotes \grquotes \ruquotes \itquotes \dequotes ;

2.37 Other macros

Miscellaneous macros are here.

\begin{verbatim}
\_codedecl \uv {Miscenaleous <2020-04-02>} % preloaded in format
\useOpTeX and \useoptex are declared as \relax.
\_let \useOpTeX = \_relax \_let \useoptex = \_relax
\end{verbatim}

The \lastpage and \totalpages get the information from the \_currpage. The \_Xpage from .ref file sets the \_currpage.

\begin{verbatim}
\_def \_totalpages {\openref \ea \_lastpageA \_currpage}
\_def \_lastpage {\openref \ea \_lastpageB \_currpage}
\_def \_lastpageA #1 #2 {#1}
\_def \_lastpageB #1 #2 {#2}
\_def \_currpage {{0} {?}}
\_public \lastpage \totalpages ;
\end{verbatim}

We need \uv, \clqq, \crqq, \flqq, \frqq, \uslang, \ehyph \chyph, \shyph, for backward compatibility with \csplain. Codes are set according to Unicode, because we are using Czech only in Unicode when Lua\TeX is used.

\begin{verbatim}
\_chardef \clqq=8222 \_chardef \crqq=8220
\_chardef \flqq=171 \_chardef \frqq=187
\_chardef \promile=8240
\_let \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
\end{verbatim}

The \letfont was used in \csplain instead of \fontlet.

\begin{verbatim}
\_let \_letfont = \_fontlet
\end{verbatim}

Non breaking space in Unicode.
2.38 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 \texttt{filename} from the second line. The file is read in the listing mode. The \printdoctail starts reading given \texttt{filename} from the first occurrence of the \_encode. The file is read in normal mode (like \input \{filename\}).

The listing mode prints the lines as listing of a code. This mode is finished when first \_doc occurs or first \_endcode occurs. At least two spaces must precede before such \_doc. On the other hand, the \_encode must be at the left edge of the line without spaces. If this rule is not met then the listing mode continues.

If the first line or the last line of the listing mode is empty then such lines are not printed. The maximal number of printed lines in the listing mode is \texttt{maxlines}. Is is set to almost infinity (100000). You can set it to a more sensible value. Such setting is valid only for the first following listing mode.

When the listing mode is finished by \_doc then next lines are read in the normal way, but the material between \begtt ... \endtt pair is shifted by three letters left. The reason is that the three spaces of indentation is recommended in the \_doc ... \_cod pair and this shifting is a compensation of this indentation.

The \_cod macro ignores the rest of current line and starts the listing mode again.

When the listing mode is finished by the \_endcode then the \endinput is applied, the reading of the file opened by \printdoc is finished.

You cannot reach the end of the file (without \_endcode) in the listing mode.

The listing mode creates all control sequences which are listed in the index as active link to the main documentation point of such control sequence and prints them in blue. Other text is printed in black.

The main documentation point is denoted by \texttt{\~\langle sequence\rangle} in red, for example \texttt{\~\foo}. The user documentation point is the first occurrence of \texttt{\~\langle sequence\rangle}, for example \texttt{\~\foo}. There can be more such markups, all of them are hyperlinks to the main documentation point. And main documentation point is hyperlink to the user documentation point, if such point exists. Finally, the \texttt{\~\langle sequence\rangle} (for example \texttt{\~\foo}) are hyperlinks to the user documentation point.

General declarations.
Maybe, somebody needs \seccc or \secccc?

\enddocument can be redefined.

Full page of listing causes underfill \vbox in output routine. We need to add a small tolerance.

The listing mode is implemented here.
The lines in the listing mode have Yellow background.

\docfile is currently documented file. \printdoc and \printdoctail macros are defined here.

The Index entries are without the trailing backslash. We must to add it when printing Index.

You can do \verbinuput \vitt{(filename)} ((from)-(to)) (filename) if you need analogical design like in listing mode.
If this macro is loaded by \load then we need to initialize catcodes using the \_afterload macro.

Main documentation point and hyperlinks to/from it.