The minitoc package*

Jean-Pierre F. Drucbert
jean-pierre.drucbert@onera.fr

16th September 2005

*This document corresponds to minitoc v43, dated 2005/09/16.

Summary

Contents

List of Figures

List of Tables

FIX INDEX? FIX INDEX?

Part I

User’s Manual

Table of Contents

Chapter 1

The `minitoc` package

Contents

Figures

Tables

1.1 Introduction

The minitoc package, initially written by Nigel Ward and Dan Jurafsky, has been almost completely redesigned by Jean-Pierre F. Drucbert (ONERA/Centre de Toulouse). This package creates a mini-table of contents (a “minitoc”[1]) at the beginning of each chapter of a document. It is also possible to have a mini-list of figures (a “minilof”) and a mini-list of tables (a “minilot”). The document class should of course define chapters (classes like book or report) or sections (classes like article[2]). Thus, this package should not be used with document classes without standard sectionning commands (like letter). When the document class defines a “part” sectionning level (*i.e.* classes like book, report and article), you can create a “partial” table of contents (a “parttoc”) at the beginning of each part of a document. It is also possible to have a partial list of figures (a “partlof”) and a partial list of tables (a “partlot”). When the document class has no \chapter command but has a \section command, you may use section level tables of contents (“secttocs”) at the beginning of each section; and you can also have section level lists of figures (“sectlofs”) or of tables (“sectlots”).

[1] The minitoc package introduces its own jargon, explained in this document. It should not be too difficult, however, to learn and use; it will be used here, of course.

[2] As the standard proc class loads the standard article class, this class will be just considered as a variant of the article class.

All these tables ("minitocs", "partlots", "sectlofs", etc.) are collectively referenced as "mini-tables" (or sometimes "mini-lists").

Note: you cannot use chapter level and section level mini-tables in the same document. This restriction is intented to avoid documents with full of local tables of contents, lists of figures and tables at every sectionning level.

Note: the commands relative to the part level are defined only if the document class defines `\part`. The commands relative to the section level are defined only if the document class defines `\section` but does not define `\chapter`.

The current version of this package is #43.

1.1.1 License

This package must be distributed and/or may be modified under the conditions of the **LaTeX Project Public License**, either version 1.3 of this license or (as convenient) any later version. The latest version of this license is in

    http://www.latex-project.org/lppl.txt

and version 1.3 or later is part of all distributions of LaTeX version 2003/12/01 or later.

But please don't bother me about hacked versions; they will not be supported.

1.2 How to use the `minitoc` package

1.2.1 Loading the package and creating the mini-tables

`\usepackage`
`\minitoc`

To use the `minitoc` package, you must introduce a command:

```
\usepackage[...options...]{minitoc}
```

in the preamble of the document[3]. The mini-table of contents will be in the chapter, after the `\chapter` command, at the point of the `\minitoc` command. The `\minitoc` command may occur *almost anywhere*[4] inside a chapter. Of course, it is better to put

[3]This command must be placed *after* any modification done on the sectionning commands; if you modify sectionning commands after loading the `minitoc` package, this one might not work properly.

[4]"Almost anywhere" means "in a normal place", like between two paragraphs of normal text, or in a (wide enough) minipage, but not in a too strange position (like a marginal note or a footnote). Even a multicolumn or a floating environment can be used, but with care. But note that a minitoc can be rather long, if the chapter is complex and if you ask details with a high value for `minitocdepth`.

Table 1.1: Commands for a minitoc

```
\documentclass[...]{book}
\usepackage[...options...]{minitoc}
...
\setlength{\mtcindent}{24pt}                 default
\setlength{\mtcskipamount}{\bigskipamount}   default
...
\setcounter{minitocdepth}{2}                 default
\renewcommand{\mtcfont}{\small\rmfamily\upshape\mdseries}          default
\renewcommand{\mtcSfont}{\small\rmfamily\upshape\bfseries}         default
or:
\mtcsetdepth{minitoc}{2}                                           default
\mtcsetfont{minitoc}{*}{\small\rmfamily\upshape\mdseries}          default
\mtcsetfont{minitoc}{section}{\small\rmfamily\upshape\bfseries}    default
...
\begin{document}
...
\dominitoc
\dominilof
\dominilot
\tableofcontents                             or \faketableofcontents
\listoffigures                               or \fakelistoffigures
\listoftables                                or \fakelistoftables
...
\chapter{...}
\minitoc                                     if you want one
\mtcskip
\minilof                                     if you want one
\mtcskip
\minilot                                     if you want one
...
```

it at the beginning of the chapter, eventually after some introductory material. But you can also decide to put it at the end of the chapter. You should use the same conventions in all chapters. If you want to add the mini-table of contents for a chapter, you must use the sequence given in table 1.1. For each mini-table of contents, an auxiliary file will be created with a name of the form ⟨*document*⟩`.mtc`⟨*N*⟩, where ⟨*N*⟩ is the absolute chapter number. "Absolute" means that this number is unique, and always increasing from the first chapter[5]. The suffix is `.mlf`⟨*N*⟩ for mini-lists of figures and is `.mlt`⟨*N*⟩ for mini-lists of tables. (If under MS-DOS or any operating system with short extensions to filenames, see section 1.6 on page 40 and section 2.5 on page 44). There are similar commands for mini-tables at the part or section level, depending of the document class.

1.2.2 Preparing the mini-tables

\dominitoc
\dominilof
\dominilot

The commands[6] `\dominitoc`, `\dominilof`, and `\dominilot` (for mini-tables at the

[5] The concept of an "absolute" counter for the mini-tables has solved some obscure problems, and also made obsolete some commands, like `\firstpartis`, `\firstchapteris`, and `\firstsectionis`.

chapter level), take the ⟨*document*⟩`.toc`, ⟨*document*⟩`.lof`, and ⟨*document*⟩`.lot` files, respectively, and cut slices from them to create the ⟨*document*⟩`.mtc`⟨*N*⟩, ⟨*document*⟩`.mlf`⟨*N*⟩, and ⟨*document*⟩`.mlt`⟨*N*⟩ files.

`\dosecttoc` `\dosectlof` `\dosectlot` `\doparttoc` `\dopartlof` `\dopartlot` `\tableofcontents` `\listoffigures` `\listoftables`

The commands `\dosecttoc`, `\dosectlof`, and `\dosectlot` (for mini-tables at the section level) and `\doparttoc`, `\dopartlof`, and `\dopartlot` (for mini-tables at the part level) are analog.

To obtain a satisfactory result (*i.e.* non empty), please note that all these commands must *imperatively be put before* any command analog to the `\tableofcontents`, `\listoffigures`, and `\listoftables` commands, or their `\fake...` siblings.

It is also *strongly* recommended to put these commands *before* any sectionning command producing an entry in the table of contents (for the `\do...toc` commands), and *before* any `\caption`-like command producing an entry in the list of figure (for the `\do...lof` commands) or in the list of tables (for the `\do...lot`) commands; else disorder in the mini-tables may result.

1.2.3 Placing the mini-tables

`\mtcskip` `\mtcskipamount` `\bigskipamount`

The `\mtcskip` command may be used to add a vertical skip between two mini-tables. Its height is `\mtcskipamount` (equal to `\bigskipamount` by default). `\mtcskip` eliminates any immediate previous vertical skip, to not accumulate vertical space when a mini-table is empty and skipped by the `checkfiles` option.

`\secttoc` `\section`

The section-level table of contents will be in the section, after the `\section` command, at the point of the `\secttoc` command. The `\secttoc` command may occur *almost anywhere* inside a section. Of course, it is better to put it at the beginning of the section, or after some short introductory material. You should use the same conventions in all sections. If you want to add the section-level table of contents for a section, you must use the sequence given in Table 1.2 on the next page. For each section-level table of contents, an auxiliary file will be created with a name of the form ⟨*document*⟩`.stc`⟨*N*⟩, where ⟨*N*⟩ is the absolute section number. The suffix is `.slf`⟨*N*⟩ for section-level lists of figures and is `.slt`⟨*N*⟩ for section-level lists of tables. (If under MS-DOS or any operating system with short extensions to filenames, see section 1.6 on page 40 and section 2.5 on page 44).

`\usepackage` `\FloatBarrier`

As floats (figures and tables) could drift somewhere outside the printing area of the text of the section, the sectlofs and sectlots can be rather strange. In order to have a better behaviour of these mini-tables, it may be useful to add the `insection` option in the `\usepackage` command:

```
\usepackage[insection]{minitoc}
```

if you want more coherent sectlofs and sectlots. Sometimes, it might be necessary to use the `\FloatBarrier` command of this package to correctly place the figure or ta-

[6]The code of these `\do...` commands is directly derived from that of the `xr` package [12], by David CARLISLE, with his permission.

Table 1.2: Commands for a `secttoc`

```
\documentclass[...]{article}
\usepackage[...options...]{minitoc}
...
\setlength{\stcindent}{24pt}   default
...
\setcounter{secttocdepth}{2}   default
\renewcommand{\stcfont}{\small\rmfamily\upshape\mdseries}           default
\renewcommand{\stcSSfont}{\small\rmfamily\upshape\bfseries}         default
or:
\mtcsetdepth{secttoc}{2}                                            default
\mtcsetfont{secttoc}{*}{\small\rmfamily\upshape\mdseries}           default
\mtcsetfont{secttoc}{subsection}{\small\rmfamily\upshape\bfseries}   default
...
\begin{document}
...
\dosecttoc
\dosectlof
\dosectlot
\tableofcontents               or \faketableofcontents
\listoffigures                 or \fakelistoffigures
\listoftables                  or \fakelistoftables
...
\section{...}
\secttoc                       if you want one
\sectlof                       if you want one
\sectlot                       if you want one
...
```

ble and have a correct mini-table. The `insection` option loads the `placeins` package [3] with its `verbose`, `section` and `bottom` options. The `placeins` package, by Donald ARSENAU, is available on CTAN archives; `placeins.sty` contains its own documentation, with a copy in `placeins.txt`. You need a version whose date is at least 2005/04/18.

If you want to add the partial table of contents for a part, you must use the sequence given in Table 1.3 on the following page. For each partial table of contents, an auxiliary file will be created with a name of the form ⟨*document*⟩`.ptc`⟨*N*⟩, where ⟨*N*⟩ is the absolute part number. The suffix is `.plf`⟨*N*⟩ for partial lists of figures and is `.plt`⟨*N*⟩ for partial lists of tables. (If under MS-DOS or any operating system with short extensions to filenames, see section 1.6 on page 40 and section 2.5 on page 44).

Note: the user is responsible of asking or not asking a mini-table (mini-toc, -lof or -lot) for some chapter. Asking a minilof for a chapter without any figure would result in an empty and ugly mini-list of figures (*i.e.* the title and two horizontal rules). He is also responsible of requiring or not requiring a partial toc (lof or lot) for some part. Asking a partlof for a part without any figure would result in an empty and ugly part list of figures (*i.e.* the title alone on a page). Analogous remarks apply to section-level mini-tables (secttoc, sectlof, and sectlot) and to the part-level mini-tables (parttoc, partlof, and partlot).

Table 1.3: Commands for a parttoc

```
\documentclass[...]{book}
\usepackage[...options...]{minitoc}
...
\setlength{\ptcindent}{0pt}    default
...
\setcounter{parttocdepth}{2}   default
\renewcommand{\ptcfont}{\normalsize\rmfamily\upshape\mdseries}        default
\renewcommand{\ptcCfont}{\normalsize\rmfamily\upshape\bfseries}       default
\renewcommand{\ptcSfont}{\normalsize\rmfamily\upshape\mdseries}       default
or:
\mtcsetdepth{parttoc}{2}                                              default
\mtcsetfont{parttoc}{*}{\normalsize\rmfamily\upshape\mdseries}        default
\mtcsetfont{parttoc}{chapter}{\normalsize\rmfamily\upshape\bfseries}   default
\mtcsetfont{parttoc}{section}{\normalsize\rmfamily\upshape\mdseries}   default
...
\begin{document}
...
\doparttoc
\dopartlof
\dopartlot
\tableofcontents               or \faketableofcontents
\listoffigures                 or \fakelistoffigures
\listoftables                  or \fakelistoftables
...
\part{...}
\parttoc                       if you want one
\partlof                       if you want one
\partlot                       if you want one
...
```

But since version #35, empty mini-tables are just ignored and this problem should disappear in normal circumstances. Nevertheless, it is recommended to put no `\minitoc` command in a chapter without sections and no `\minilof` or `\minilot` command in a chapter without figures or tables. The `checkfiles` (see section 1.2.3 on page 20) package option (default) skips empty mini-tables (with a note in the `.log` file); the `nocheckfiles` package option restores the old behaviour (empty mini-tables are displayed).

By default, the mini-tables and partial tables of contents contain only references higher and to sections and subsections. The counters `parttocdepth`, `minitocdepth` and `secttocdepth`, similar to `tocdepth`, allow the user to modify this behaviour. Mini or partial lists of figures or tables are not affected by the value of these counters, but if there are depth counters for these lists (`lofdepth` and `lotdepth`), as done by the `subfig` package [13] from Steven Douglas Cochran, new depth counters are created if necessary, with obvious names like `partlofdepth`, `partlotdepth`, `minilofdepth`, `minilotdepth`, `sectlofdepth`, and `sectlotdepth`.

1.2.4 Starred chapters, parts and sections

\addstarredpart
\addstarredchapter
\addstarredsection

NOTE: if using `\chapter*` and a

```
\addcontentsline{toc}{chapter}{...}
```

command to add something in the table of contents, the numbering of the `minitoc` auxiliary files would be altered. To avoid that problem, say:

```
\addstarredpart{...}
\addstarredchapter{...}
\addstarredsection{...}
```

\addcontentsline

These commands apply only for the level of a part-, mini- or sect-toc; for lower levels, use, as usual:

```
\addcontentsline{toc}{section}{...}
```

for example, to add a section-level entry in the global toc and in the minitoc of a starred chapter:

```
\chapter*{Title of chapter}
\addstarredchapter{Title of chapter}
\minitoc
\section*{First section}
\addcontentsline{toc}{section}{First section}
\section*{Second section}
\addcontentsline{toc}{section}{Second section}
```

\adjustptc
\adjustmtc
\adjuststc

There is sometimes a problem with mini-tables when you use `\chapter*` (or `\section*`): the minitocs appear in the wrong chapter. You can add a `\adjustmtc` (or `\adjuststc` or `\adjustptc`) command at the end of the starred chapter (or section or part) to increment the corresponding counter. Do not use commands like `\stepcounter{mtc}` or `\addtocounter{mtc}{...}` (which should work, but it is cheating), because the `mtcoff` package (see section 1.8 on page 41) knows what to do about `\adjustmtc` (and others), but can do nothing about `\stepcounter` or `\addtocounter`, as they are a standard basic commands of LaTeX, not `minitoc` specific commands. Syntax:

```
\adjustptc[n]
\adjustmtc[n]
\adjuststc[n]
```

where *n* is the increment (default: 1).

Table 1.4: Commands to add an entry in the table of contents for a starred part, chapter, or section.

Level	With title
part	`\mtcaddpart[`⟨*title*⟩`]`
chapter	`\mtcaddchapter[`⟨*title*⟩`]`
section	`\mtcaddsection[`⟨*title*⟩`]`

`\decrementptc`
`\decrementmtc`
`\decrementstc`
`\incrementptc`
`\incrementmtc`
`\incrementstc`

There are similar commands to *decrement* or *increment* by 1 these counters: `\decrementptc`, `\decrementmtc`, `\decrementstc`, `\incrementptc`, `\incrementmtc`, and `\incrementstc`; the same remarks as above apply.

`\mtcaddpart`
`\mtcaddchapter`
`\mtcaddsection`

A more clever way to solve this problem is to use commands similar to:

`\mtcaddchapter[`⟨*title*⟩`]`

This command adds an entry in the table of contents (and adjusts the counter, because it calls `\adjustmtc`). The table 1.4 summarizes these commands, that you put *after* `\chapter*`, etc. If the optional argument is omitted or empty or blank, no entry will be visible in the table of contents nor in the minitocs. If the optional argument is something invisible (like ~, `\space` or `\quad`), the result will be strange but still logically correct.

1.3 Typesetting of the mini-tables

The mini-tables are typeset in a `verse`-like environment, and can be split over pages.

1.3.1 Chapter-level mini-tables

`\mtcfont`
`\mtcSfont`
`\mtcSSfont`
`\mtcSSSfont`
`\mtcPfont`
`\mtcSPfont`
`\mlffont`
`\mltfont`

The mini-table of contents is typeset in the `\mtcfont` font, which is `\small\rmfamily` by default. Section entries are typeset in the `\mtcSfont` font, which is `\small\bfseries` by default. For subsections, subsubsections, paragraphs and subparagraphs, the commands `\mtcSSfont`, `\mtcSSSfont`, `\mtcPfont` and `\mtcSPfont` are available (by default, `\small\rmfamily`) to enable the use of various fonts. Mini lists of figures and tables are typeset in the fonts `\mlffont` and `\mltfont`, which are `\small\rmfamily` by default. Tables 1.5 on the following page and 1.6 on page 26 summarize these many commands[7].

Note that the default choice of fonts is certainly not perfect and hence is not definitive. A symptom of this imperfection is the presence of poor alignments in the mini-tables, if

[7]Thanks to Stefan ULRICH, who contributed these tables initially.

Table 1.5: Fonts and titles for the mini-table commands.

Command	Font default setting	Title string default setting	Title font default setting
For the \part... commands:			
\parttoc	\ptcfont	\ptctitle	\ptifont
	\normalsize\rmfamily*	Table of Contents†	\LARGE\bfseries*
	\small\rmfamily**		\Large\bfseries**
\partlof	\plffont	\plftitle	\ptifont
	\normalsize\rmfamily*	List of Figures†	\LARGE\bfseries*
	\small\rmfamily**		\Large\bfseries**
\partlot	\pltfont	\plttitle	\ptifont
	\normalsize\rmfamily*	List of Tables†	\LARGE\bfseries*
	\small\rmfamily**		\Large\bfseries**
For the \mini... commands:*			
\minitoc	\mtcfont	\mtctitle	\mtifont
	\small\rmfamily	Contents†	\large\bfseries
\minilof	\mlffont	\mlftitle	\mtifont
	\small\rmfamily	Figures†	\large\bfseries
\minilot	\mltfont	\plttitle	\mtifont
	\small\rmfamily	Tables†	\large\bfseries
For the \sect... commands:**			
\secttoc	\stcfont	\stctitle	\stifont
	\small\rmfamily	Contents†	\Large\bfseries
\sectlof	\slffont	\mlftitle	\stifont
	\small\rmfamily	Figures†	\Large\bfseries
\sectlot	\sltfont	\plttitle	\stifont
	\small\rmfamily	Tables†	\Large\bfseries

*for document classes with \chapter level (e.g. book, report).
**for document classes with no \chapter level (e.g. article).
†default for english; changed by the language definition files or \renewcommand .
All these fonts use \rmfamily, \upshape, and \mdseries by default.

bold and non-bold font are mixed [8] (the true length of 1em is not the same for the fonts). This can eventualy be adjusted by changing some fonts.

1.3.2 Titles for chapter-level mini-tables

\mtifont
\mtctitle
\mlftitle
\mlttitle
\mtcsettitle
\mtcselectlanguage

Titles are typeset in the \mtifont (\large\bfseries by default) font and the text strings of the titles are defined by \mtctitle, \mlftitle and \mlttitle, which are the strings “Contents”, “Figures” and “Tables” by default. These title commands should be redefined by \renewcommand or \mtcsettitle for languages other than english. The language option files like french.mld and english.mld (the suffix .mld means “minitoc language definition (file)”) (and others, see table 1.7 on page 27 and section 1.3.12 on page 31) are available. You can easily prepare a similar file for a preferred language (see section 2.26 on page 51). You can change the language of these titles by using the \mtcselectlanguage{*language*} macro.

[8]This appears e.g. if you are using the Computer Modern Roman (CMR) fonts. The symptom disappears if you do not use bold CMR fonts or if you use the TX fonts (txfonts package [40]), by example, like in this document. See also section 2.29 on page 52.

Table 1.6: Fonts for the mini-table entries.

Level	Font	default setting
For the \parttoc entries:		
Chapter*	\ptcCfont*	\normalsize\bfseries*
Section	\ptcSfont	\normalsize\rmfamily*
		\small\bfseries**
Subsection	\ptcSSfont	*(like \ptcfont)*
Subsubsection	\ptcSSfont	*(like \ptcfont)*
Paragraph	\ptcPfont	*(like \ptcfont)*
Subparagraph	\ptcSPfont	*(like \ptcfont)*
For the \minitoc entries:*		
Section	\mtcSfont	\small\bfseries
Subsection	\mtcSSfont	*(like \mtcfont)*
Subsubsection	\mtcSSfont	*(like \mtcfont)*
Paragraph	\mtcPfont	*(like \mtcfont)*
Subparagraph	\mtcSPfont	*(like \mtcfont)*
For the \secttoc entries:**		
Subsection	\stcSSfont	\normalsize\bfseries
Subsubsection	\stcSSfont	*(like \stcfont)*
Paragraph	\stcPfont	*(like \stcfont)*
Subparagraph	\stcSPfont	*(like \stcfont)*

*for document classes with \chapter level (e.g. book, report).
**for document classes with no \chapter level (e.g. article).

1.3.3 Part-level mini-tables

\ptcfont \ptcCfont \ptcSfont \ptcSSfont \ptcSSSfont \ptcPfont \ptcSPfont \plffont \pltfont

The partial table of contents is typeset in the \ptcfont font, which is defined as \normalsize\rmfamily by default. Chapter entries are typeset in the \ptcCfont font, which is \normalsize\bfseries by default. Section entries are typeset in the \ptcSfont font, which is \normalsize\rmfamily by default. For subsections, subsubsections, paragraphs and subparagraphs, the commands \ptcSSfont, \ptcSSSfont, \ptcPfont, and \ptcSPfont are available (by default, \normalsize\rmfamily) if you want to use various fonts. Partial lists of figures and tables are typeset in the fonts \plffont and \pltfont, which are \normalsize\rmfamily by default.

1.3.4 Titles for part-level mini-tables

\ptifont \ptctitle \plftitle \plttitle \mtcsettitle \mtcselectlanguage

Titles are typeset in the \ptifont (\LARGE\bfseries by default) font and the text strings of the titles are defined by \ptctitle, \plftitle and \plttitle, which are the strings “Table of Contents”, “List of Figures” and “List of Tables” by default. These title commands should be redefined by \renewcommand or \mtcsettitle for languages other than english. The language definition files like french.mld and english.mld (and many others, see table 1.7 on the following page and

Table 1.7: Available languages

1. afrikaan (afrikaans)
2. arab (arabic) [a,c]
3. armenian [c]
4. bahasa [c]
5. bangla [c]
6. basque
7. bicig [c]
8. brazil (brazilian)
9. breton
10. bulgarian [c]
11. bulgarianb [c]
12. buryat [c]
13. catalan
14. chinese1 [c,g]
15. chinese2 [c,g]
16. croatian
17. czech
18. danish
19. dutch
20. english (american, british, canadian, UKenglish, USenglish)
21. esperant (esperanto)
22. estonian
23. ethiopia (ethiopian) [c]
24. farsi1 [c,f,g]
25. farsi2 [c,f,g]
26. finnish
27. finnish2
28. french (frenchb, frenchle, frenchpro, francais, acadien, canadien)
29. galician
30. german (austrian)
31. germanb
32. greek [c]
33. greek-mono [c,e]
34. greek-polydemo [c,e]
35. greek-polykatha [c,e]
36. guarani [h]
37. hangul1 [c,d,g]
38. hangul2 [c,d,g]
39. hangul3 [c,d,g]
40. hangul4 [c,d,g]
41. hanja1 [c,d,g]
42. hanja2 [c,d,g]
43. hebrew [c,d]
44. icelandic [f]
45. interlingua
46. irish
47. italian
48. japanese [c,d,g]
49. japanese2 [c,d,g]
50. japanese3 [c,d,g]
51. japanese4 [c,d,g]
52. japanese5 [c,d,g]
53. latin
54. latin2
55. latvian (letton) [e]
56. lithuanian
57. lsorbian
58. magyar (hungarian)
59. magyar2
60. malayalam-keli [c]
61. malayalam-rachana [c]
62. malayalam-rachana2 [c]
63. mongol [c]
64. ngermanb (ngerman, naustrian)
65. norsk
66. nynorsk
67. polish
68. polish2 [c,e]
69. portuguese (portuges)
70. romanian
71. russian [b,c]
72. russianb [b,c]
73. russianc [b,c]
74. russian2m [c,e]
75. russian2o [c,e]
76. samin
77. scottish
78. serbian
79. serbianc [c]
80. slovak
81. slovene
82. spanish (castillan, castillian)
83. spanish2
84. spanish3 [e,f]
85. swedish
86. thai [c,d,f,g]
87. turkish
88. ukraineb [b,c]
89. usorbian
90. vietnam (vietnamese) [c,d]
91. welsh

[a] The `arab(ic)` language requires the use of ArabTEX [25, 26].
[b] The `russian` language is not yet supported by `babel` [10], but `russianb` is supported if you use babel-3.6 or higher ; `russianc` is an extra.
[c] Some languages may require specific fonts.
[d] Requires the CJK package [29].
[e] Requires Lambda, the version of LATEX for Omega.
[f] Requires a 8-bit input encoding.
[g] Uses also a `.mlo` file.
[h] Requires a specific input encoding.

section 1.3.12 on page 31) are available. You can easily prepare a similar file for a preferred language (see section 2.26 on page 51). You can change the language of these titles by using the `\mtcselectlanguage{`*language*`}` macro.

1.3.5 Section-level mini-tables

\stcfont \stcSSfont \stcSSSfont \stcPfont \stcPSfont \slffont \sltfont

The section-level table of contents is typeset in the \stcfont font, which is defined as \normalsize\rmfamily by default. Subsection entries are typeset in the \stcSSfont font, which is \normalsize\bfseries by default. Subsubsection entries are typeset in the \stcSSSfont font, which is \normalsize\rmfamily by default. For paragraphs and subparagraphs, the commands \stcPfont and \stcSPfont are available (by default, \normalsize\rmfamily) if you want to use various fonts. Section-level lists of figures and tables are typeset in the fonts \slffont and \sltfont, which are defined as \normalsize\rmfamily by default.

1.3.6 Titles for section-level mini-tables

\stifont \stctitle \slftitle \slttitle \mtcsettitle \mtcselectlanguage

Titles are typeset in the \stifont (\normalsize\bfseries by default) font and the text strings of the titles are defined by \stctitle, \slftitle and \slttitle, which are the strings “Contents”, “Figures” and “Tables” by default. These title commands should be redefined by \renewcommand or \mtcsettitle for languages other than english. The language option files like french.mld and english.mld (and also many others, as listed in table 1.7 on the page before and explained in section 1.3.12 on page 31) are available. You can easily prepare a similar file for your preferred language (see section 2.26 on page 51). You can change the language of these titles by using the \mtcselectlanguage{*language*} macro.

1.3.7 Position of the titles

1.3.7.1 For mini-tables at the part level

\doparttoc \dopartlof \dopartlot \parttoc \partlof \partlot

By default, titles are on the left. The preparation commands \doparttoc, \dopartlof and \dopartlot accept an optional argument to change the default position of the corresponding title: [l] for left (default), [c] for center, [r] for right, or [e] (or [n]) for empty (no title). The change is global for all the document. If you want to change the position of the title for only one parttoc (or partlof or partlof), just use such an optional argument with the command \parttoc (or \partlof or \partlot).

1.3.7.2 For mini-tables at the chapter level

\dominitoc \dominilof \dominilot \minitoc \minilof \minilot

By default, titles are on the left. The preparation commands \dominitoc, \dominilof and \dominilot accept an optional argument to change the default position of the corresponding title: [l] for left (default), [c] for center, [r] for right, or [e] (or [n]) for “empty” (“no” title). The change is global for all the document. If you want to change the position of the title for only one minitoc (or minilof or minilof), just use such an optional argument with the command \minitoc (or \minilof or \minilot).

1.3.7.3 For mini-tables at the section level

\dosecttoc \dosectlof \dosectlot \secttoc \sectlof \sectlot

By default, titles are on the left. The preparation commands `\dosecttoc`, `\dosectlof` and `\dosectlot` accept an optional argument to change the default position of the corresponding title: `[l]` for left (default), `[c]` for center, `[r]` for right, or `[e]` (or `[n]`) for empty (no title). The change is global for all the document. If you want to change the position of the title for only one secttoc (or sectlof or sectlof), just use such an optional argument with the command `\secttoc` (or `\sectlof` or `\sectlot`).

1.3.7.4 Summary of the positionning of titles

\doparttoc \dopartlof \dopartlot \dominitoc \dominilof \dominilot \dosecttoc \dosectlof \dosectlot \parttoc \partlof \partlot \minitoc \minilof \minilot \secttoc \sectlof \sectlot

To summarize: by default, all titles are on the left. However, each one of the following preparation commands:

`\doparttoc`, `\dopartlof`, `\dopartlot`,
`\dominitoc`, `\dominilof`, `\dominilot`,
`\dosecttoc`, `\dosectlof`, `\dosectlot`

accepts an optional argument to change the positionning of the title: `[l]` for left (default), `[c]` for center, `[r]` for right, `[e]` or `[n]` for empty (no title), for all the corresponding mini-tables. The following insertions commands:

`\parttoc`, `\partlof`, `\partlot`,
`\minitoc`, `\minilof`, `\minilot`,
`\secttoc`, `\sectlof`, `\sectlot`

accept the same optional arguments, but these options change the positionning only for the title of the current mini-table.

1.3.8 Line spacing in the mini-tables

\iftightmtc \tightmtctrue \tightmtcfalse

With the commands `\tightmtctrue` (or the `tight` package option) and `\tightmtcfalse` (or the `loose` package option, which is the default), the mini-tables will have less (tight) or more (loose) space between contents lines.

But with the Koma-script classes (`scrartcls`, `scrbook` and `scrreprt`), it may sometimes be necessary to use the following options or commands, because we need to set `\parskip` to zero in place of `\parsep` to tighten the mini-table. The efficiency of the following options depends of the options given to these Koma-script classes (`parindent` option, `parskip` option and variants).

\ifktightmtc \ktightmtctrue \ktightmtcfalse

For the Koma-script classes, with the commands `\ktightmtctrue` (or the `k-tight` package option) and `\ktightmtcfalse` (or the `k-loose` package option, which is the default), the mini-tables will have less (tight) or more (loose) space between contents lines.

1.3.9 Simplified commands for fonts

\mtcsetfont
\mtcsettitlefont

To simplify the redefinition of the fonts for mini-tables, there are two useful commands:

\mtcsetfont{*mini-table*}{*sectionning-level*}{*commands*}
\mtcsettitlefont{*mini-table*}{*commands*}

By example,

```
\mtcsetfont{minitoc}{subsection}%
           {\small\rmfamily\upshape\bfseries}
```

will redefine `\mtcSSfont` with the given font commands.

Note that `\mtcsetfont{parttoc}{*}{...}` allows also to redefine `\ptcfont`, etc.

Moreover,

```
\mtcsettitlefont{parttoc}{\Large\rmfamily\itshape\mdseries}
```

will redefine `\ptifont` (for titles in the parttocs, partlofs and partlofs) with the given font commands.

1.3.10 Simplified command for mini-table titles

\mtcsettitle

To simplify the redefinition of the titles for mini-tables, the `\mtcsettitle` command is also available:

\mtcsettitle{*mini-table*}{*title string*}

By example,

```
\mtcsettitle{minitoc}{Description of contents}
```

will redefine `\mtctitle` with the given string. This command checks that you redefine a title for a mini-table type available in your document class.

1.3.11 Simplified command for mini-table depths

`\mtcsetdepth`

To simplify the redefinition of the depths for mini-tables, the `\mtcsetdepth` command is also available:

`\mtcsetdepth{`*mini-table*`}{`*depth*`}`

By example,

```
\mtcsetdepth{minitoc}{4}
```

will set the counter `minitocdepth` with the given value. This command checks that you set a depth for a mini-table type available in your document class (and that it is possible to change its depth).

1.3.12 Languages for the titles

Most of the strings defined in the language option files (`.mld`) were taken from the superb `babel` [10] package by Johannes Braams, some were adapted, others were made available by gentle users or taken from specific packages, like ArabTEX [25, 26], ArmTEX (armenian) [16], BangTEX (bangla) [36], `ethiop` [6], `guarani` [7], Malayalam [1], Montex (mongol) [14, 15], CJK (chinese, corean-hangul/hanja, japanese, thai) [29], FarsiTEX (farsi or iranian), or `vietnam.sty` — latvian (letton), greek-mono, greek-polydemo, greek-polykatha, polish2, russian2m, russian2o and spanish3 need Lambda, *i.e.* the Omega version of LATEX, (see [24]), or even found by searching on the Web (`bulgarianb.mld` for upper cyrillic bulgarian, `japanese.mld` for japanese, `serbianc.mld` for cyrillic serbian). Other languages are welcome. See table 1.7 on page 27.

But for some oriental languages[9], the sources of the titles use some exotic encodings which are difficult to manipulate in a `.dtx` file, hence the `.mld` file is then just a wrapper which loads a special file, nicknamed a `.mlo` file[10], not generated by the `.dtx` file in the current version of `minitoc` package, but via `filecontents` environments in the `minitoc.ins` file, and playing with the “catcode” of the “delete” character.

1.3.13 Altering the layout of the mini-tables *(experimental feature)*

The layout of a mini-table is described in the figure 1.1 on the next page (this figure is adapted from [44]), which defines some internal commands (these are not *dimensions*, but LATEX commands, created by `\newcommand`, modifiable via `\renewcommand`).

[9] Mainly for chinese, farsi, hangul (korean), hanja (korean), japanese and thai variants.
[10] The suffix `.mlo` means *minitoc language object*.

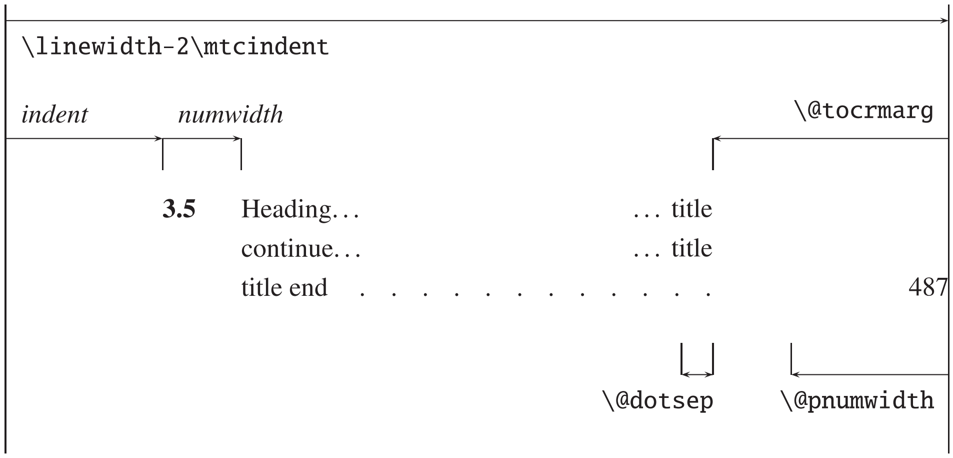


Figure 1.1: Layout of a ToC (LoF, LoT) entry

- `\@dotsep`, which is the separation between the dots in the dotted line. It is a pure number expressing *math units*; 18 math units make `1em` (one quad), which is about the width of a "m" in the current font.
- `\@pnumwidth`, is the width of the space reserved for the page number. It is a LaTeX command containing the representation of a length (e.g. `1.55em`).
- `\@tocrmarg`, is the distance (margin) between the right border of the table and the end of the dotted line. It should be larger than `\@pnumwidth`, and can be a rubber length (*i.e.* contains some glue, like `2.55em plus 1fil`); if you specify the "`... plus 1fil`" portion, the text of the entry will be ragged on right; it is useful if you have long entries, and it can avoid most hyphenations.

`\mtcsetformat` As these commands are internal (their names contain the "`@`" character) and must have a local effect only on specific mini-tables, you should alter then indirectly via the `\mtcsetformat` command:

`\mtcsetformat{`*mini-table*`}{`*parameter*`}{`*value*`}`

where *mini-table* is one of the `parttoc`, `partlof`, `partlot`, `minitoc`, `minilof`, `minilot`, `secttoc`, `sectlof` or `sectlot` keywords; *parameter* is one of the `dotinterval` (for `\@dotsep`), `pagenumwidth` (for `\@pnumwidth`), or `tocrightmargin` (for `\@tocrmarg`) keywords; so:

```
\mtcsetformat{partlof}{tocrightmargin}{2.55em plus 1fil}
```

will set the right margin to `2.55em plus 1fil` in the lists of tables at the part level. The elasticity (`plus 1fil`) is useful if the table captions are long.

Note that the `tocrightmargin` (for `\@tocrmarg`) parameter should obviously be greater than the `pagenumwidth` parameter (this appears in the figure 1.1).

Table 1.8: Horizontal rules

				defaults for		
	rules in		no rules in	book	report	article
\ptcrule	parttocs	\noptcrule	parttocs	**N**	**N**	**Y**
\plfrule	parttocs	\noplfrule	parttocs	**N**	**N**	**Y**
\pltrule	parttocs	\nopltrule	parttocs	**N**	**N**	**Y**
\mtcrule	minitocs	\nomtcrule	minitocs	**Y**	**Y**	**N-A**
\mlfrule	minitocs	\nomlfrule	minitocs	**Y**	**Y**	**N-A**
\mltrule	minitocs	\nomltrule	minitocs	**Y**	**Y**	**N-A**
\stcrule	secttocs	\nostcrule	secttocs	**N-A**	**N-A**	**Y**
\slfrule	secttocs	\noslfrule	secttocs	**N-A**	**N-A**	**Y**
\sltrule	secttocs	\nosltrule	secttocs	**N-A**	**N-A**	**Y**

Table 1.9: Page numbers

Type	**Page numbers (Default)**	**No page numbers**
parttoc	\ptcpagenumbers	\noptcpagenumbers
minitoc	\mtcpagenumbers	\nomtcpagenumbers
secttoc	\stcpagenumbers	\nostcpagenumbers
partlof	\plfpagenumbers	\noplfpagenumbers
minilof	\mlfpagenumbers	\nomlfpagenumbers
sectlof	\slfpagenumbers	\noslfpagenumbers
partlot	\pltpagenumbers	\nopltpagenumbers
minilot	\mltpagenumbers	\nomltpagenumbers
sectlot	\sltpagenumbers	\nosltpagenumbers

If the `dotinterval` parameter (for `\@dotsep`) is very large (try 450, then increase or decrease), the dots of leaders will be so much spread out that they will disappear.

1.4 Special Features

1.4.1 Horizontal Rules

`\mtcsetrules` By default, most of mini-tables have horizontal rules after their titles and at their ends. The exception is the "parttoc" in a `book`- or `report`-like document (*i.e.* when `\chapter` is defined). To activate or deactivate these rules, the commands of the table 1.8 are available. But you can also use the following command, which is simpler:

`\mtcsetrules{`*mini-table*`}{on|off}`

where *mini-table* is one of the `parttoc`, `partlof`, `partlot`, `minitoc`, `minilof`, `minilot`, `secttoc`, `sectlof`, or `sectlot` keywords; the keywords `on` and `off` have the following synonyms [11] :

- `on`, `ON`, `yes`, `YES`, `y`, `Y`, `true`, `TRUE`, `t`, `T`, `vrai`, `VRAI`, `v`, `V`, `oui`, `OUI`, `o`, `O`, and `1`;
- `off`, `OFF`, `no`, `NO`, `n`, `N`, `false`, `FALSE`, `faux`, `FAUX`, `f`, `F`, `non`, `NON`, and `0`.

1.4.2 Page Numbers, Leaders

`\mtcsetpagenumbers`

By default, the page numbers are listed in each minitoc, minilof, etc. Some authors want only the section titles (with the section numbers), but without page numbers. Hence the obvious declarations of table 1.9 on the preceding page are available. But you can also use the following command:

`\mtcsetpagenumbers{`*mini-table*`}{on|off}`

where *mini-table* is one of the `parttoc`, `partlof`, `partlot`, `minitoc`, `minilof`, `minilot`, `secttoc`, `sectlof`, or `sectlot` keywords; the keywords `on` and `off` have the following synonyms [11] :

- `on`, `ON`, `yes`, `YES`, `y`, `Y`, `true`, `TRUE`, `t`, `T`, `vrai`, `VRAI`, `v`, `V`, `oui`, `OUI`, `o`, `O`, and `1`;
- `off`, `OFF`, `no`, `NO`, `n`, `N`, `false`, `FALSE`, `faux`, `FAUX`, `f`, `F`, `non`, `NON` and `0`.

In the mini-tables, they are leaders of dots between the section titles and the page numbers. The `undotted` package option removes these dots. The `dotted` package option is the default. See also section 1.3.13 on page 31.

1.4.3 Features for parttocs and other mini-tables

By default, a parttoc (or a partlof or a partlot) is preceeded and followed by a `\cleardoublepage`, and has a page style of `empty`. Since version #32, you can modify this behaviour by redefining the commands of table 1.10 on the following page, whose meaning is obvious.

`\mtcsetfeature`

The command:

`\mtcsetfeature{`*mini-table*`}{`*keyword*`}{`*commands*`}`

[11] `O` and `o` are the letter `O`, `0` is the zero digit.

Table 1.10: Features for parttoc-s and other mini-tables

Type	**Command**	**Default**
parttoc	`\beforeparttoc`	`\cleardoublepage`
parttoc	`\afterparttoc`	`\cleardoublepage`
parttoc	`\thispageparttocstyle`	`\thispagestyle{empty}`
partlof	`\beforepartlof`	`\cleardoublepage`
partlof	`\afterpartlof`	`\cleardoublepage`
partlof	`\thispagepartlofstyle`	`\thispagestyle{empty}`
partlot	`\beforepartlot`	`\cleardoublepage`
partlot	`\afterpartlot`	`\cleardoublepage`
partlot	`\thispagepartlotstyle`	`\thispagestyle{empty}`
minitoc	`\beforeminitoc`	`\empty`
minitoc	`\afterminitoc`	`\empty`
minitoc	`\thispageminitocstyle`	`\empty`
minilof	`\beforeminilof`	`\empty`
minilof	`\afterminilof`	`\empty`
minilof	`\thispageminilofstyle`	`\empty`
minilot	`\beforeminilot`	`\empty`
minilot	`\afterminilot`	`\empty`
minilot	`\thispageminilotstyle`	`\empty`
secttoc	`\beforesecttoc`	`\empty`
secttoc	`\aftersecttoc`	`\empty`
secttoc	`\thispagesecttocstyle`	`\empty`
sectlof	`\beforesectlof`	`\empty`
sectlof	`\aftersectlof`	`\empty`
sectlof	`\thispagesectlofstyle`	`\empty`
sectlot	`\beforesectlot`	`\empty`
sectlot	`\aftersectlot`	`\empty`
sectlot	`\thispagesectlotstyle`	`\empty`

`\mtcsetfeature{`*mini-table*`}{before|after|pagestyle}{`*command*`}`
Modifies the features for a mini-table.

allows you to redefine any of these commands. *mini-table* is one of the mini-table names: `parttoc`... `sectlot`. *keyword* is one of the followings: `before`, `after` or `pagestyle`. *commands* is either a sequence of commands like `\clearpage`, `\cleardoublepage`, `\thispagestyle{...}`, etc., either `\empty` (does nothing).

1.4.3.1 Remark about page styles [12]

The default commands for part-level mini-tables page styles are defined as being `\thispagestyle{empty}`, because in document classes defining the `\chapter` command, the part-level mini-tables are on their own pages. If the document is printed recto-verso, the first page is recto. Usually, these pages are not numbered and have no header and no footer. This behaviour comes from the default definitions of the commands of table 1.10 on the page before. If you want an other behaviour, you can change these definitions. Note that, by default, only the *first* page of these mini-tables are in the `empty` page style. You can set the style of this first page by using `\thispagestyle` and set the style of the following pages by using `\pagestyle`, but you must not forget to reset the normal style after the mini-table. Look at this short example [13]:

```
\mtcsetfeature{parttoc}{before}%
  {\cleardoublepage}
\mtcsetfeature{parttoc}{thispagestyle}%
  {\thispagestyle{empty}\pagestyle{myheadings}}
\mtcsetfeature{parttoc}{after}%
  {\cleardoublepage\pagestyle{headings}}
```

where we add a `\cleardoublepage` before each parttoc, then we set the `empty` page style for the first page of the parttocs, the `myheadings` page style for the following pages of the parttocs, and set `headings` page style for the pages after the mini-table, after a `\cleardoublepage`.

1.4.4 The "Chapter 0" Problem (solved)

Some documents do not begin with chapter number one, but with chapter number zero (or even a weirder number).

1.4.4.1 Before version #23 (1994/11/08)

`\firstpartis`
`\firstchapteris`
`\firstsectionis`

To make the `minitoc` package work with such documents, you must insert the command:

[12]This remark is taken and adapted from a draft of the second edition of the JMPL [4], by Benjamin Bayart, where he comments the `minitoc` package.

[13]This example shows that the third argument can be a *sequence* of commands: we set the style of the current page and the style of the following pages.

```
\firstchapteris{⟨N⟩}
```

before the `\dominitoc` and analogous commands. ⟨*N*⟩ is the number of the first chapter. This command *does not* modify the numbering of chapters, you must use a `\addtocounter{chapter}{-1}` command to get a first chapter numbered 0. The `\firstpartis` and `\firstsectionis` commands are similar for parts and sections with a non standard numbering.

1.4.4.2 Since version #23 (1994/11/08)

These commands are now obsolete, as this problem has been solved (via the "absolute" numbering of the mini-table auxiliary files). Thus they just produce harmless warnings.

1.4.5 Special Entries for TOC, LOF, LOT, Bibliography and Index

If you want to add entries in the Table of Contents for objects like the Table of Contents itself, the List of Figures, the List of Tables, the Bibliography or the Index, you should use the `tocbibind` [47] package by Peter R. Wilson (this package is available from the CTAN archives).

`\dominitoc`

But these entries are considered as chapters (or sections in an article class document) when the `.toc` file is scanned to prepare the minitocs (the `\dominitoc` phase).

`\mtcaddchapter` `\tableofcontents` `\listoffigures` `\listoftables` `\adjustmtc` `\bibliography` `\printindex` `\addcontentsline` `\mtcaddchapter` `\mtcfixindex`

So you must add an `\mtcaddchapter` command, *without argument*, after each of the involved commands `\tableofcontents`, `\listoffigures`, and `\listoftables`.

For the bibliography, you should add a `\adjustmtc` command after the `\bibliography` command.

For the index, it is a bit more complicated, you add the following commands just after the `\printindex` command:

```
\addcontentsline{lof}{xchapter}{}
\addcontentsline{lot}{xchapter}{}
\mtcaddchapter
```

But this can be done by:

```
\mtcfixindex[chapter|section|part]
```

where the optional argument is the level for the index entry in the TOC. By default, if `\chapter` is defined, the `chapter` level is used, else the `section` level. If neither `\chapter` or `\section` are defined, the `part` level will be used if `\part` is defined; else an error is reported. It is *recommended* that you check the result and, if necessary, you adjust the optional argument.

\printglossary
\addcontentsline
\mtcaddchapter
\mtcfixglossary

For the glossary, it is like for the index, you add the following commands just after the \printglossary command:

```
\addcontentsline{lof}{xchapter}{}
\addcontentsline{lot}{xchapter}{}
\mtcaddchapter
```

But this can be done by:

```
\mtcfixglossary[chapter|section|part]
```

where the optional argument is the level for the glossary entry in the TOC. By default, if \chapter is defined, the chapter level is used, else the section level. If neither \chapter or \section are defined, the part level will be used if \part is defined; else an error is reported. It is *recommended* that you check the result and, if necessary, you adjust the optional argument.

Of course, in documents were the TOC, LOF, LOT, bibliography and/or index (or glossary) are processed as starred sections, you must modify these additions to use section level commands.

And proceed with care, tracking in the .log file the insertion of .mtc⟨*N*⟩ files (and siblings). They are some examples in the add.tex file distributed with minitoc.

1.4.6 The notoccite option

\cite

This option loads the notoccite [2] package (by Donald Arseneau). It avoids problems with \cite commands in sectionning commands or captions: if you then run BibTeX using the unsrt (unsorted) style, or a similar style, these citations get numbered starting from the page in the table of contents where is the parasite citation, not the number they should have in the main text. The notoccite package prevents this. As minitoc prints TOCs, it is subject to the same problem.

1.5 The hints option

This package option detects some actions and the loading of some packages and classes known as interacting with minitoc, and also some frequent misuses and errors. This list of interacting packages and classes is, of course, not closed. If a known package is loaded, this option writes some hints in the .log file and emits a warning. The hints written in the .log file may suggest you to consult the present document or the minitoc.bug file.

Your advice about this option will be welcome. This option is activated by default. You can inhibit it via the `nohints` option. The following problems and potential problems are currently detected:

\part \@part \spart \@spart \chapter \@chapter \schapter \@schapter \section \@sect \@ssect

- Alteration of some of the following commands[14]: `\part`, `\@part`, `\@spart`, `\chapter`, `\@chapter`, `\@schapter`, `\section`, `\@sect`, and `\@ssect`. Note that the `hyperref` (see section 2.17 on page 48) package alters these commands at `\begin{document}`, hence this problem might be reported if you use this package, but these alterations seem harmless.
- Presence of the following packages or classes, which need some precautions: `amsbook` (class), `memoir` (class), `appendix`, `placeins` (beware to its options and its release date (2005/04/18 at least)), `tocbibind`, and `tocloft`.
- Presence of the following packages or classes, which are incompatible with the `minitoc` package: `amsart` (class), `amsproc` (class), `titlesec`, and `titletoc`[15].

\parttoc \doparttoc \sectlot \dosectlot \tableofcontents \listoftables \faketableofcontents \fakelistoftables \sectlof \sectlot

- Usage of `\parttoc` without calling `\doparttoc`, … , usage of `\sectlot` without calling `\dosectlot`. Or the reverse.
- Usage of `\parttoc` without calling `\[fake]tableofcontents`, … , usage of `\sectlot` without calling `\[fake]listoftables`.
- Usage of `\sectlof` and/or `\sectlot` without using the `insection` package option of `minitoc` (or the `placeins` package without its `section` and `below` options).
- If you are using short extensions (because of your operating system or the `shortext` package option, see section 1.6 on the next page) and go beyond the limit of 99 parts, chapters or sections, the `hints` package option displays a warning.

abstract \mtcaddchapter \mtcaddsection

- If the `abstract` package [45] (by Peter R. Wilson), is used with its `addtotoc` option, a “Abstract” entry is added to the table of contents, as a starred chapter if the document class defines `\chapter`, else as a starred section. This is detected and you should add a `\mtcaddchapter[]` or a `\mtcaddsection[]` command after your `abstract` environment.
- If the `sectsty` package [32] (by Rowland McDonnell) is used, it must be loaded *before* the `minitoc` package.
- If you try to insert empty mini-tables, the `hints` option gives a global warning (except if you used also the `nocheckfiles` option, see section 1.2.3 on page 20).

\firstpartis \firstchapteris \firstsectionis

- If you use one of the obsolete commands (`\firstpartis`, `\firstchapteris`, or `\firstsectionis`), a warning is issued for each use, of course, but also a global hint as reminder.

[14]The commands containing the “@” character in their names are internal commands of LaTeX, of a package or of a class; they are sometimes altered by another packages; reconsider then the loading order of the packages.

[15]The `titlesec` package redefines the sectionning commands in a way completely alien to the standard LaTeX way; hence `minitoc` and `titlesec-titletoc` are fundamentaly incompatible, and it is very sad.

Table 1.11: Extensions of the auxiliary files

mini-table	**long extensions** (UNIX, etc.)	**short extensions** (MS-DOS, etc,)
parttoc	`.ptc`$\langle N\rangle$	`.P`$\langle N\rangle$
partlof	`.plf`$\langle N\rangle$	`.G`$\langle N\rangle$
partlot	`.plf`$\langle N\rangle$	`.U`$\langle N\rangle$
minitoc	`.mtc`$\langle N\rangle$	`.M`$\langle N\rangle$
minilof	`.mlf`$\langle N\rangle$	`.F`$\langle N\rangle$
minilot	`.mlf`$\langle N\rangle$	`.T`$\langle N\rangle$
secttoc	`.stc`$\langle N\rangle$	`.S`$\langle N\rangle$
sectlof	`.slf`$\langle N\rangle$	`.H`$\langle N\rangle$
sectlot	`.slf`$\langle N\rangle$	`.V`$\langle N\rangle$

1.6 Usage with MS-DOS

Under MS-DOS (and other PC oriented old operating systems), the filename extensions are limited to 3 characters. The `minitoc` package determines dynamically the type of extensions available and will use it. All other modifications will be done automatically. The `.mtc`$\langle N\rangle$ extensions will become `.M`$\langle N\rangle$, where $\langle N\rangle$ is the absolute chapter number. The extensions `.mlf`$\langle N\rangle$ and `.mlt`$\langle N\rangle$ become `.F`$\langle N\rangle$ and `.T`$\langle N\rangle$. The `.ptc`$\langle N\rangle$ extensions become `.P`$\langle N\rangle$, where $\langle N\rangle$ is the absolute part number. The extensions `.plf`$\langle N\rangle$ and `.plt`$\langle N\rangle$ become `.G`$\langle N\rangle$ and `.U`$\langle N\rangle$. The `.stc`$\langle N\rangle$ extensions become `.S`$\langle N\rangle$, where $\langle N\rangle$ is the absolute section number. The extensions `.slf`$\langle N\rangle$ and `.slt`$\langle N\rangle$ become `.H`$\langle N\rangle$ and `.V`$\langle N\rangle$. All these extensions are listed in table 1.11. Of course, this implies a limit of 99 chapters in a document, but do you really need so many chapters (or sections in an article)? The limit of 99 parts does not seem too serious for most documents, but for sections, it could be tragical. The `hints` option (section 1.5 on page 38) will report such situations. See also section 2.5 on page 44.

1.7 Why several LaTeX runs are required?

The mini-tables, at part, chapter and section levels, are using some space on the first pages on each chapter, part or section, thus the page numbers are altered. After the first LaTeX run, the mini-tables and lists, partial tables and lists and section-level tables and lists will be empty (in fact skipped since version #35); after the second run, they appear (if not empty), but because they modify the page numbering, page numbers are wrong; after the third LaTeX run, the mini, part- and section-level tables and lists should be correct (see figure 2.1 on page 44).

1.8 The mtcoff package

If a document has been prepared with the minitoc package, it contains many minitoc specific commands, most of them being \dominitoc, \faketableofcontents, and \minitoc commands (and their equivalents for lists of figures and tables). If you want to typeset this document without any mini-table, you have just to replace the minitoc package by the mtcoff package (without option), and all these commands will be ignored, eventually writing warning messages in the .log file. At least two LaTeX runs will be necessary to get a correct page numbering and cross references. It also sanitizes the .aux, .toc, .lof, and .lot files from minitoc specific spurious commands.

Chapter 2

Frequently Asked Questions

Contents

Figures

Here is a list of problems and frequently asked questions about minitoc.sty. If the version has a number less than 43, please upgrade to version #43. This list is also given in the minitoc.bug file, in pure text form.

If a problem arises, it is often wise to: a) use the hints option (see section 1.5 on page 38), which is activated by default, and b) read the ⟨*document*⟩.log file, which may contain pertinent messages.

2.1 How to avoid a page break near the rules before and after the mini-table?

\enlargethispage

This problem seemed solved since version #8, but version #12 adds better fixes. You may have to make some final tuning with \enlargethispage. See the LaTeX manual [27]. The needspace [43] package may also be useful.

2.2 How about implementing others layouts for a mini-table?

Suggestions are welcome, but look at the section 1.3.13 on page 31.

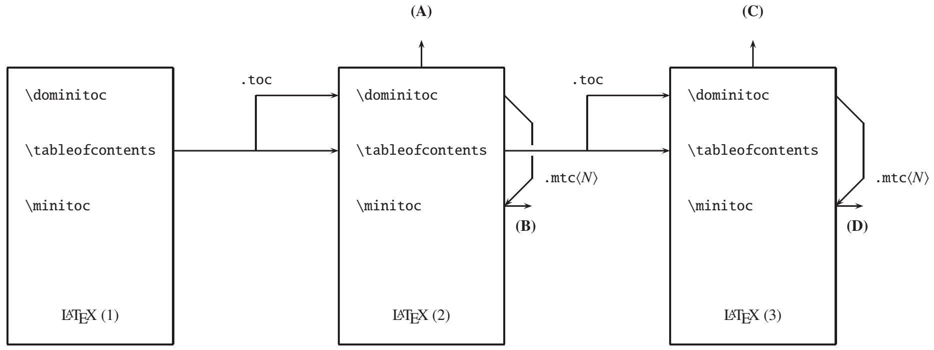
2.3 Two consecutive backslashes in a contents line make an error

Use \protect\linebreak.

2.4 Reordering chapters makes havoc

If you reorder chapters, havoc follows... mini-tables going in wrong chapters.

The best way seems to make one run with the mtcoff package replacing the minitoc package, then restore the minitoc package and re-execute LaTeX three times (yes, it is time consuming...). See figure 2.1 on the next page. Running with the mtcoff package



(A) `\tableofcontents` produces a table of contents, which is likely inaccurate.

(B) `\minitoc` produces minitocs, which are likely inaccurate.

(C) `\tableofcontents` produces a table of contents, which is accurate.

(D) `\minitoc` produces minitocs, which are accurate.

Figure 2.1: Three compilations for `minitoc`

ensures that the standard auxiliary files are cleared from "spurious" commands introduced by `minitoc`. A more radical solution is to delete the `.aux`, `.toc`, `.lof` and `.lot` files relative to the document, then re-execute LaTeX three times.

2.5 Extensions for the names of auxiliary files

This package creates auxiliary files with extensions like `.mtc⟨N⟩`*. Some operating systems allow only 3 characters extensions. What to do?*

No modification is needed: all became automatic since version #28! If you insist to use 3 characters extensions, even on operating systems allowing more, just use the package option `shortext`. Then you will get first the auto-configuration messages, then a message saying that you will use short extensions. But be careful to not have more than 99 mini-tables of the same kind (even empty)!

2.6 Playing with the chapter number

Do not cheat with the "`chapter`" counter, *i.e.* do not write ugly things like

```
\setcounter{chapter}{6}
```

The mechanism would break. It is better to add `\chapter` commands, to create empty (but numbered in a legal way) chapters. Since version #10, the `minitoc` package works with appendices. Version #19 allows to begin with a chapter other that number 1. And look at “Special Entries for TOC, LOF, LOT, Bibliography and Index”, section 1.4.5 on page 37.

The same remark applies to the part and section counters.

2.7 Supported document classes

The minitoc package is restricted to document classes which define chapters in the standard way, like ‘book’ and ‘report’, or sections in the standard way, like ‘article’. There are “parttocs” if the document class defines the \part command. Note that classes like “letter”, which have not the classical sectionning structure, cannot be supported. Classes using sectionning commands with other names are not supported[1]. See also section 2.24 on page 50.

2.8 Compatibility with LaTeX versions

Some users have failed to make minitoc to work. They got a message like:

```
Undefined command ... \@inputcheck ...
Your version of latex.tex is obsolete. Trying to continue...
```

or:

```
Undefined command ... \reset@font ...
Your version of latex.tex is very obsolete.
Trying to continue... crossing fingers.
```

The \reset@font command has been added to latex.tex on September 29th, 1991 and the \@inputcheck command on March 18th, 1992 and this version of latex.tex has been released on March 25th, 1992. If you get this message, you have an old version of latex.tex. Get a recent one from the archives and regenerate a latex.fmt format via initex (or your configuration tool).

2.9 Other mini-tables

Some demanding users want to have minilof, minilot and minibbl. First, “minibbl” is another problem, strongly related to the BibTeX’s dealing with .aux files. Look at

[1]This would be very difficult: any user can create new sectionning commands (often with the help from some packages) with standard or new names; this is only limited by the imagination. The minitoc package relies on the names of the standard sectionning commands and on the syntax of these commands.

the chapterbib, bibunits, multibib, and bibtopic packages. Version #13 has implemented basic minilofs and minilots. Minibbls are not the aim of this package (see http://www.tex.ac.uk/cgi-bin/texfaq2html?label=multbib).

2.10 Why so many auxiliary files?

This package creates a lot of auxiliary files and some users have argued that it is too many. A deep redesign would be necessary to avoid that. Using only one big auxiliary file (or one for all minitocs, one for all minilofs, ...) would make the reading of such file very slow, as it would be read for each \miniXXX macro! Moreover, this would make the checkfiles (see section 1.2.3 on page 20) package option impractical to implement. Note that the many files *.mtc*, etc., may be deleted after the LaTeX run. They are rebuilt by the \dominitoc command (and siblings). But, since version #35, minitoc is able to detect and skip empty *.mtc* files (and siblings) to avoid ugly titles with just two thin rules. It would not be easy to do with one big auxiliary file.

These files contain the mini-tables extracted from the .toc, .lof, and .lot files. They are no more useful after the LaTeX run. If you run LaTeX via a script or a "makefile", it may be useful to add to it a cleaning feature (which should be optional, to allow debugging). The table 1.11 on page 40 gives the list of the extensions for these files (note that a ⟨*document*⟩.mtc auxiliary file is also created as a scratch file).

As an example, you can look at the rubber script [5] (written in Python) by Emmanuel Beffara:

```
http://rubber.sourceforge.net/
http://www.pps.jussieu.fr/~beffara/soft/rubber/
```

2.11 How to do mini-tables at levels other than chapter?

Here also, some redesign was needed. From version #15, there are parttocs, partlofs and partlots for the part level in book|report-like and article-like documents, secttocs, sectlofs and sectlots for the section level in article-like documents. Note that you can not have minitocs features at chapter and section level in the same document, because doing so would make an unreadable monster. The user must choose the main class of the document according to the size of it (e.g. do not write an article of more than 100 sections: this is a report, or even a book!).

	part	chapter	section
book	*	*	
report	*	*	
article	*		*

2.12 Incompatibility with LaTeX2.09

\protect \contentsline The more recent version of LaTeX 2ε adds `\protect` before `\contentsline` in the `.toc`, `.lof` and `.lof` files. The version #17 of `minitoc` attempts to be compatible with LaTeX 2ε and LaTeX2.09. This will be the *last* version usable with LaTeX2.09. Versions #18 and later are LaTeX 2ε specific, and no more compatible with LaTeX2.09, which is completely obsolete.

2.13 Documents resetting the chapter number at each part

Since version #23, `minitoc` works with document classes resetting chapter (or section) number at each part (or chapter). This is possible because the auxiliary files for the mini-tables have now an *absolute* number.

2.14 The mini-tables have too much spaced lines

From version #29, you can have tight mini-tables with the `tight` option, and with the `k-tight` option for the Koma-script classes (since version #43).

2.15 The secttocs are wrong

Secttocs did not work: corrected (version #38).

2.16 How to remove the lines of dots?

The lines of dots (leaders) between section titles and page numbers are removed by the `undotted` option (#29). See also section 1.3.13 on page 31.

2.17 How to use the `hyperref` package with `minitoc`?

Since version #31, `minitoc` works correctly with the powerful `hyperref` package [39], thanks to Heiko Oberdiek, who used the work of Bernd Jaehne and Didier Verna. If you add the loading of the `hyperref` package to a document yet using `minitoc`, you will get error message about spurious closing braces. Just let finish the LaTeX run, then re-LaTeX the document. There will be no problem if you remove the loading of `hyperref` and add it again: the problem occurs only when upgrading from `minitoc` #30 to `minitoc` #31 (or higher) with a document already processed and adding `hyperref` at the same time! It seems better to process the document with `minitoc` #31 (or higher) without `hyperref`, then with `hyperref`, because some internal commands written into the auxiliary files have been modified. If used, the `hyperref` package must be loaded *before* `minitoc`. Note that `(f)minitoc.dtx` show a (not so) basic example of the use of the `hyperref` package with `minitoc`.

2.18 Problem while upgrading `minitoc`

If upgrading from version #30 or lower to version #31 or higher, you should delete the `.aux`, `.toc`, `.lof`, `.lot` files of the document, else the first LaTeX run with version #31 or higher will produce a lot of errors (the next run should be ok). See also the section 2.17.

2.19 A local table of contents for the set of appendices

`\doparttoc` `\tableofcontents` `\appendix` `\part` `\parttoc` `\addtocontents` `\protect` `\setcounter` `\chapter` `\partbegin`

Some users need a table of contents for the appendices, but without putting the entries of it into the main table of contents. The solution is to put the appendices in a `\part` subdivision of the document and ask for a table of contents at the `\part` level:

```
\doparttoc          % after \begin{document}
. . .
\tableofcontents . . .
\appendix
\part{Appendices}   % create a part level subdivision
\parttoc            % create a local table of contents
% To suppress the appendix part in the main toc
\addtocontents{toc}{\protect\setcounter{tocdepth}{-1}}
\chapter{First appendix}
. . .
% Add this at the end of appendices if there is something
% after the appendices (like an index or a bibliography)
% to put a bound to the contents of \parttoc
\addtocontents{toc}{\protect\partbegin}
```

See also section 2.25 on page 51.

2.20 Use with the `appendix` package

`appendices` `\addcontentsline` `\adjustmtc` `\adjuststc`

If you use the `appendix` package [46] (by Peter R. Wilson), you will observe a serious problem with minitocs in the `appendices` environment (and after it): they do not match with their appendix. In fact, the environnement opening `\begin{appendices}` hides a `\addcontentsline` command for a chapter or a section, putting trouble in the numbering of minitocs or secttocs. Two solutions are available. The first one is to add a `\adjustmtc` or `\adjuststc` command (depending if the appendices are at the chapter or section level) after *each* `\begin{appendices}` command. The other solution is to add the following commands in the preamble *after* the loading of the `appendix` package:

```
\let\oldappendices\appendices
\def\appendices{\oldappendices\adjustmtc}
```

if appendices are at the chapter level, OR:

```
\let\oldappendices\appendices
\def\appendices{\oldappendices\adjuststc}
```

if appendices are at the section level.

2.21 Use with the `tocloft` package

`\mtcsetfont`

(This answer is given in the documentation of the `tocloft` package [44].) The `tocloft` (by Peter R. Wilson) and `minitoc` packages have an unfortunate interaction [2], which fortunately can be fixed. In the normal course of events, when `minitoc` is used in a chaptered document it will typeset section entries in the minitocs in bold font. If `tocloft` is used in conjunction with `minitoc`, then the minitoc section entries are typeset in the normal font, except for the page numbers which are in bold font, while the ToC section entries are all in normal font.

One cure, if you want the minitoc section entries to be all in normal small font, is to put:

```
\renewcommand{\mtcSfont}{\normalfont\small}
```

or:

```
\mtcsetfont{minitoc}{section}{\normalfont\small}
```

[2] Discovered by Lyndon Dudding.

in the preamble.

Otherwise, the cure is the following incantation:

```
\renewcommand{\cftsecfont}{\bfseries}
\renewcommand{\cftsecleader}{\bfseries\cftdotfill{\cftdotsep}}
\renewcommand{\cftsecpagefont}{\bfseries}
```

To have the section entries in both the ToC and the minitocs in bold then put the incantation in the preamble. To have only the minitoc section entries in bold while the ToC entries are in the normal font, put the incantation between the `\tableofcontents` command and the first `\chapter` command.

As `tocloft` is a very powerful and useful package, these cures are worth to be added if you need the benefits of this package.

2.22 Use with the memoir class

The `memoir` class [48] offers basically the functionnalities of the `appendix`, `tocbibind` and `tocloft` packages (this class and these packages have the same author, Peter R. WILSON), hence it has the same problems; see above the available solutions. If your version of the `memoir` class is recent, the syntax of the `\chapter` command is different and the `memoir` class is *no more compatible* with the `minitoc` package, but a patch is inserted to fix the problem.

2.23 There are too many commands for fonts, titles, and depths

`\mtcsetfont` `\mtcsettitlefont` Since version #41, the `\mtcsetfont` and `\mtcsettitlefont` commands are available. You do not need anymore to know `\mtcSSSfont`, `\ptifont`, etc.

`\mtcsettitle` Since version #42, the `\mtcsettitle` command is available. You do not need anymore to know `\mtctitle`, `\slttitle`, etc.

`\mtcsetdepth` Since version #43, the `\mtcsetdepth` command is available. You do not need anymore to know the counters `minitocdepth`, `seclotdepth`, etc.

2.24 Compatibility with the $\mathcal{AMS}$ document classes

`\mtcaddchapter` The `amsart.cls` and `amsproc.cls` document classes are incompatible with `minitoc`.

The `amsbook.cls` document class requires the insertion of commands if you want a list of figures and/or a list of tables:

```
\listoffigures
\mtcaddchapter % added
\listoftables
\mtcaddchapter % added
```

2.25 How to hide some entries from the main table of contents

mtchideinmaintoc
mtchideinmainlof
mtchideinmainlot

It is a problem similar to that of section 2.19 on page 48. An example is having a local table of contents for a chapter (`\minitoc`) whose entries should not appear in the main table of contents. Just use the `mtchideinmaintoc` environment:

```
\chapter{Title}
\begin{mtchideinmaintoc}[level]
\minitoc
\section{sub-title}
...
\end{mtchideinmaintoc}
```

This environment accepts an optional numeric argument, which is the depth of hiding in the main toc (default: `-1`, complete hiding).

Of course, the environments `mtchideinmainlof` and `mtchideinmainlot` are also available, to hide some entries in the main list of figures or of tables.

2.26 How to define your own .mld file?

\mtcsettitle
\mtcselectlanguage

First, you should not directly modify one of the distributed `.mld` files. The simplest way to alter some title is to redefine the corresponding command via `\renewcommand` or better via `\mtcsettitle`. If you really want to have your own `.mld` file, you copy an existing `.mld` file into one with a new name (not the name of a distributed `.mld` file). Then you modify this new `.mld` file and you can use it via `\mtcselectlanguage`. You can always contact me to add this new `.mld` file to the distribution. These remarks apply also to the ⟨*language*⟩`[.mld-.mlo]` pairs of language definition files.

2.27 Use with the abstract package

abstract
\mtcaddchapter
\mtcaddsection
\chapter

If the abstract package [45] (by Peter R. Wilson), is used with its addtotoc option, a "Abstract" entry is added to the table of contents, as a starred chapter if the document class defines \chapter, else as a starred section. This problem is detected by the hints option and you should add a \mtcaddchapter[] or a \mtcaddsection[] command after your abstract environment.

2.28 Use with the sectsty package

If the sectsty package [32] (by Rowland McDonnell) is used, it must be loaded *before* the minitoc package, because it alters (redefines) the sectionning commands. Of course, hints option detects this problem.

2.29 Strange alignment in the minitocs

In minitocs, subsections titles are not aligned with sections, as they are in the main table of contents.

The entries of a table of contents are formatted via internal commands like \l@part, \l@chapter, \l@section, etc.

The "part" and "chapter" levels (and "section" for an article) use specific commands which are somewhat complex for a more elaborated formatting. For the "section" (in the report and book classes) and lower levels, these commands are (book class, book.cls) by default:

```
\newcommand*\l@section{\@dottedtocline{1}{1.5em}{2.3em}}
\newcommand*\l@subsection{\@dottedtocline{2}{3.8em}{3.2em}}
\newcommand*\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
\newcommand*\l@paragraph{\@dottedtocline{4}{10em}{5em}}
\newcommand*\l@subparagraph{\@dottedtocline{5}{12em}{6em}}
```

which will be applied in the main table of contents and in the minitocs. The arguments of \@dottedtocline are

1) the logical depth (which will be compared to tocdepth or minitocdepth).

2) the indentation.

3) the width reserved for the page number.

In the standard `book`, `report` and `article` classes, the dimensions (second and third arguments) are given in "em" units, and this unit depends of the current font. In the main table of contents, the section and subsection entries are witten in the *same* font, hence usually the alignment is correct. But in the minitocs, the section entries are written in a bold font while the subsection entries are written in a non bold font (see table 1.6 on page 26), hence one "em" has different sizes in these two fonts and the alignement is changed.

There are several solutions:

- Redefine the `\l@section` ... `\l@subparagraph` commands to use font independent units (pt, mm, pc, etc.). This redefinition must be performed between `\makeatletter` and `\makeatother`, because these commands have a `@` in their names; you must use `\renewcommand*` to redefine these commands.
- Use the `tocloft` [44] package to change the indentation, with font independent units. But then see section 2.21 on page 49.
- Use the same font for the section and subsection entries in the minitocs, using the `\mtcsetfont` command (see section 1.3.9 on page 30) or redefining the `\mtcSfont`, `\mtcSSfont`, `\mtcSSSfont`, `\mtcPfont` and `\mtcSPfont` commands (see table 1.6 on page 26), or similar.

Chapter 3

Installation

Tables

Installation of the minitoc package (version #43).

This package contains a lot of files. The list of all files is given in minitoc.l. See table 3.1 on page 57.

The files are sorted into "classes" below (a file can appear in more than one class). Each class specifies the function and the placement of its files.

(0) The files minitoc.ins and minitoc.dtx are the basic source files of this package. The file fminitoc.dtx loads minitoc.dtx but selects the french documentation. The language selection is done by using \ifcase ... \or ... \fi constructs.

(1) The files minitoc.sty, mtcoff.sty, and *all* *.mld and *.mlo files are the package itself [1]. mtcpatchmem.sty is a temporary fix for compatibility with the memoir class. They must be *all* installed in a directory where LaTeX 2_{ε} finds the .sty files.

(2) Informative text files:

- INSTALL is a file describing the installation of the package. You are (almost) reading it (but it is shorter).
- minitoc.l contains the list of all files of the minitoc distribution. See table 3.1 on page 57.
- README is a file describing briefly the minitoc package, plus some useful infos.

[1] The large number *.mld files is (partialy) a consequence of the fact that some languages have aliases (or dialects) and hence one *.mld file for each name (a *.mld file can load another one); the english and french languages are evident examples. For some languages, the multiplicity of the *.mld files corresponds to a multiplicity of fonts and/or encodings (chinese, greek, japanese, korean, malayalam, polish, russian, serbian), or even for spelling reforms (german, greek, norsk).

- catalog contains basic infos about the minitoc package (abstract, date, author, version, licence).
- TODO lists some suggested developments of the package, not yet implemented. Comments and suggestions are welcome.

(3) minitoc-ex.tex, mini-art.tex, add.tex, add.bib are example files, to play with.

(4) minitoc.bug, minitoc.sum are plain text documentation: list of problems (faq, see chapter 2 on page 42) and summary of commands (see chapter 4 on page 58).

(5) minitoc.ins, minitoc.ist, mtcglo.ist, minitoc.lan, minitoc.dtx, and minitoc.bib are the source of the documentation in (non perfect) english.

(6) minitoc.dtx, fminitoc.dtx, fminitoc.bib, fminitoc.ist, fminitoc.lan, franc.sty, frbib.sty, frnew.sty, frplain1.bst are the source (and tools) of the documentation in french[2].

(7) minitoc.pdf, minitoc.ps are the documentation in (non perfect) english, in PDF and PostScript formats.

(8) fminitoc.pdf, fminitoc.ps are the documentation in french, in PDF and PostScript formats. The french documentation and source files must not be omitted.

(9) pmk is a shell script[3] to prepare the package and its documentation; you should adapt it to your needs. There are also four partial scripts (to be adapted):

- imk, which prepares the package from minitoc.ins and minitoc.dtx;
- emk, which prepares the english documentation from minitoc.dtx;
- fmk, which prepares the french documentation from fminitoc.dtx and minitoc.dtx;
- rmk, which sorts the files into classes (one directory for each class).

These scripts are currently written in C-shell, but there are very simple, and should be easy to convert in another classic shell.

- The files of (0) must be installed in a directory where LaTeX 2_ε finds .dtx and .ins files.
- The files of (1) must be installed in a directory where LaTeX 2_ε finds .sty files.
- The files of (2), (3), (4), (5), (6) and (9) must be installed in a separate directory, but must not be omitted.
- The files of (7) and (8) must be installed as on-line documentation. Note that the PostScript files of the distribution can be printed recto-verso on some printers.

[2]This seems rather strange. In fact, the english and french documentations are both in the minitoc.dtx file. fminitoc.dtx sets a flag then loads minitoc.dtx; hence the file fminitoc.dtx is much smaller than minitoc.dtx. Thus, minitoc.ins contains also some utility files which are automatically created (some .sty files, minitoc.ist, mtcglo.ist, fminitoc.ist, minitoc.lan, fminitoc.lan). The english and french versions are not word-by-word translations, but they are in parallel in the file minitoc.dtx, and this helps the maintenance. Note that the fminitoc.ins file does not exist.

[3]You can sip a cappuccino while this script is running!

Note that `minitoc.dtx` and `fminitoc.dtx` are (not so trivial) examples of using `minitoc` (with `hyperref`).

Table 3.1: List of files (`minitoc.l`)

class (0) :

–`minitoc.ins`
–`minitoc.dtx`
–`fminitoc.dtx`

class (1) :

–`minitoc.sty`
–`mtcoff.sty`
–`mtcpatchmem.sty`
–`acadian.mld`
–`acadien.mld`
–`afrikaan.mld`
–`afrikaans.mld`
–`american.mld`
–`arab.mld`
–`arabic.mld`
–`armenian.mld`
–`austrian.mld`
–`bahasa.mld`
–`bangla.mld`
–`basque.mld`
–`bicig.mld`
–`brazil.mld`
–`brazilian.mld`
–`breton.mld`
–`british.mld`
–`bulgarian.mld`
–`bulgarianb.mld`
–`buryat.mld`
–`canadian.mld`
–`canadien.mld`
–`castillan.mld`
–`castillian.mld`
–`catalan.mld`
–`chinese1.mld`, and `chinese1.mlo`
–`chinese2.mld`, and `chinese2.mlo`
–`croatian.mld`
–`czech.mld`
–`danish.mld`
–`dutch.mld`
–`english.mld`
–`esperant.mld`
–`esperanto.mld`
–`estonian.mld`
–`ethiopia.mld`
–`ethiopian.mld`
–`farsi1.mld`, and `farsi1.mlo`
–`farsi2.mld`, and `farsi2.mlo`
–`finnish.mld`
–`finnish2.mld`
–`francais.mld`
–`french.mld`
–`frenchb.mld`
–`frenchle.mld`
–`frenchpro.mld`
–`galician.mld`
–`german.mld`
–`germanb.mld`
–`greek.mld`
–`greek-mono.mld`
–`greek-polydemo.mld`
–`greek-polykatha.mld`
–`guarani.mld`
–`hangul1.mld`, and `hangul1.mlo`
–`hangul2.mld`, and `hangul2.mlo`
–`hangul3.mld`, and `hangul3.mlo`
–`hangul4.mld`, and `hangul4.mlo`
–`hanja1.mld`, and `hanja1.mlo`
–`hanja2.mld`, and `hanja2.mlo`
–`hebrew.mld`
–`hungarian.mld`
–`icelandic.mld`
–`interlingua.mld`
–`irish.mld`
–`italian.mld`
–`japanese.mld`, and `japanese.mlo`
–`japanese2.mld`, and `japanese2.mlo`
–`japanese3.mld`, and `japanese3.mlo`
–`japanese4.mld`, and `japanese4.mlo`
–`japanese5.mld`, and `japanese5.mlo`
–`latin.mld`
–`latin2.mld`
–`latvian.mld`
–`letton.mld`
–`lithuanian.mld`
–`lsorbian.mld`
–`magyar.mld`
–`magyar2.mld`
–`malayalam-keli.mld`
–`malayalam-rachana.mld`
–`malayalam-rachana2.mld`
–`mongol.mld`
–`naustrian.mld`
–`ngerman.mld`
–`ngermanb.mld`
–`norsk.mld`
–`nynorsk.mld`
–`polish.mld`
–`polish2.mld`
–`portuges.mld`
–`portuguese.mld`
–`romanian.mld`
–`russian.mld`
–`russianb.mld`
–`russianc.mld`
–`russian2m.mld`
–`russian2o.mld`
–`samin.mld`
–`scottish.mld`
–`serbian.mld`
–`serbianc.mld`
–`slovak.mld`
–`slovene.mld`
–`spanish.mld`
–`spanish2.mld`
–`spanish3.mld`
–`swedish.mld`
–`thai.mld`, and `thai.mlo`
–`turkish.mld`
–`UKenglish.mld`
–`ukraineb.mld`
–`USenglish.mld`
–`usorbian.mld`
–`vietnam.mld`
–`vietnamese.mld`
–`welsh.mld`

class (2) :

–`INSTALL`
–`minitoc.l`
–`README`
–`catalog`
–`TODO`

class (3) :

–`minitoc-ex.tex`
–`mini-art.tex`
–`add.tex`
–`add.bib`

class (4) :

–`minitoc.bug`
–`minitoc.sum`

class (5) :

–`minitoc.ins`
–`minitoc.dtx`
–`minitoc.bib`
–`minitoc.ist`
–`mtcglo.ist`
–`minitoc.lan`

class (6) :

–`minitoc.dtx`
–`fminitoc.dtx`
–`fminitoc.bib`
–`fminitoc.ist`
–`fminitoc.lan`
–`franc.sty`
–`frbib.sty`
–`frnew.sty`
–`frplain1.bst`

class (7) :

–`minitoc.pdf`
–`minitoc.ps`

class (8) :

–`fminitoc.pdf`
–`fminitoc.ps`

class (9) :

–pmk
–imk
–emk
–fmk
–rmk

Chapter 4

Memento

Tables

Table 4.1: Package options

Options	**Default**	**Meaning**
`shortext`	`*NO*`	To use short extensions for auxiliary files.
`loose`, `tight`	`loose`	Spacing of lines in mini-tables.
`k-loose`, `k-tight`	`k-loose`	Spacing of lines in mini-tables (Koma-script classes).
`dotted`, `undotted`	`dotted`	Dotted lines from entries to page numbers.
`insection`	`*NO*`	Keeps floats (figures and tables) from drifting outside of their section. Useful if you use sectlofs/sectlots.
`notoccite`	`*NO*`	Useful if you have `\cite` commands in sectionning titles and use an unsorted bibliographic style.
`hints`, `nohints`	`hints`	Adds hints in the `.log` file. Useful to detect some problems.

Language options are listed in table 1.7 on page 27. Default: `english`.

Table 4.2: General commands

Command	Meaning
\faketableofcontents	Replaces \tableofcontents if you want mini-tables of contents but no main table of contents.
\fakelistoffigures	Replaces \listoffigures if you want mini-lists of figures but no main list of figures.
\fakelistoftables	Replaces \listoftables if you want mini-lists of tables but no main list of tables.
\mtcselectlanguage{*language*}	
	Loads *language*.mld to select a language for mini-tables titles.
\mtcsetdepth{*mini-table*}{*depth*}	
	Changes the depth for some mini-tables.
\mtcsetfeature{*mini-table*}{before|after|pagestyle}{*commands*}	
	Modifies the features for a mini-table.
\mtcsetfont{*mini-table*}{*sectionning-level*}{*font commands*}	
	Redefines a minitoc font command.
\mtcsetformat{*mini-table*}{*parameter*}{*value*}	
	Changes the layout of some mini-tables.
\mtcsetpagenumbers{*mini-table*}{on|off}	
	Activates/inhibits page numbers in some mini-tables.
\mtcsetrules{*mini-table*}{on|off}	
	Activates/inhibits horizontal rules in some mini-tables.
\mtcsettitle{*mini-table*}{*title string*}	
	Changes the title for some mini-tables.
\mtcsettitlefont{*mini-table*}{*font commands*}	
	Changes the font of the title for some mini-tables.
\mtcskip	To add a vertical skip between the mini-tables.
\mtcskipamount	Length of \mtcskip. Default: \bigskipamount.
\tightmtcfalse	Loose mini-tables. Default.
\tightmtctrue	Tight mini-tables.
\ktightmtcfalse	Loose mini-tables. Default. (Koma-script classes).
\ktightmtctrue	Tight mini-tables. (Koma-script classes).
\undottedmtcfalse	Dotted lines in mini-tables (from entry to page number). Default.
\undottedmtctrue	No dotted lines in mini-tables (from entry to page number).

Table 4.3: Commands, part level

Command	Meaning
`\doparttoc[`*x*`]`	Before `\[fake]tableofcontents` if you use `\parttoc`*.
`\dopartlof[`*x*`]`	Before `\[fake]listoffigures` if you use `\partlof`*.
`\dopartlot[`*x*`]`	Before `\[fake]listoftables` if you use `\partlot`*.
`\parttoc[`*x*`]`	After each `\part` command for which a parttoc is needed*.
`\partlof[`*x*`]`	After each `\part` command for which a partlof is needed*.
`\partlot[`*x*`]`	After each `\part` command for which a partlot is needed*.
`\setcounter{parttocdepth}{`*depth*`}`	
	Depth of the following parttocs. Analog to `tocdepth`. Default: 2. Has no action on partlofs and partlots.
or:	
`\mtcsetdepth{parttoc|partlof|partlot}{`*depth*`}`	
	Idem, but can also act on partlofs and partlots.
`\ptcindent`	Left/right indentation of a partial table. Default: `24pt`.
`\ptcfont`	Font command for parttoc. Default: `\small\rmfamily\upshape\mdseries` (article) or: `\normalsize\rmfamily\upshape\mdseries` (book, report).
`\ptcCfont`	Font command for parttoc, chapter entries. Default: `\normalsize\rmfamily\upshape\bfseries`.
`\ptcSfont`	Font command for parttoc, section entries. Default: `\small\rmfamily\upshape\bfseries` (article) or: `\small\rmfamily\upshape\bfseries` (book, report).
`\ptcSSfont`	Font command for parttoc, subsection entries**.
`\ptcSSSfont`	Font command for parttoc, subsubsection entries**.
`\ptcPfont`	Font command for parttoc, paragraph entries**.
`\ptcSPfont`	Font command for parttoc, subparagraph entries**.
`\plffont`	Font for partlof. Default: `\small\rmfamily\upshape\mdseries`.
`\pltfont`	Font for partlot. Default: `\small\rmfamily\upshape\mdseries`.
`\ptctitle`	Title of parttocs. Default: `Table of Contents`.
`\plftitle`	Title of partlofs. Default: `List of Figures`.
`\plttitle`	Title of partlots. Default: `List of Tables`.
`\ptifont`	Font for partXXX titles. Default: `\Large\rmfamily\upshape\bfseries` (article) or: `\LARGE\rmfamily\upshape\bfseries` (book, report).

*: `[`*x*`]` is an optional argument to set the position of the title; the setting is local for the `\partXXX` commands, global for the `\dopartXXX` commands. The values of *x* are: `l` for left (default), `c` for centered, `r` for right, `n` or `e` for no title.
**: defaults like `\ptcfont`.

Table 4.4: Commands, chapter level

Command	**Meaning**
`\dominitoc[`*x*`]`	Before `\[fake]tableofcontents` if you use `\minitoc`[*].
`\dominilof[`*x*`]`	Before `\[fake]listoffigures` if you use `\minilof`[*].
`\dominilot[`*x*`]`	Before `\[fake]listoftables` if you use `\minilot`[*].
`\minitoc[`*x*`]`	After each `\chapter` command for which a minitoc is needed[*].
`\minilof[`*x*`]`	After each `\chapter` command for which a minilof is needed[*].
`\minilot[`*x*`]`	After each `\chapter` command for which a minilot is needed[*].
`\setcounter{minitocdepth}{`*depth*`}`	
	Depth of the following minitocs. Analog to `tocdepth`. Default: 2. Has no action on minilofs and minilots.
or:	
`\mtcsetdepth{minitoc|minilof|minilot}{`*depth*`}`	
	Idem, but can also act on minilofs and minilots.
`\mtcindent`	Left/right indentation of a mini-table. Default: `24pt`.
`\mtcfont`	Font command for minitoc. Default: `\small\rmfamily\upshape\mdseries`.
`\mtcSfont`	Font command for minitoc, section entries. Default: `\small\rmfamily\upshape\bfseries`.
`\mtcSSfont`	Font command for minitoc, subsection entries[**].
`\mtcSSSfont`	Font command for minitoc, subsubsection entries[**].
`\mtcPfont`	Font command for minitoc, paragraph entries[**].
`\mtcSPfont`	Font command for minitoc, subparagraph entries[**].
`\mlffont`	Font for minilof. Default: `\small\rmfamily\upshape\mdseries`.
`\mltfont`	Font for minilot. Default: `\small\rmfamily\upshape\mdseries`.
`\mtctitle`	Title of minitocs. Default: `Contents`.
`\mlftitle`	Title of minilofs. Default: `Figures`.
`\mlttitle`	Title of minilots. Default: `Tables`.
`\mtifont`	Font for miniXXX titles. Default: `\large\rmfamily\upshape\bfseries`.

$*$: `[`*x*`]` is an optional argument to set the position of the title; the setting is local for the `\miniXXX` commands, global for the `\dominiXXX` commands. The values of *x* are: `l` for left (default), `c` for centered, `r` for right, `n` or `e` for no title.

$**$: defaults like `\mtcfont`.

Table 4.5: Commands, section level

Command	Meaning
\dosecttoc[*x*]	Before \[fake]tableofcontents if you use \secttoc*.
\dosectlof[*x*]	Before \[fake]listoffigures if you use \sectlof*.
\dosectlot[*x*]	Before \[fake]listoftables if you use \sectlot*.
\secttoc[*x*]	After each \section command for which a secttoc is needed*.
\sectlof[*x*]	After each \section command for which a sectlof is needed*.
\sectlot[*x*]	After each \section command for which a sectlot is needed*.
\setcounter{secttocdepth}{*depth*}	
	Depth of the following secttocs. Analog to tocdepth. Default: 2. Has no action on sectlofs and sectlots.
or:	
\mtcsetdepth{secttoc|sectlof|sectlot}{*depth*}	
	Idem, but can also act on sectlofs and sectlots.
\stcindent	Left/right indentation of a mini-table. Default: 24pt.
\stcfont	Font command for secttoc. Default: \small\rmfamily\upshape\mdseries.
\stcSSfont	Font command for secttoc, subsection entries**.
\stcSSSfont	Font command for secttoc, subsubsection entries**.
\stcPfont	Font command for secttoc, paragraph entries**.
\mtcSPfont	Font command for secttoc, subparagraph entries**.
\slffont	Font for sectlof. Default: \small\rmfamily\upshape\mdseries.
\sltfont	Font for sectlot. Default: \small\rmfamily\upshape\mdseries.
\stctitle	Title of secttocs. Default: Contents.
\slftitle	Title of sectlofs. Default: Figures.
\slttitle	Title of sectlots. Default: Tables.
\stifont	Font for sectXXX titles. Default: \large\rmfamily\upshape\bfseries.

$*$: [*x*] is an optional argument to set the position of the title; the setting is local for the \sectXXX commands, global for the \dosectXXX commands. The values of *x* are: l for left (default), c for centered, r for right, n or e for no title.
$**$: defaults like \stcfont.

Table 4.6: Commands for horizontal rules

Command	Meaning
\[no]ptcrule	Activates or inhibits rules in parttocs.
\[no]mtcrule	Activates or inhibits rules in minitocs.
\[no]stcrule	Activates or inhibits rules in secttocs.
\[no]plfrule	Activates or inhibits rules in partlofs.
\[no]mlfrule	Activates or inhibits rules in minilofs.
\[no]slfrule	Activates or inhibits rules in sectlofs.
\[no]pltrule	Activates or inhibits rules in partlots.
\[no]mltrule	Activates or inhibits rules in minilots.
\[no]sltrule	Activates or inhibits rules in sectlots.
\mtcsetrules{*mini-table*}{on|off}	
	Activates/inhibits horizontal rules in some mini-tables.

By default, parttocs have no rules; minitocs and secttocs have rules. In articles, parttocs have rules.

Table 4.7: Commands for page numbers

Command	Meaning
\[no]ptcpagenumbers	Activates or inhibits page numbers in parttocs.
\[no]plfpagenumbers	Activates or inhibits page numbers in partlofs.
\[no]pltpagenumbers	Activates or inhibits page numbers in partlots.
\[no]mtcpagenumbers	Activates or inhibits page numbers in minitocs.
\[no]mlfpagenumbers	Activates or inhibits page numbers in minilofs.
\[no]mltpagenumbers	Activates or inhibits page numbers in minilots.
\[no]stcpagenumbers	Activates or inhibits page numbers in secttocs.
\[no]slfpagenumbers	Activates or inhibits page numbers in sectlofs.
\[no]sltpagenumbers	Activates or inhibits page numbers in sectlots.
\mtcsetpagenumbers{*mini-table*}{on|off}	Activates/inhibits page numbers in some mini-tables.

By default, the page numbers are present.

Table 4.8: Specific commands for parttocs and other mini-tables features

Command	Default	Meaning
\beforeparttoc	\cleardoublepage	Action before a parttoc.
\beforepartlof	\cleardoublepage	Action before a partlof.
\beforepartlot	\cleardoublepage	Action before a partlot.
\afterparttoc	\cleardoublepage	Action after a parttoc.
\afterpartlof	\cleardoublepage	Action after a partlof.
\afterpartlot	\cleardoublepage	Action after a partlot.
\thispageparttocstyle	\thispagestyle{empty}	Page style for a parttoc.
\thispagepartlofstyle	\thispagestyle{empty}	Page style for a partlof.
\thispagepartlotstyle	\thispagestyle{empty}	Page style for a partlot.
\beforeminitoc	\empty	Action before a minitoc.
\beforeminilof	\empty	Action before a minilof.
\beforeminilot	\empty	Action before a minilot.
\afterminitoc	\empty	Action after a minitoc.
\afterminilof	\empty	Action after a minilof.
\afterminilot	\empty	Action after a minilot.
\thispageminitocstyle	\empty	Page style for a minitoc.
\thispageminilofstyle	\empty	Page style for a minilof.
\thispageminilotstyle	\empty	Page style for a minilot.
\beforesecttoc	\empty	Action before a secttoc.
\beforesectlof	\empty	Action before a sectlof.
\beforesectlot	\empty	Action before a sectlot.
\aftersecttoc	\empty	Action after a secttoc.
\aftersectlof	\empty	Action after a sectlof.
\aftersectlot	\empty	Action after a sectlot.
\thispagesecttocstyle	\empty	Page style for a secttoc.
\thispagesectlofstyle	\empty	Page style for a sectlof.
\thispagesectlotstyle	\empty	Page style for a sectlot.
\mtcsetfeature{*mini-table*}{before|after|pagestyle}{*commands*}		Modifies the features for a mini-table.

Table 4.9: Preparation and insertion commands

Type	**Phase**	**Level**		
		part	**chapter**	**section**
table of contents	preparation	`\doparttoc`[*p*]	`\dominitoc`[*p*]	`\dosecttoc`[*p*]
	insertion	`\parttoc`[*p*]	`\minitoc`[*p*]	`\secttoc`[*p*]
list of figures	preparation	`\dopartlof`[*p*]	`\dominilof`[*p*]	`\dosectlof`[*p*]
	insertion	`\partlof`[*p*]	`\minilof`[*p*]	`\sectlof`[*p*]
list of tables	preparation	`\dopartlot`[*p*]	`\dominilot`[*p*]	`\dosectlot`[*p*]
	insertion	`\partlot`[*p*]	`\minilot`[*p*]	`\sectlot`[*p*]

Each of these commands accepts one optional argument *p*, which specifies the position of the title of the mini-table. This argument *p* has a global effect for the preparation commands, but local for the insertion commands. It is a letter: `[l]` for left aligned (default), `[c]` for centered, `[r]` for right aligned, `[e]` or `[n]` for empty (no title).

Table 4.10: Adjustment commands

Command	**Meaning**
`\adjustptc`[*n*]	Adjusts (increments) the parttoc counter `ptc` by *n*.
`\adjustmtc`[*n*]	Adjusts (increments) the minitoc counter `mtc` by *n*.
`\adjuststc`[*n*]	Adjusts (increments) the secttoc counter `stc` by *n*.
`\decrementptc`	Adjusts (decrements by 1) the parttoc counter `ptc`.
`\decrementmtc`	Adjusts (decrements by 1) the minitoc counter `mtc`.
`\decrementstc`	Adjusts (decrements by 1) the secttoc counter `stc`.
`\incrementptc`	Adjusts (increments by 1) the parttoc counter `ptc`.
`\incrementmtc`	Adjusts (increments by 1) the minitoc counter `mtc`.
`\incrementstc`	Adjusts (increments by 1) the secttoc counter `stc`.
`\mtcaddpart`[*title*]	Adds the title of a `\part*` in the ToC.
`\mtcaddchapter`[*title*]	Adds the title of a `\chapter*` in the ToC.
`\mtcaddsection`[*title*]	Adds the title of a `\section*` in the ToC.
`\mtcfixindex[chapter|section|part]`	
	Adjusts the entry for index in the ToC.
`\mtcfixglossary[chapter|section|part]`	
	Adjusts the entry for glossary in the ToC.
`\begin{mtchideinmaintoc}`[*depth*] ... `\end{mtchideinmaintoc}`	
	Environment to hide entries in the main table of contents.
`\begin{mtchideinmainlof}`[*depth*] ... `\end{mtchideinmainlof}`	
	Environment to hide entries in the main list of figures.
`\begin{mtchideinmainlot}`[*depth*] ... `\end{mtchideinmainlot}`	
	Environment to hide entries in the main list of tables.

Table 4.11: Obsolete commands

Command	**Meaning**
`\firstpartis`{*N*}	*N* is the number of the first part.
`\firstchapteris`{*N*}	*N* is the number of the first chapter.
`\firstsectionis`{*N*}	*N* is the number of the first section.

Acknowledgments

I need to thank the following peoples [1], for their help, their questions, their interventions in the news groups [2], and/or the packages and classes they wrote:
Hassan Abolhassani, Alex AJ, Achod André Aradian, Donald Arseneau, David Aspinall, Philipp Bachmann, Marin Balgarensky, Benjamin Bayart, Stephan P. von Bechtolsheim, Emmanuel Beffara, Javier Bezos, Laurent Bloch, Johannes Braams, Daniel Bourbonnais, David Carlisle, Steven Douglas Cochran, David B. Cook, Oliver Corff, Prakash Countcham, Serguei Dachian, Adrian Daerr, Ben De Rydt, Lyndon Dudding, Marko Èehaja, Victor Eijkhout, Thomas Esser, Karl F. Everitt, Robin Fairbairns, Ulrike Fischer, Mohammad Ghodsi, Michel Goossens, Boumediene Hamzi, Danny Heap, Thorsten Hein, Florence Henry, Morten Høgholm, Don Hosek, Yufan Hu, Dmitry Ivanov, Bernd Jaehne, Radwan Jalam, Dan Jurafsky, Bil Kleb, Thankmar Kronzucker, Alexej M. Kryukov, Toshiki Kumazawa, Frank Küster, Fabio Lanari, Robert Lange, Dag Langmyhr, Jean-Marc Lasgouttes, Claire Lauvernet, Werner Lemberg, Thomas Leonhardt, Stéphane Lepolozec, Adam Lewenberg, Knut Lickert, Dan Luecking, Anders Lyhne, Marcus Marr, Rowland McDonnell, Yanick Michou, Frank Mittelbach, Lapo Filippo Mori, Michael Morrison, Cuong Nguyen, Julien Nicolas, Heiko Oberdiek, Piet van Oostrum, Palash Baran Pal, Roozbeh Pournader, Sebastian Rahtz, Tony Roberts, Denis Roegel, Jan Michel Rynning, David Samsoen, Uwe Schneider, Dung Ta Quang, Laurent Tordella, Stefan Ulrich, Jari Vaario, Vincent Vaquin, Didier Verna, Sylvain Vesco, Nigel Ward, Staszek Wawrykiewicz, Stephan Webanck, Peter R. Wilson, andTim Wright.

[1] And I apologize to whose I forgot.
[2] Mainly, `fr.comp.text.tex` and `comp.text.tex`.

Part II

Implementation

Table of Contents

Chapter 5

Commented code of `minitoc.sty`

Contents

Figures

5.1 Introduction

This long chapter presents the code of the `minitoc` package and attempts to explain it. Some comments of the original source file [1] are skipped, like the history, because they do not need further examination (they will be put in the change history).

The code is split in sections to make the reading easier, and the sections are sometimes reordered to make the reading easier.

Most of the `minitoc` external commands have `mtc`, `ptc`, `stc`, or one of the mini-table names (`parttoc`,..., `sectlot`) in their names. Most of the `minitoc` internal commands have `@mtc`, `@ptc`, `@stc`, or `parttoc@`,..., `sectlot@` in their names, or a similar convention. The few exceptions should be explicit enough to not conflict with other packages.

5.2 Identification code

The code of `minitoc.sty` starts here:

```
⟨*minitoc⟩
```

`\NeedsTeXFormat`
`\ProvidesPackage`
`\PackageInfo`

This code section identifies the package with its name, version number and date. A trace is written in the `.log` file. This package will not work with LaTeX2.09.

```
\NeedsTeXFormat{LaTeX2e}[1996/06/01]%
\ProvidesPackage{minitoc}%
  [2005/09/16 v43 Package minitoc]
\PackageInfo{minitoc}%
  {*** minitoc package, version 43 ***\@gobble}
```

[1] The source file of version #42. Version #43 includes the conversion of the package to `.dtx-.ins` format. Version #42 has not been distributed because of that.

5.3 A file descriptor to write

\tf@mtc
\newwrite

A file descriptor is needed to write the files containing the mini-tables, it is \tf@mtc.

```
\newwrite\tf@mtc
```

5.4 Indentation and skip

\mtcindent
\mtcskip
\mtcskipamount
\parskip

We define the indentation \mtcindent (both sides) of the mini-tables and the command \mtcskip to make a vertical skip before a mini-table, its value is \mtcskipamount (default: \bigskipamount).

```
\newlength\mtcindent
\newskip\mtcskipamount
\setlength{\mtcskipamount}{\bigskipamount}
\def\mtcskip{\unskip\removelastskip{\parskip=\z@\addvspace{\mtcskipamount}}}
```

Note that \mtcskip uses a local group to avoid the influence of \parskip.

5.5 Tests and flags

We need to declare some flags [2] (via \newif) to detect the loading of some packages or classes and the availability of some commands (this will be used by the hints option (section 5.76 on page 190) or to allow the definition of some minitoc commands).

5.5.1 Use of section-level mini-lists of floats.

\if@mtc@sect@floats@
\dosectlof
\dosectlot

We will check if the commands \dosectlof and \dosectlot are used:

```
\newif\if@mtc@sect@floats@ \@mtc@sect@floats@false
```

[2] Not so many years ago, some authors had a preference for using counters rather than flags, because a flag costs 3 control sequences (\iffoo, \foofalse and \footrue), which use memory. But the number of count registers is limited to 256 in the native TEX engine (much more with ε-TEX, but still limited in number), while memory has become rather cheap today. And a code programmed with flags (\iffoo ... \else ... \fi) is easier to structure than a code programmed with counters, IMHO.

5.5.2 Presence of some packages and classes.

\if@mtc@placeinsLoaded@
\if@mtc@memoirLoaded@
\if@mtc@memoirnew@
\if@mtc@sectstyLoaded@
\if@mtc@sectstyLoaded@a@

We will check if the placeins package is loaded, then if the memoir is loaded (and if it is a recent version), then if the sectsty package is loaded (before or after minitoc).

```
\newif\if@mtc@placeinsLoaded@ \@mtc@placeinsLoaded@false
\newif\if@mtc@memoirLoaded@ \@mtc@memoirLoaded@false
\newif\if@mtc@memoirnew@ \@mtc@memoirnew@false
\newif\if@mtc@sectstyLoaded@ \@mtc@sectstyLoaded@false
\newif\if@mtc@sectstyLoaded@a@ \@mtc@sectstyLoaded@a@false
```

\if@mtc@empty@parttoc@
\if@mtc@empty@partlof@
\if@mtc@empty@partlot@
\if@mtc@empty@minitoc@
\if@mtc@empty@minitoc@
\if@mtc@empty@minilof@
\if@mtc@empty@sectlot@
\if@mtc@empty@sectlof@
\if@mtc@empty@sectlot@

We will check if you have attempted to insert some empty mini-tables:

```
\newif\if@mtc@empty@parttoc@ \@mtc@empty@parttoc@false
\newif\if@mtc@empty@partlof@ \@mtc@empty@partlof@false
\newif\if@mtc@empty@partlot@ \@mtc@empty@partlot@false
\newif\if@mtc@empty@minitoc@ \@mtc@empty@minitoc@false
\newif\if@mtc@empty@minilof@ \@mtc@empty@minilof@false
\newif\if@mtc@empty@minilot@ \@mtc@empty@minilot@false
\newif\if@mtc@empty@secttoc@ \@mtc@empty@secttoc@false
\newif\if@mtc@empty@sectlof@ \@mtc@empty@sectlof@false
\newif\if@mtc@empty@sectlot@ \@mtc@empty@sectlot@false
```

5.5.3 Presence or absence of some sectionning commands

We define and set flags about the presence of the sectionning commands (in fact, the counters associated with these commands).

\if@mtc@part@def@

The part counter:

```
\newif\if@mtc@part@def@ \@mtc@part@def@false
  \@ifundefined{part}{\@mtc@part@def@false}{\@mtc@part@def@true}
```

\if@mtc@chapter@def@

The chapter counter:

```
\newif\if@mtc@chapter@def@ \@mtc@chapter@def@false
  \@ifundefined{chapter}{\@mtc@chapter@def@false}{\@mtc@chapter@def@true}
```

\if@mtc@section@def@

The section counter:

```
\newif\if@mtc@section@def@ \@mtc@section@def@false
  \@ifundefined{section}{\@mtc@section@def@false}{\@mtc@section@def@true}
```

We define and set flags about the absence of the sectionning commands.

\if@mtc@part@undef@

The `part` counter:

```
\newif\if@mtc@part@undef@ \@mtc@part@undef@true
   \@ifundefined{part}{\@mtc@part@undef@true}{\@mtc@part@undef@false}
```

\if@mtc@chapter@undef@

The `chapter` counter:

```
\newif\if@mtc@chapter@undef@ \@mtc@chapter@undef@true
   \@ifundefined{chapter}{\@mtc@chapter@undef@true}{\@mtc@chapter@undef@false}
```

\if@mtc@section@undef@

The `section` counter:

```
\newif\if@mtc@section@undef@ \@mtc@section@undef@true
   \@ifundefined{section}{\@mtc@section@undef@true}{\@mtc@section@undef@false}
```

5.5.4 Flags to check to see if some commands are used

We define a pair of flags for each mini-table type: one for the command itself and one for the preparation command (`\do...`). These flags will be used by the `hints` package option (section 5.76 on page 190).

\if@parttoc@used@
\if@partlof@used@
\if@partlot@used@
\if@doparttoc@used@
\if@dopartlof@used@
\if@dopartlot@used@

For the part level:

```
\newif\if@parttoc@used@ \global\@parttoc@used@false
\newif\if@partlof@used@ \global\@partlof@used@false
\newif\if@partlot@used@ \global\@partlot@used@false
\newif\if@doparttoc@used@ \global\@doparttoc@used@false
\newif\if@dopartlof@used@ \global\@dopartlof@used@false
\newif\if@dopartlot@used@ \global\@dopartlot@used@false
```

\if@minitoc@used@
\if@minilof@used@
\if@minilot@used@
\if@dominitoc@used@
\if@dominilof@used@
\if@dominilot@used@

For the chapter level:

```
\newif\if@minitoc@used@ \global\@minitoc@used@false
\newif\if@minilof@used@ \global\@minilof@used@false
\newif\if@minilot@used@ \global\@minilot@used@false
\newif\if@dominitoc@used@ \global\@dominitoc@used@false
\newif\if@dominilof@used@ \global\@dominilof@used@false
\newif\if@dominilot@used@ \global\@dominilot@used@false
```

\if@secttoc@used@
\if@sectlof@used@
\if@sectlot@used@
\if@dosecttoc@used@
\if@dosectlof@used@
\if@dosectlot@used@

For the section level:

```
\newif\if@secttoc@used@ \global\@secttoc@used@false
\newif\if@sectlof@used@ \global\@sectlof@used@false
\newif\if@sectlot@used@ \global\@sectlot@used@false
\newif\if@dosecttoc@used@ \global\@dosecttoc@used@false
\newif\if@dosectlof@used@ \global\@dosectlof@used@false
\newif\if@dosectlot@used@ \global\@dosectlot@used@false
```

\if@firstpartis@used@
\if@firstchapteris@used@
\if@firstsectionis@used@

We also detect the use of some obsolete commands:

```
\newif\if@firstpartis@used@ \global\@firstpartis@used@false
\newif\if@firstchapteris@used@ \global\@firstchapteris@used@false
\newif\if@firstsectionis@used@ \global\@firstsectionis@used@false
```

5.6 Preparation for the notoccite option

\mtc@hook@beforeinputfile
\if@mtc@notoccite@

We declare a flag for the presence of this option and the new internal command `\mtc@hook@beforeinputfile`, which is a "hook" command (redefinable command) used by this option (requested by Donald Arseneau for his `notoccite` package [2]). See section 1.4.6 on page 38.

```
\newif\if@mtc@notoccite@ \@mtc@notoccite@false
\@ifundefined{mtc@hook@beforeinputfile}%
   {\let\mtc@hook@beforeinputfile\relax}{}
```

5.7 Preparation for the tight and k-tight options

\iftightmtc
\ifktightmtc

We just declare a flag for each of these options; they are set false by default (`loose` and `k-loose` options):

```
\newif\iftightmtc \tightmtcfalse
\newif\ifktightmtc \ktightmtcfalse
```

5.8 Preparation to work with hyperref

\AtBeginDocument
\if@mtc@hyper@used@

This code prepares the interface with the `hyperref` package [39]. A flag is defined, then this preparation is performed in `\AtBeginDocument` if this package is loaded. This action defines some commands for the `hyperref` package.

```
\PackageInfo{minitoc}{*** compatible with hyperref ***\@gobble}
\newif\if@mtc@hyper@used@ \global\@mtc@hyper@used@false
\AtBeginDocument{%
  \@ifpackageloaded{hyperref}{%
    \global\@mtc@hyper@used@true
    \def\toclevel@xpart{1000}%
    \def\toclevel@xchapter{1000}%
    \def\toclevel@xsect{1000}%
    \let\toclevel@starpart\toclevel@part
    \let\toclevel@starchapter\toclevel@chapter
    \let\toclevel@starsection\toclevel@section
    \let\toclevel@starsubsection\toclevel@subsection
    \let\toclevel@starsubsubsection\toclevel@subsubsection
    \let\toclevel@starparagraph\toclevel@paragraph
    \let\toclevel@starsubparagraph\toclevel@subparagraph
  }{}%
}
```

5.9 Check if the sectsty package is loaded, and when

\AtBeginDocument
\if@mtc@sectstyLoaded@
\if@mtc@sectstyLoaded@a@

We must test if the sectsty package [32] is loaded before or after minitoc, so we test when minitoc is loaded and also in a \AtBeginDocument block, when all packages have been loaded. See section 5.76.2.6 on page 204.

```
 \@ifpackageloaded{sectsty}{\@mtc@sectstyLoaded@true}{}
 \AtBeginDocument{\@ifpackageloaded{sectsty}{\@mtc@sectstyLoaded@a@true}{}}
```

5.10 Is the memoir class loaded?

\if@mtc@memoirLoaded@
\if@mtc@memoirnew@
\if@mtcpatchmemoir@

We test if the memoir [48] class is loaded. This class needs some compatibility adjustments or may be incompatible if too recent. In the later case, a patch is inserted (see chapter 7 on page 222).

```
\newif\if@mtcpatchmemoir@ \@mtcpatchmemoir@false
\@ifclassloaded{memoir}%
   {\@mtc@memoirLoaded@true\relax%
   \PackageInfo{minitoc}%
      {*** the memoir class is loaded: compatibility attempted ***\@gobble}}%
   {\@mtc@memoirLoaded@false}
\if@mtc@memoirLoaded@
 \@ifundefined{@m@mchapter}%
   {\@mtc@memoirnew@false\PackageInfo{minitoc}%
      {*** old version of the memoir class ***\@gobble}}
   {\@mtc@memoirnew@true\PackageInfo{minitoc}%
      {*** recent version of the memoir class ***\@gobble}
```

```
    \PackageInfo{minitoc}{*** This version of the memoir class uses \MessageBreak
                              a version of \string\chapter\space which is \MessageBreak
                              incompatible with the minitoc package. \MessageBreak
We try to patch ***\@gobble}%
  \@mtcpatchmemoir@true}
\fi
```

\if@mtcpatchmemoir@

And now the patch:

```
\if@mtcpatchmemoir@
\InputIfFileExists{mtcpatchmem.sty}{}{%
  \Package{Error}{minitoc}
    {*** Unable to patch the memoir class ***}%
    {So it remains incompatible. Sorry.}}
\fi
```

5.11 Testing the emptiness of a file

\mtc@ifmtarg
\mtc@xifmtarg
\mtc@EndWhile
\mtc@WhilePreCondition
\mtc@WhileCondition
\mtc@WhileBody
\mtc@While
\mtc@WhileNext

Some macros for testing if an argument of a macro is empty (taken from the package `ifmtarg` [49], by Peter R. Wilson and Donald Arseneau, and from `while.tip`, by Stephan P. von Bechtolsheim [42]). The group is necessary to keep local the catcode change of “Q”, hence a `\gdef` is needed for `\mtc@ifmtarg`.

```
\begingroup
\catcode`\Q=3
\long\gdef\mtc@ifmtarg#1{%
\mtc@xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}
\long\gdef\mtc@xifmtarg#1#2Q#3#4#5\@nil{#4}
\endgroup
\let\mtc@EndWhile = \fi
\def\mtc@While #1#2#3\mtc@EndWhile{%
  \def\mtc@WhilePreCondition{#1}%
  \def\mtc@WhileCondition{#2}%
  \def\mtc@WhileBody{#3}%
  \mtc@@While
}
\def\mtc@@While{%
  \mtc@WhilePreCondition
  \mtc@WhileCondition
    \def\mtc@WhileNext{%
        \mtc@WhileBody
        \mtc@@While
    }%
  \else
    \def\mtc@WhileNext{}%
  \fi
  \mtc@WhileNext
}
```

\if@mtc@checkfiles
\if@mtc@FE
\if@mtc@LI
\mtc@While
\mtc@Body
\mtc@EndWhile
\mtc@CkFile
\mtc@CkStr
\mtc@Rline
\tf@mtc
\@inputcheck

Some macros to test if a file is empty or not: `\mtc@CkFile{file}` returns `\@mtc@FEtrue` if the file is empty, `\@mtc@FEfalse` if the is file not empty. An inexistent file is empty. A file full of white space (space, tabulation, newline) is empty. Comments are empty.

Note: on a big empty file, the `\mtc@While` loop may be time consuming, but not an eternity (33 s for 10^6 lines on my computer), and the first non-empty line stops the loop. `\jobname.mtc` is used as scratch file. It is erased after use.

```
\newif\if@mtc@LI\@mtc@LItrue
\newif\if@mtc@FE\@mtc@FEtrue
\newif\if@mtc@checkfiles\@mtc@checkfilestrue
\def\mtc@Body{\immediate\read\@inputcheck to
                                        \mtc@Rline\relax
  \ifeof\@inputcheck\relax\@mtc@LIfalse\fi
  \expandafter\ifx\mtc@Rline\par\relax
     \def\mtc@Rline{}
  \else
     \ifeof\@inputcheck\relax\global\@mtc@LIfalse\fi
     \mtc@ifmtarg{\mtc@Rline}{\relax}%
               {\@mtc@FEfalse\@mtc@LIfalse}
  \fi
}
\def\mtc@CkFile#1{%
\@mtc@LItrue\@mtc@FEtrue
\if@mtc@checkfiles
\IfFileExists{#1}{%
\immediate\openin\@inputcheck #1\relax
\mtc@While{}{\if@mtc@LI\relax}%
   {\mtc@Body}%
\mtc@EndWhile}%
{\@mtc@FEtrue}%
\else
\@mtc@FEfalse%
\fi}
\closein\@inputcheck\relax
\def\mtc@CkStr#1{%
\immediate\openout\tf@mtc \jobname.mtc
\immediate\write\tf@mtc{#1}%
\immediate\closeout\tf@mtc
\mtc@CkFile{\jobname.mtc}%
\immediate\openout\tf@mtc \jobname.mtc
\immediate\closeout\tf@mtc}
```

5.12 Internal macros to decrement minitoc counters

\mtc@onebackpart
\mtc@onebackchapter
\mtc@onebacksection
\addtocounter

It is sometimes necessary to decrement a minitoc counter (ptc, mtc or stc) by one. These macros are:

```
\def\mtc@onebackpart{\addtocounter{ptc}{-1}}
\def\mtc@onebackchap{\addtocounter{mtc}{-1}}
\def\mtc@onebacksect{\addtocounter{stc}{-1}}
```

5.13 Patching the `\part` command

\part \mtc@svspart \mtc@svpart \@spart \@part \stepcounter

If the `\part` command is not defined (by the document class, usually), we cannot patch it and a warning is displayed[3]. Else, we patch its two branches, `\@part` (for the unstarred version) or `\@spart` (for the starred version, `\part*`): we add `\stepcounter{ptc}` to increment the parttoc counter `ptc`. See also section 5.51 on page 131.

The code of the next section (section 5.14) is also skipped if `\part` is not defined.

```
\@ifundefined{part}{%
  \PackageWarningNoLine{minitoc}%
     {*** part level macros NOT available ***}
}{%% else undefined part (\part defined)
  \PackageInfo{minitoc}%
     {*** part level macros available ***\@gobble}
  \let\mtc@svspart\@spart
  \def\@spart{\stepcounter{ptc}\mtc@svspart}
  \let\mtc@svpart\@part
  \def\@part{\stepcounter{ptc}\mtc@svpart}
```

5.14 Adding an entry in the TOC for a starred part

\mtcaddpart \mtc@ifmtarg \contentsline \addcontentsline \adjustptc \l@xpart \l@part

To add an entry in the TOC for a starred part, we need the `\mtcaddpart` macro, which has an optional argument, the title of the part as if should appear in the TOC.

By default, this argument is empty. If it is empty (tested via `\mtc@ifmtarg`) or omitted, we add a `\contentsline{xpart}{}...` line in the `.toc` file. If it is not empty, we add a `\contentsline{part}{title...}...` line in the `.toc` file. We always add a `\contentsline{xpart}{}...` line in the `.lof` and `.lot` files. Then we increment the `ptc` counter, via `\adjustptc` (defined in section 5.45 on page 118). Using `xpart` as first argument of `\contentsline` means that `\l@xpart` will be invoked in place of `\l@part` to print the entry in the TOC, but `\l@xpart` uses a huge depth (10000) for this entry, hence it will never be really printed (except if you cheat).

```
\newcommand{\mtcaddpart}[1][]{%
    \mtc@ifmtarg{#1}{\addcontentsline{toc}{xpart}{}}%
                    {\addcontentsline{toc}{part}{#1}}%
```

[3]Document classes with sectionning commands but no `\part` command are likely non standard, hence the warning displayed on the terminal.

```
    \addcontentsline{lof}{xpart}{}%
    \addcontentsline{lot}{xpart}{}%
    \adjustptc}
```

This code terminates (temporarily) the part level commands.

```
}%
```

5.15 Section level macros

\chapter \section

The section level macros are defined if `\chapter` is not defined and `\section` defined, i.e. in document classes like `article`, but not in document classes like `book` or `report`. So we test if `\chapter` is defined and if `\section` is defined, with adequate warnings. If neither are defined, you are in big trouble to use the `minitoc` package with the class of your document.

```
\@ifundefined{chapter}{\PackageInfo{minitoc}%
           {*** chapter level macros NOT available ***\@gobble}%
  \@ifundefined{section}{\PackageInfo{minitoc}%
           {*** section level macros NOT available ***\@gobble}%
    \PackageWarningNoLine{minitoc}%
       {*** no section or chapter level macros available ***
        \MessageBreak
        *** PLEASE VERIFY YOUR MAIN DOCUMENT CLASS ***}}%
  {\PackageInfo{minitoc}%
       {*** section level macros available ***\@gobble}%
```

5.16 Corrections for numbering

\mtc@onebacksect \tableofcontents \listoffigures \listoftables \mtcsv@tableofcontents \mtcsv@listoffigures \mtcsv@listoftables

As the TOC, the LOF and the LOT are considered as (starred) sections, we must decrement the secttoc counter (`stc`) via `\mtc@onebacksect` when the corresponding commands are executed. Hence we patch these commands.

```
    \let\mtcsv@tableofcontents\tableofcontents
    \let\mtcsv@listoffigures\listoffigures
    \let\mtcsv@listoftables\listoftables
    \def\tableofcontents{\mtcsv@tableofcontents\mtc@onebacksect}
    \def\listoffigures{\mtcsv@listoffigures\mtc@onebacksect}
    \def\listoftables{\mtcsv@listoftables\mtc@onebacksect}
```

5.17 Patching the `\section` command

\mtc@svsection
\mtc@svss
\@ssect
\@sect
\section
\stepcounter

If the `\section` command is not defined (by the document class, usually), we cannot patch it and a warning is displayed. Else, we patch its two branches, `\@sect` (for the unstarred version) or `\@ssect` (for the starred version, `\section*`): we add `\stepcounter{stc}` to increment the secttoc counter `stc`, only in the unstarred case (the version #25 has removed a spurious decrementation of this counter).

```
    \let\mtc@svsection\section
    \def\section{\stepcounter{stc}\mtc@svsection}
    \let\mtc@svss\@ssect
```

5.18 Adding an entry in the TOC for a starred section

\mtcaddsection
\mtc@ifmtarg
\contentsline
\adjuststc
\l@xsection
\l@section

To add an entry in the TOC for a starred section, we need the `\mtcaddsection` macro, which has an optional argument, the title of the section as it should appear in the TOC. By default, this argument is empty. If it is empty (tested via `\mtc@ifmtarg`) or omitted, we add a `\contentsline{xsection}{}...` line in the `.toc` file. If it is not empty, we add a `\contentsline{section}{title...}...` line in the `.toc` file. We always add a `\contentsline{xsection}{}...` line in the `.lof` and `.lot` files. Then we increment the `stc` counter, via `\adjuststc` (defined in section 5.54 on page 142). Using `xsection` as first argument of `\contentsline` means that `\l@xsection` will be invoked in place of `\l@section` to print the entry in the TOC, but `\l@xsection` uses a huge depth (10000) for this entry, hence it will never be really printed (except if you cheat).

```
\newcommand{\mtcaddsection}[1][]{%
   \mtc@ifmtarg{#1}{\addcontentsline{toc}{xsection}{}}%
                   {\addcontentsline{toc}{section}{#1}}%
   \addcontentsline{lof}{xsection}{}%
   \addcontentsline{lot}{xsection}{}%
   \adjuststc}
```

This code terminates (temporarily) the section level commands, and we continue with chapter level macros.

```
 }}%
  {%
```

5.19 Chapter level macros

\chapter

The chapter level macros are defined if \chapter is defined, *i.e.* in document classes like book or report. So we test if \chapter is defined, with adequate warnings. The test is already done above, we are in the “else” branch of \@ifundefined{chapter}.

```
\PackageInfo{minitoc}{*** chapter level macros available ***\@gobble}
```

5.20 Patching the \chapter command

\chapter \@chapter \mtc@svchapter \stepcounter

The \chapter command is defined (by the document class, usually). We patch its two branches, \@chapter (for the unstarred version) or \@schapter (for the starred version, \chapter*): we add call to \stepcounter{mtc} to increment the minitoc counter mtc. Only the unstarred branch (\@chapter) is patched here. The other branch is patched later (section 5.36 on page 103).

```
\let\mtc@svchapter\@chapter
\def\@chapter{\stepcounter{mtc}\mtc@svchapter}
```

5.21 Adding an entry in the TOC for a starred chapter

\mtcaddchapter \mtc@ifmtarg \contentsline \adjustmtc \l@xchapter \l@chapter

To add an entry in the TOC for a starred chapter, we need the \mtcaddchapter macro, which has an optional argument, the title of the chapter as if should appear in the TOC. By default, this argument is empty. If it is empty (tested via \mtc@ifmtarg) or omitted, we add a \contentsline{xchapter}{}... line in the .toc file. If it is not empty, we add a \contentsline{chapter}{title...}... line in the .toc file. We always add a \contentsline{xchapter}{}... line in the .lof and .lot files. Then we increment the mtc counter, via \adjustmtc (defined in section 5.31 on page 91). Using xchapter as first argument of \contentsline means that \l@xchapter will be invoked in place of \l@chapter to print the entry in the TOC, but \l@xchapter uses a huge depth (10000) for this entry, hence it will never be really printed (except if you cheat).

```
\newcommand{\mtcaddchapter}[1][]{%
    \mtc@ifmtarg{#1}{\addcontentsline{toc}{xchapter}{}}%
                    {\addcontentsline{toc}{chapter}{#1}}%
    \addcontentsline{lof}{xchapter}{}%
    \addcontentsline{lot}{xchapter}{}%
    \adjustmtc}
```

This code terminates (temporarily) the chapter level commands, *i.e.* terminates the `\@ifundefined{chapter}` at the beginning of section 5.15 on page 80.

```
}%
```

5.22 Miscellaneous declarations

\newread \mtc@toks \mtc@string \mtc@strut \mtc@strutbox \mtc@hstrut \mtc@hstrutbox \mtc@v \mtc@zrule \mtc@BBR

The `\newread` command must be redeclared as being `\outer` (as Donald Arseneau told me). We need a token register (`\mtc@toks`), a temporary string (`\mtc@string`), struts (two kinds, each one using a box containing an invisible vertical rule), a rule with all dimensions equal to zero (`\mtc@zrule`) and a command discouraging page breaks (`\mtc@BBR`, for "bad break"). For the struts, which are boxes containing an invisible vertical rule, we use "`ex`" units, to follow the current font.

```
\def\newread{\alloc@6\read\chardef\sixt@@n}
\newtoks\mtc@toks
\def\mtc@string{\relax}
\newbox\mtc@strutbox
\setbox\mtc@strutbox=\hbox{\rule[1.8ex]{\z@}{2.5ex}}
\def\mtc@strut{\relax\ifmmode\copy\mtc@strutbox
                      \else\unhcopy\mtc@strutbox\fi}
\newbox\mtc@hstrutbox
\setbox\mtc@hstrutbox=\hbox{\rule[1.ex]{\z@}{1.ex}}
\def\mtc@hstrut{\relax\ifmmode\copy\mtc@hstrutbox
                       \else\unhcopy\mtc@hstrutbox\fi}
\def\mtc@v{\leavevmode\mtc@strut}
\def\mtc@zrule{\rule[\z@]{\z@}{\z@}}
\def\mtc@BBR{\unpenalty\nopagebreak[4]}
```

5.23 Autoconfiguration of extensions

\tf@mtc

This code is a hack to determine if the operating system is able or unable to use long extensions (>3 characters) in file names. We define a file descriptor (`\tf@mtc`) to write files [4]. This code is verbose if long extensions cannot be used, else the messages are only

[4] Note that it is the *only* new file descriptor created by the `minitoc` package. All files written by `minitoc` use this descriptor, or one of the standard descriptors, e.g. for the `.log` file. In fact, `minitoc` writes also in the `.toc`, `.lof` and `.lot` files, but via file descriptors already used by standard commands like `\tableofcontents`, `\listoffigures` and `\listoftables`. We can conclude that `minitoc` itself uses only one file descriptor (or write stream). Some other attempts to make per chapter TOCs have failed by quickly leading to exhaustion of file descriptors (TEX offers only 16 file descriptors for writing), because they called the standard internal `\@starttoc` macro, which invokes `\newwrite`, for each mini-table. As `minitoc` writes into only one file at a time (and in the `.log` file, and in the standard contents files, of course), we can reuse the same file descriptor and avoid this serious problem. The `minitoc` package writes in the contents files when it encounters a major sectionning command (`\part`, `\chapter`, or `\section`), if necessary. It writes into the mini-table auxiliary files only via the mini-table preparing commands (`\doparttoc`, …, `\dosectlot`), once at a time. You do not need a new hammer for each nail.

Phase (time from left to right):		1	2	3
OS with long extensions	\jobname.mtc1	TRUE	TRUE	$*$
	\jobname.mtc		FALSE	
OS with short extensions	\jobname.mtc(1)	TRUE	FALSE	$*$

Figure 5.1: Hack to detect the limitation to short extensions.

written in the `.log` file. The sequencing of these operations is vital. The figure 5.1 shows this sequence. A star ($*$) denotes which file is read in phase 3.

`\if@longextensions@` `\tf@mtc` `\openout` `\write` `\closeout` `\input` `\jobname`

(0) First, a message and a new flag:

```
\PackageInfo{minitoc}%
   {*** Autoconfiguration of extensions ***\@gobble}
\newif\if@longextensions@\@longextensions@false
```

(1) We write "`\@longextensions@true`" in `\jobname.mtc1`. But if the OS has short extensions, the real name of the file will be truncated to `\jobname.mtc`.

```
\immediate\openout\tf@mtc \jobname.mtc1
\immediate\write\tf@mtc{\string\@longextensions@true}
\immediate\closeout\tf@mtc
```

(2) We write "`\@longextensions@false`" in `\jobname.mtc`.

```
\immediate\openout\tf@mtc \jobname.mtc
\immediate\write\tf@mtc{\string\@longextensions@false}
\immediate\closeout\tf@mtc
```

(3) We read `\jobname.mtc1`. But if the OS has short extensions, the real name of the file will be truncated to `\jobname.mtc`.

```
\input{\jobname.mtc1}
```

(4) Hence, the flag is true if we read really from `\jobname.mtc1`, but false if we read from `\jobname.mtc`. The text and the severity of the messages are different.

```
\if@longextensions@
 \PackageInfo{minitoc}%
   {*** Long extensions (Unix-like) will be used ***\@gobble}
 \PackageInfo{minitoc}%
 {==> this version is configured for UNIX-like \MessageBreak
   \space\space\space\space(long extensions) file names\@gobble}%
\else
 \PackageWarningNoLine{minitoc}%
   {*** Short extensions (MSDOS-like) will be used ***sight***}
 \PackageWarningNoLine{minitoc}%
   {==> this version is configured for MSDOS-like
   \MessageBreak \space\space\space\space(8+3) file names}
\fi
```

(5) We erase the contents of the two files (because `\jobname.mtc` is also used later as a scratch file, see section 5.11 on page 77).

```
  \immediate\openout\tf@mtc \jobname.mtc
  \immediate\closeout\tf@mtc
  \immediate\openout\tf@mtc \jobname.mtc1
  \immediate\closeout\tf@mtc
```

5.24 Detecting obsolete versions of LaTeX

`\@inputcheck` `\reset@font`

This code detects old versions of the LaTeX kernel that are no more supported and with which the `minitoc` package can hardly work. The trick is to detect the absence of some internal LaTeX commands, `\@inputcheck` and `\reset@font`. If you get one of these messages, your are in bad luck and should *urgently* update your LaTeX installation, which is rusting since… many years!

```
\@ifundefined{@inputcheck}%
 {\PackageWarningNoLine{minitoc}%
    {Your version of latex.tex is obsolete.\MessageBreak
    Trying to continue..}\newread\@inputcheck\relax}{}
\@ifundefined{reset@font}%
 {\PackageWarningNoLine{minitoc}%
    {Your version of latex.tex is very obsolete.\MessageBreak
    Trying to continue... crossing fingers}%
 \let\reset@font\relax}{}
```

5.25 A macro to make a TOC entry without leaders nor page numbers

`\@undottedtocline`

The (internal) macro `\@undottedtocline` is a modified version of the standard command `\@dottedtocline`. It will be used in customization macros.

```
\def\@undottedtocline#1#2#3#4#5{%
 \ifnum #1>\c@tocdepth \else
 \vskip \z@ plus.2\p@
 {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
  \parindent #2\relax\@afterindenttrue
  \interlinepenalty\@M
  \leavevmode
  \@tempdima #3\relax \advance\leftskip \@tempdima \hbox{}%
  \hskip -\leftskip
   #4\nobreak\hfill \nobreak
         \null\par}%
 \fi}
```

5.26 Default values for the page-number customization macros

\if@mtc@memoirLoaded@

This section defines some customization macros for the presence or absence of page numbers in the mini-tables. But if the `memoir` class [48] is loaded, it does the job. So, we test first `\if@mtc@memoirLoaded@` to use the commands of `memoir` when they are available.

```
\if@mtc@memoirLoaded@
```

\mtcpagenumbers
\nomtcpagenumbers

For entries in minitocs:

```
\def\mtcpagenumbers{%
  \cftpagenumberson{section}
  \cftpagenumberson{subsection}
  \cftpagenumberson{subsubsection}
  \cftpagenumberson{paragraph}
  \cftpagenumberson{subparagraph}}
\def\nomtcpagenumbers{%
  \cftpagenumbersoff{section}
  \cftpagenumbersoff{subsection}
  \cftpagenumbersoff{subsubsection}
  \cftpagenumbersoff{paragraph}
  \cftpagenumbersoff{subparagraph}}
```

\stcpagenumbers
\nostcpagenumbers

For entries in secttocs:

```
\def\stcpagenumbers{%
  \cftpagenumberson{subsection}
  \cftpagenumberson{subsubsection}
  \cftpagenumberson{paragraph}
  \cftpagenumberson{subparagraph}}
\def\nostcpagenumbers{%
  \cftpagenumbersoff{subsection}
  \cftpagenumbersoff{subsubsection}
  \cftpagenumbersoff{paragraph}
  \cftpagenumbersoff{subparagraph}}
```

\ptcpagenumbers
\noptcpagenumbers

For entries in parttocs:

```
\def\ptcpagenumbers{%
  \cftpagenumberson{chapter}
  \cftpagenumberson{section}
  \cftpagenumberson{subsection}
  \cftpagenumberson{subsubsection}
  \cftpagenumberson{paragraph}
  \cftpagenumberson{subparagraph}}
\def\noptcpagenumbers{%
  \cftpagenumbersoff{chapter}
```

```
  \cftpagenumbersoff{section}
  \cftpagenumbersoff{subsection}
  \cftpagenumbersoff{subsubsection}
  \cftpagenumbersoff{paragraph}
  \cftpagenumbersoff{subparagraph}}
```

\mlfpagenumbers \nomlfpagenumbers \slfpagenumbers \noslfpagenumbers \plfpagenumbers \noplfpagenumbers

For entries in minilofs, sectlofs, and partlofs:

```
  \def\mlfpagenumbers{\cftpagenumberson{figure}}
  \def\nomlfpagenumbers{\cftpagenumbersoff{figure}}
  \def\slfpagenumbers{\cftpagenumberson{figure}}
  \def\noslfpagenumbers{\cftpagenumbersoff{figure}}
  \def\plfpagenumbers{\cftpagenumberson{figure}}
  \def\noplfpagenumbers{\cftpagenumbersoff{figure}}
```

\mltpagenumbers \nomltpagenumbers \sltpagenumbers \nosltpagenumbers \pltpagenumbers \nopltpagenumbers

For entries in minilots, sectlots, and partlots:

```
  \def\mltpagenumbers{\cftpagenumberson{table}}
  \def\nomltpagenumbers{\cftpagenumbersoff{table}}
  \def\sltpagenumbers{\cftpagenumberson{table}}
  \def\nosltpagenumbers{\cftpagenumbersoff{table}}
  \def\pltpagenumbers{\cftpagenumberson{table}}
  \def\nopltpagenumbers{\cftpagenumbersoff{table}}
```

Else, minitoc will use its own commands.

```
\else
```

\mtcpagenumbers \nomtcpagenumbers \mlfpagenumbers \nomlfpagenumbers \mltpagenumbers \nomltpagenumbers

First, for minitocs, secttocs and parttocs:

```
 \def\mtcpagenumbers{\let\mtc@pgno\null}
 \def\nomtcpagenumbers{\let\mtc@pgno\relax}
 \def\stcpagenumbers{\let\stc@pgno\null}
 \def\nostcpagenumbers{\let\stc@pgno\relax}
 \def\ptcpagenumbers{\let\ptc@pgno\null}
 \def\noptcpagenumbers{\let\ptc@pgno\relax}
```

\mlfpagenumbers \nomlfpagenumbers \mlfpagenumbers \nomlfpagenumbers \mltpagenumbers \nomltpagenumbers

Then, for minilofs, sectlofs and partlofs:

```
 \def\mlfpagenumbers{\let\mlf@pgno\null}
 \def\nomlfpagenumbers{\let\mlf@pgno\relax}
 \def\slfpagenumbers{\let\slf@pgno\null}
 \def\noslfpagenumbers{\let\slf@pgno\relax}
 \def\plfpagenumbers{\let\plf@pgno\null}
 \def\noplfpagenumbers{\let\plf@pgno\relax}
```

\mltpagenumbers \nomltpagenumbers \mltpagenumbers \nomltpagenumbers \mltpagenumbers \nomltpagenumbers

Then, for minilots, sectlots and partlots:

```
\def\mltpagenumbers{\let\mlt@pgno\null}
\def\nomltpagenumbers{\let\mlt@pgno\relax}
\def\sltpagenumbers{\let\slt@pgno\null}
\def\nosltpagenumbers{\let\slt@pgno\relax}
\def\pltpagenumbers{\let\plt@pgno\null}
\def\nopltpagenumbers{\let\plt@pgno\relax}
\fi
```

\ptcpagenumbers \plfpagenumbers \pltpagenumbers \mtcpagenumbers \mlfpagenumbers \mltpagenumbers \stcpagenumbers \slfpagenumbers \sltpagenumbers

Then the default values are set; page numbers are present:

```
\ptcpagenumbers
\plfpagenumbers
\pltpagenumbers
\mtcpagenumbers
\mlfpagenumbers
\mltpagenumbers
\stcpagenumbers
\slfpagenumbers
\sltpagenumbers
```

5.27 "Features" for the mini-tables

Each kind of mini-table has three "features": a "before" feature, an "after" feature and a "thispagestyle" feature.

A "before" feature is defined by a macro like `\beforeparttoc` which contains code to be executed before any mini-table of a given type: `\beforeparttoc` is executed before each parttoc. Usually such features contains only trivial commands like `\clear[double]page`, or `\empty`.

An "after" feature is analog but its code is executed after each mini-table of a given type.

A "thispagestyle" feature is defined by a macro like `\thispagestyleparttoc` which contains code to define the page style implied by mini-tables of a given type: the command `\thispagestyleparttoc` can be defined as `\thispagestyle{...}`. Usually, the "thispagestyle" feature is only defined for part-level mini-tables, which use page breaks in their `before` and `after` features. For chapter- and section-level mini-tables, the "thispagestyle" feature is usually defined as `\empty`.

We set the default values for the part-level features depending on the presence of the `\chapter` command, as `article`-like documents are different from the `book`- or `report`-like documents for the layout of part-level mini-tables.

\chapter \beforeparttoc \beforepartlof \beforepartlot

If `\chapter` is not defined, the part level mini-tables have no "before" feature (by default):

```
\@ifundefined{chapter}{%
   \let\beforeparttoc\empty
   \let\beforepartlof\empty
   \let\beforepartlot\empty}%
```

\cleardoublepage

But if `\chapter` is defined, they have a `\cleardoublepage` as default "before" feature:

```
   {\let\beforeparttoc\cleardoublepage
    \let\beforepartlof\cleardoublepage
    \let\beforepartlot\cleardoublepage}
```

\beforeminitoc \beforeminilof \beforeminilot

Chapter level mini-tables have no "before" feature (by default):

```
\let\beforeminitoc\empty
\let\beforeminilof\empty
\let\beforeminilot\empty
```

\beforesecttoc \beforesectlof \beforesectlot

Section level mini-tables have no "before" feature (by default):

```
\let\beforesecttoc\empty
\let\beforesectlof\empty
\let\beforesectlot\empty
```

\chapter \afterparttoc \afterpartlof \afterpartlot

If `\chapter` is not defined, the part level mini-tables have no "after" feature (by default):

```
\@ifundefined{chapter}{%
   \let\afterparttoc\empty
   \let\afterpartlof\empty
   \let\afterpartlot\empty}%
```

\cleardoublepage

But if `\chapter` is defined, they have a `\cleardoublepage` as default "after" feature:

```
   {\let\afterparttoc\cleardoublepage
    \let\afterpartlof\cleardoublepage
    \let\afterpartlot\cleardoublepage}
```

\afterminitoc \afterminilof \afterminilot

Chapter level mini-tables have no "after" feature (by default):

```
\let\afterminitoc\empty
\let\afterminilof\empty
\let\afterminilot\empty
```

\aftersecttoc \aftersectlof \aftersectlot

Section level mini-tables have no "after" feature (by default):

```
\let\aftersecttoc\empty
\let\aftersectlof\empty
\let\aftersectlot\empty
```

\thispagestyle \thispageparttocstyle \thispagepartlofstyle \thispagepartlotstyle \thispageminitocstyle \thispageminilofstyle \thispageminilotstyle \thispagesecttocstyle \thispagesectlofstyle \thispagesectlotstyle

By default, all the "thispagestyle" features (at part level) use the `empty` page style. It affects only the first page of the mini-table.

```
\def\thispageparttocstyle{\thispagestyle{empty}}
\def\thispagepartlofstyle{\thispagestyle{empty}}
\def\thispagepartlotstyle{\thispagestyle{empty}}
\def\thispageminitocstyle{\empty}
\def\thispageminilofstyle{\empty}
\def\thispageminilotstyle{\empty}
\def\thispagesecttocstyle{\empty}
\def\thispagesectlofstyle{\empty}
\def\thispagesectlotstyle{\empty}
```

\mtcsetfeature

In section 5.65.8 on page 177, we will define the `\mtcsetfeature` macro which is much easier user interface to set the mini-tables "features".

5.28 Fake tables of contents

\faketableofcontents \fakelistoffigures \fakelistoftables \fake@starttoc

If you don't want a table of contents, but want minitocs, you need to create the `.toc` file, without inserting it into your document. This `\faketableofcontents` command is a stripped off version of the standard command `\tableofcontents`. We define in the same way the commands `\fakelistoffigures` and `\fakelistoftables`, using in fact a stripped version `\fake@starttoc` of `\@starttoc`. But it is nice to reset to zero the `ptc`, `mtc`, and `stc` counters now, if they are defined[5].

```
\def\faketableofcontents{\fake@starttoc{toc}%
    \@ifundefined{c@ptc}{}{\setcounter{ptc}{0}}%
    \@ifundefined{c@mtc}{}{\setcounter{mtc}{0}}%
    \@ifundefined{c@stc}{}{\setcounter{stc}{0}}%
    }
\def\fakelistoffigures{\fake@starttoc{lof}}
\def\fakelistoftables{\fake@starttoc{lot}}
\def\fake@starttoc#1{\begingroup
  \makeatletter
  \if@filesw \expandafter\newwrite\csname tf@#1\endcsname
             \immediate\openout \csname tf@#1\endcsname
             \jobname.#1\relax
  \fi
  \global\@nobreakfalse \endgroup}
```

[5] Remember the infamous "`stc0`" bug.

This code uses the same file descriptors (for writing) than the original commands.

5.29 Depth counters for minilofs and minilots

\AtBeginDocument \c@lofdepth \c@lotdepth

If the counters `lofdepth` and `lotdepth` are defined, we create the necessary new counters: `minilofdepth` and `minilofdepth`. These counters are initialized to 2. This is done after the loading of the packages, in an `\AtBeginDocument` block:

```
\AtBeginDocument{%
 \@ifundefined{c@lofdepth}{}%
   {\newcounter{minilofdepth}\setcounter{minilofdepth}{2}}%
 \@ifundefined{c@lotdepth}{}%
   {\newcounter{minilotdepth}\setcounter{minilotdepth}{2}}%
}
```

5.30 Chapter level commands

From here, we define the chapter-level commands.

\mtc@markboth \@mkboth

First, we memorize the marks (not used today, but...):

```
\global\let\mtc@markboth\markboth
\global\let\@mkboth\markboth
```

5.31 Starred parts, chapters or sections

\addst@rred \addcontentsline \stepcounter \c@ptc \c@mtc \c@stc

We define commands to manage the starred sectionning commands: `\part*`, `\chapter*` and `\section*`. The section-level is different depending if `\chapter` is defined or not. Eventually, a counter is incremented. A contents line is added in the `.toc` file, with the right depth to print it (see `\l@star...` later, in section 5.63 on page 162).

```
\def\addst@rred#1#2{%
  \addcontentsline{toc}{star#1}{#2}%
  \@ifundefined{c@ptc}{}{%
    \expandafter\ifx\csname #1\endcsname\part\relax
      \stepcounter{ptc}%
    \fi
  }%
  \@ifundefined{c@mtc}{}{%
    \expandafter\ifx\csname #1\endcsname\chapter\relax
```

```
      \stepcounter{mtc}%
    \fi
  }%
  \@ifundefined{c@stc}{}{%
    \expandafter\ifx\csname #1\endcsname\section\relax
%%      \@ifundefined{chapter}{\stepcounter{stc}}{}%
      \stepcounter{stc}%
    \fi
  }%
}%
```

\addstarredsection \chapter \addst@rred

If `\chapter` is not defined, we just define `\addstarredsection`:

```
\@ifundefined{chapter}{%
\gdef\addstarredsection#1{\addst@rred{section}{#1}}
}%
```

else we begin to define the stuff for chapter-level commands (the "else" branch of `\@ifundefined{chapter}`):

```
{%
```

\The@mtc \firstchapteris \if@firstchapteris@used@ \adjustmtc \decrementmtc \incrementmtc \themtc \mtc@rule \mlf@rule \mlt@rule \plf@rule \plt@rule \slf@rule \slt@rule \mtcindent

We define now: the internal format of the `mtc` counter (`\The@mtc`), the obsolete command `\firstchapteris` (it just emits a harmless warning), the `mtc` counter (initialized to 0), the `\adjustmtc` command (increments `mtc`, by 1 by default), the `\decrementmtc` command (decrements `mtc` by 1), the `\incrementmtc` command (increments `mtc` by 1), the format of the `mtc` counter (`\themtc`), the counter `minitocdepth`, initialized to 2, for the depth of a minitoc (analog to the standard `tocdepth` counter), the horizontal rules to draw before and after minitocs (`\mtc@rule`), and we copy that definition into analog macros for other kinds of mini-tables. We also set the default value (24pt) of `\mtcindent`, the indentation for minitocs (both sides). The rules are 0.4pt thick. They are defined via `\hrule` to stay in vertical mode for the final `\kern`.

```
\def\The@mtc{\arabic{mtc}}
\def\firstchapteris#1%
  {\PackageWarning{minitoc}%
     {*** \string\firstchapteris \space is an obsolete command ***}%
     \@firstchapteris@used@true}
\newcounter{mtc}
\setcounter{mtc}{0}
\newcommand{\adjustmtc}[1][1]{\addtocounter{mtc}{#1}}
\def\decrementmtc{\addtocounter{mtc}{-1}}
\def\incrementmtc{\addtocounter{mtc}{+1}}
\gdef\themtc{\arabic{mtc}}
\newcounter{minitocdepth}
\setcounter{minitocdepth}{2}
\def\mtc@rule{\kern-3\p@ \hrule \@width\columnwidth \kern2.6\p@}
\let\mlf@rule\mtc@rule
\let\mlt@rule\mtc@rule
```

```
\let\plf@rule\mtc@rule
\let\plt@rule\mtc@rule
\let\slf@rule\mtc@rule
\let\slt@rule\mtc@rule
\mtcindent=24\p@
```

5.32 Font commands for the mini-tables

\mtcfont \mtcSfont \mtcSSfont \mtcSSSfont \mtcPfont \mtcSPfont \mlffont \mltfont \mtifont

We define these commands with full NFSS [28] descriptions. These definitions are effective if `\chapter` is defined. The fonts for titles are also defined here. See also the `\mtcsetfont` macro (section 5.65.2 on page 165) and the `\mtcsettitlefont` macro later (section 5.65.3 on page 167).

```
\def\mtcfont{\small\rmfamily\upshape\mdseries}
\def\mtcSfont{\small\rmfamily\upshape\bfseries}
\let\mtcSSfont\mtcfont
\let\mtcSSSfont\mtcfont
\let\mtcPfont\mtcfont
\let\mtcSPfont\mtcfont
\let\mlffont\mtcfont
\let\mltfont\mtcfont
\def\mtifont{\large\rmfamily\upshape\bfseries}
```

\coffeefont

And `\coffeefont` is used for “coffee breaks” in the `minutes` package [30].

```
\def\coffeefont{\small\rmfamily\slshape\mdseries}
```

5.33 Internal commands to position the mini-table titles

\df@mtitc \df@mtilf \df@mtilt \do@mtitc \do@mtilf \do@mtilt

The commands `\miniXXX` and `\dominiXXX` accept an optional argument to left justify, center, right justify or omit the title of the chapter-level mini-tables. By default, these titles are left justified. The choice made in a `\dominiXXX` command is global and memorized in `\df@mtitc`, `\df@mtilf` or `\df@mtilt`; the choice made in a `\miniXXX` command is local and stored in `\do@mtitc`, `\do@mtilf` or `\do@mtilt`. See the `\minitoc@` macro later (section 5.35.1 on the next page). An empty title needs a vertical correction (Frank Mittelbach).

\c@mti \l@mti \r@mti \e@mti \n@mti

Centering, flushleft, flushright or empty titles:

```
\def\c@mti#1{\null\hfill #1\hfill\null}
\def\l@mti#1{\null #1\hfill\null}
\def\r@mti#1{\null\hfill #1\null}
```

```
\def\e@mti#1{\vspace{-\baselineskip}}
\def\n@mti#1{\vspace{-\baselineskip}}
```

\l@mti \do@mtitc \df@mtitc \do@mtilf \df@mtilf \do@mtilt \df@mtilt

Default: titles on left:

```
\let\do@mtitc\l@mti
\let\df@mtitc\l@mti
\let\do@mtilf\l@mti
\let\df@mtilf\l@mti
\let\do@mtilt\l@mti
\let\df@mtilt\l@mti
```

5.34 The mtc@verse environment

mtc@verse \iftightmtc \ifktightmtc \\ \list \itemsep \itemindent \listparindent \topsep \parsep \mtcindent

Each minitoc is placed inside a mtc@verse environment. This environment is analog to the standard verse environment and hence defined via two commands: \mtc@verse and \endmtc@verse. As it is a list environment, we first define (in a local way) \\, then call \list{} and set some dimensions like \itemsep, \itemindent, \listparindent, \topsep. \parsep is set to zero if the tight option is active (to reduce the spacing of the lines). Both margins are set to \mtcindent. \endmtc@verse terminates the list and discourages a page break.

```
\def\mtc@verse{\let\\=\@centercr
  \list{}{\itemsep\z@
  \itemindent \z@
  \listparindent \itemindent
  \partopsep\z@
  \iftightmtc \parsep\z@ \fi
  \ifktightmtc \parskip\z@ \fi
  \topsep=1ex
  \leftmargin\mtcindent
  \rightmargin\leftmargin}\item[]}
\def\endmtc@verse{\nopagebreak[4]\endlist}
```

5.35 The \minitoc, \minilof, and \minilot commands

These three commands are very similar, with only cosmetic differences.

5.35.1 The \minitoc command

\minitoc \chapter

The \minitoc command must be used after \chapter if you need a minitoc (no automatic minitoc).

\dominitoc \minitoc

This command accepts an optional argument, whose default value has eventually been set earlier by a `\dominitoc` command. The letter "d" represents this default value. `\dominitoc` has itself an optional argument which sets the default value of the optional argument of `\minitoc`. The default value of the optional argument of the `\dominitoc` command is "l". It seems tortuous, but it is simple to use: we have a default behaviour (l) which can be altered globally via the optional argument of `\dominitoc`, or locally via the optional argument of `\minitoc`.

\minitoc \minitoc@ \@ifnextchar

So we define `\minitoc` with an optional argument and its (current) default value, and call the true code in the `\minitoc@` macro (which has one delimited argument); we use the `\@ifnextchar` trick to detect a left bracket for the optional argument:

```
\def\minitoc{\@ifnextchar[{\minitoc@}{\minitoc@[d]}}
```

The real code of `\minitoc` is in `\minitoc@`, which has a mandatory argument (delimited by brackets) specifying the position of the title.

\if@minitoc@used@

First, we set the global flag `\@minitoc@used@true` to note that `\minitoc` has been called (this will be used by a hint later, section 5.76.2.2 on page 198).

```
\def\minitoc@[#1]{%
\global\@minitoc@used@true
```

\@tocfile \if@longextensions

The name of the file containing the minitoc is constructed from `\jobname` and a suffix `\@tocfile`, which is `mtc` (long extensions) or `M` (short extensions) followed by the absolute number of the minitoc.

```
\if@longextensions@
\def\@tocfile{mtc\The@mtc}%
\else
\def\@tocfile{M\The@mtc}%
\fi
```

\mtc@CkFile \if@mtc@FE \if@mtc@empty@minitoc@

Then we test (via `\mtc@CkFile`) the emptiness of this file. A warning is given if the file is empty and a flag is set (a hint will signal that and empty minitoc has been requested).

```
        \mtc@CkFile{\jobname.\@tocfile}
        \if@mtc@FE
        \PackageInfo{minitoc}%
           {\jobname.\@tocfile\space is empty}
        \@mtc@empty@minitoc@true
        \else
```

\thispageminitocstyle

We call `\thispageminitocstyle` to set the page style (by default, this does nothing because, by default, there is no page break before a minitoc). The marks are not treated, because usually there is no new page for a minitoc.

```
        \thispageminitocstyle
%%        \mtc@markboth{\uppercase{\mtctitle}}{\uppercase{\mtctitle}}%
```

\beforeminitoc samepage \do@mtitc \e@mti \n@mti \c@mti \l@mti \r@mti \df@mtic \mtc@CkStr \mtctitle \if@mtc@FE \mtcfont \mtifont \mtc@rule tabular

We call `\beforeminitoc`, then begin a `samepage` environment (to try to discourage page breaks in a minitoc) and look at the position of the title. If the title is empty, the layout is corrected. We print the title with its font (`\mtifont`), then the top rule of the minitoc (if rules are present), using a `tabular` environment (to inhibit a page break between the title and the top rule). The font is set to `\mtcfont`.

```
        \beforeminitoc
        \relax\begin{samepage}%
        \if #1e\let\do@mtitc\e@mti
        \else\if #1n\let\do@mtitc\n@mti
        \else\if #1c\let\do@mtitc\c@mti
        \else\if #1l\let\do@mtitc\l@mti
        \else\if #1r\let\do@mtitc\r@mti
        \else\if #1d\let\do@mtitc\df@mtitc
        \fi\fi\fi\fi\fi\fi
        \mtc@CkStr{\mtctitle}\if@mtc@FE \let\do@mtitc\e@mti\relax\fi
        \raggedright
        \parskip=\z@%
        \reset@font\mtcfont%
        \parindent=\z@%
        \nopagebreak[4]%
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent %%
        \ifx\mtc@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\mtifont\do@mtitc{\mtc@v\mtctitle}\\
        \end{tabular}%
      \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\mtifont\do@mtitc{\mtc@v\mtctitle}\\\hline
        \end{tabular}%
      \fi
```

\mtc@zrule \mtc@BBR \mtcindent mtc@verse

We forbid a page break after the title and the top rule, then set some layout parameters and begin an `mtc@verse` environment:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\\mtc@BBR
        \leftmargin\mtcindent
        \rightmargin\mtcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
        \begin{mtc@verse}%
```

\c@tocdepth \c@minitocdepth \\ \mtc@BBR

We force the effective depth of the mini-table (`\c@tocdepth`) to the required depth (`\c@minitocdepth`), so the printing is done inside the `mtc@verse` environment, where `tocdepth` has been forced to `minitocdepth`, to print only the entries whose level is low enough. then inhibit a page break. The blank line is necessary to avoid a parasite negative indentation.

```
        \c@tocdepth=\c@minitocdepth
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\mtc@pgno
\@dottedtocline
\@undottedtocline
\mtc@hook@beforeinputfile
\mtc@setform
\mtcsetformat
\mtc@strut

We test the presence of leaders and page numbers, then print the minitoc by inputing the minitoc file. But before reading the minitoc file, we must call the hook macro (asked for by Donald Arseneau for his notoccite package [2]) \mtc@hook@beforeinputfile and the macro \mtc@setform which adjusts some layout parameters (defined by the user via \mtcsetformat). We work in a group to keep local some macro redefinitions.

```
\begingroup
  \makeatletter
  \@ifundefined{mtc@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \mtc@setform%
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-\baselineskip}
  \leavevmode\mtc@strut
  \global\@nobreakfalse\endgroup
```

mtc@verse
\mtc@bottom@rule
\\
samepage
\afterminitoc

We close the mtc@verse environment, add the bottomrule (while preventing a page break), then close the samepage environment, and call \afterminitoc. The blank line (\\) is essential.

```
        \end{mtc@verse}%
        \kern0.\baselineskip%
        \nopagebreak[4]\mtc@bottom@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule
        \end{samepage}%
        \par\pagebreak[1]\vspace*{-1ex}\afterminitoc\fi}
```

\mtc@bottom@rule
\mtc@rule

And we define the bottom rule for a minitoc, with some space under the minitoc:

```
\def\mtc@bottom@rule{%
  \ifx\mtc@rule\relax\relax\else
      \vskip -2.5ex
        \rule[2.4\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}\fi}
```

5.35.2 The \minilof command

\minilof

The \minilof command is very similar to the \minitoc command.

\minilof \chapter

The \minilof command must be used after \chapter if you need a minilof (no automatic minilof).

\dominilof \minilof

This command accepts an optional argument, whose default value has eventually been set earlier by a \dominilof command. The letter “d” represents this default value. \dominilof has itself an optional argument which sets the default value of the optional argument of \minilof. The default value of the optional argument of the \dominilof command is “l”. It seems tortuous, but it is simple to use: we have a default behaviour (l) which can be altered globally via the optional argument of \dominilof, or locally via the optional argument of \minilof.

\minilof \minilof@ \@ifnextchar

So we define \minilof with an optional argument and its (current) default value, and call the true code in the \minilof@ macro (which has one delimited argument); we use the \@ifnextchar trick to detect a left bracket for the optional argument:

```
\def\minilof{\@ifnextchar[{\minilof@}{\minilof@[d]}}
```

The real code of \minilof is in \minilof@, which has a mandatory argument (delimited by brackets) specifying the position of the title.

\if@minilof@used@

First, we set the global flag \@minilof@used@true to note that \minilof has been called (this will be used by a hint later, section 5.76.2.2 on page 198).

```
\def\minilof@[#1]{%
\global\@minilof@used@true
```

\@tocfile \if@longextensions@

The name of the file containing the minilof is constructed from \jobname and a suffix \@tocfile, which is mlf (long extensions) or F (short extensions) followed by the absolute number of the minilof.

```
\if@longextensions@%
\def\@tocfile{mlf\The@mtc}%
\else
\def\@tocfile{F\The@mtc}%
\fi
```

\mtc@CkFile \if@mtc@FE \if@mtc@empty@minilof@

Then we test (via \mtc@CkFile) the emptiness of this file. A warning is given if the file is empty and a flag is set (a hint will signal that and empty minilof has been requested).

```
        \mtc@CkFile{\jobname.\@tocfile}
        \if@mtc@FE
        \PackageInfo{minitoc}%
           {\jobname.\@tocfile\space is empty}
        \@mtc@empty@minilof@true
        \else
```

`\thispageminilofstyle`

We call `\thispageminilofstyle` to set the page style (by default, this does nothing because, by default, there is no page break before a minilof). The marks are not treated, because usually there is no new page for a minilof.

```
        \thispageminilofstyle
%%        \mtc@markboth{\uppercase{\mlftitle}}{\uppercase{\mlftitle}}%
```

`\beforeminilof` `samepage` `\do@mtitc` `\e@mti` `\n@mti` `\c@mti` `\l@mti` `\r@mti` `\df@mtic` `\mtc@CkStr` `\mtctitle` `\if@mtc@FE` `\mlffont` `\mtifont` `\mtc@rule` `tabular`

We call `\beforeminilof`, then begin a `samepage` environment (to try to discourage page breaks in a minilof) and look at the position of the title. If the title is empty, the layout is corrected. We print the title with its font (`\mtifont`), then the top rule of the minilof (if rules are present), using a `tabular` environment (to inhibit a page break between the title and the top rule). The font is set to `\mlffont`.

```
        \beforeminilof
        \relax\begin{samepage}%
        \if #1e\let\do@mtilf\e@mti
        \else\if #1n\let\do@mtilf\n@mti
        \else\if #1c\let\do@mtilf\c@mti
        \else\if #1l\let\do@mtilf\l@mti
        \else\if #1r\let\do@mtilf\r@mti
        \else\if #1d\let\do@mtilf\df@mtilf
        \fi\fi\fi\fi\fi\fi
        \mtc@CkStr{\mlftitle}\if@mtc@FE \let\do@mtilf\e@mti\relax\fi
        \raggedright
        \parskip=\z@
        \reset@font\mlffont
        \parindent=\z@
        \nopagebreak[4]%
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent
        \ifx\mlf@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\mtifont\do@mtilf{\mtc@v\mlftitle}\\
        \end{tabular}%
        \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\mtifont\do@mtilf{\mtc@v\mlftitle}\\\hline
        \end{tabular}%
        \fi
```

`\mtc@zrule` `\mtc@BBR` `\mtcindent` `mtc@verse`

We forbid a page break after the title and the top rule, then set some layout parameters and begin an `mtc@verse` environment:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\\mtc@BBR
        \leftmargin\mtcindent
        \rightmargin\mtcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
        \begin{mtc@verse}%
```

\c@tocdepth
\c@minilofdepth
\\
\mtc@BBR

We force the effective depth of the mini-table (`\c@tocdepth`) to the required depth (`\c@minilofdepth`), so the printing is done inside the `mtc@verse` environment, where `tocdepth` has been forced to `minilofdepth`, to print only the entries whose level is low enough. then inhibit a page break. The blank line is necessary to avoid a parasite negative indentation.

```
        \@ifundefined{c@lofdepth}{}{\c@tocdepth=\c@minilofdepth}
        \ifnum\c@tocdepth<1\relax\c@tocdepth=1\fi
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\mtc@pgno
\@dottedtocline
\@undottedtocline
\mtc@hook@beforeinputfile
\mlf@setform
\mtcsetformat
\mtc@strut

We test the presence of leaders and page numbers, then print the minilof by inputing the minilof file. But before reading the minilof file, we must call the hook macro (asked for by Donald Arseneau for his `notoccite` package [2]) `\mtc@hook@beforeinputfile` and the macro `\mlf@setform` which adjusts some layout parameters (defined by the user via `\mtcsetformat`). We work in a group to keep local some macro redefinitions.

```
\begingroup
  \makeatletter
  \@ifundefined{mlf@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \mlf@setform
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-\baselineskip}
  \leavevmode\mtc@strut
  \global\@nobreakfalse\endgroup
```

mtc@verse
\mtc@bottom@rule
\\
samepage
\afterminilof

We close the `mtc@verse` environment, add the bottomrule (while preventing a page break), then close the `samepage` environment, and call `\afterminilof`. The blank line (`\\`) is essential.

```
        \end{mtc@verse}%
        \kern-1.\baselineskip%
        \nopagebreak[4]\mlf@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
        \par\pagebreak[1]\vspace*{-1ex}\afterminilof\fi}%
```

5.35.3 The `\minilot` command

\minilot

The `\minilot` command is absolutely similar to the `\minilof` command:

\minilot
\chapter

The `\minilot` command must be used after `\chapter` if you need a minilot (no automatic minilot).

\dominilot \minilot

This command accepts an optional argument, whose default value has eventually been set earlier by a `\dominilot` command. The letter “d” represents this default value. `\dominilot` has itself an optional argument which sets the default value of the optional argument of `\minilot`. The default value of the optional argument of the `\dominilot` command is “l”. It seems tortuous, but it is simple to use: we have a default behaviour (l) which can be altered globally via the optional argument of `\dominilot`, or locally via the optional argument of `\minilot`.

\minilot \minilot@ \@ifnextchar

So we define `\minilot` with an optional argument and its (current) default value, and call the true code in the `\minilot@` macro (which has one delimited argument); we use the `\@ifnextchar` trick to detect a left bracket for the optional argument:

```
\def\minilot{\@ifnextchar[{\minilot@}{\minilot@[d]}}
```

The real code of `\minilot` is in `\minilot@`, which has a mandatory argument (delimited by brackets) specifying the position of the title.

\if@minilot@used@

First, we set the global flag `\@minilot@used@true` to note that `\minilot` has been called (this will be used by a hint later, section 5.76.2.2 on page 198).

```
\def\minilot@[#1]{%
\global\@minilot@used@true
```

\@tocfile \if@longextensions@

The name of the file containing the minilot is constructed from `\jobname` and a suffix `\@tocfile`, which is `mlt` (long extensions) or `T` (short extensions) followed by the absolute number of the minilot.

```
\if@longextensions@%
\def\@tocfile{mlt\The@mtc}%
\else
\def\@tocfile{T\The@mtc}%
\fi
```

\mtc@CkFile \if@mtc@FE \if@mtc@empty@minilot@

Then we test (via `\mtc@CkFile`) the emptiness of this file. A warning is given if the file is empty and a flag is set (a hint will signal that and empty minilot has been requested).

```
        \mtc@CkFile{\jobname.\@tocfile}
        \if@mtc@FE
        \PackageInfo{minitoc}%
           {\jobname.\@tocfile\space is empty}
        \@mtc@empty@minilot@true
        \else
```

\thispageminilotstyle

We call `\thispageminilotstyle` to set the page style (by default, this does nothing because, by default, there is no page break before a minilot). The marks are not treated, because usually there is no new page for a minilot.

```
        \thispageminilotstyle
%%        \mtc@markboth{\uppercase{\mlttitle}}{\uppercase{\mlttitle}}%
```

\beforeminilot
samepage
\do@mtitc
\e@mti
\n@mti
\c@mti
\l@mti
\r@mti
\df@mtic
\mtc@CkStr
\mtctitle
\if@mtc@FE
\mltfont
\mtifont
\mtc@rule
tabular

We call `\beforeminilot`, then begin a `samepage` environment (to try to discourage page breaks in a minilot) and look at the position of the title. If the title is empty, the layout is corrected. We print the title with its font (`\mtifont`), then the top rule of the minilot (if rules are present), using a `tabular` environment (to inhibit a page break between the title and the top rule). The font is set to `\mltfont`.

```
        \beforeminilot
        \relax\begin{samepage}%
        \if #1e\let\do@mtilt\e@mti
        \else\if #1n\let\do@mtilt\n@mti
        \else\if #1c\let\do@mtilt\c@mti
        \else\if #1l\let\do@mtilt\l@mti
        \else\if #1r\let\do@mtilt\r@mti
        \else\if #1d\let\do@mtilt\df@mtilt
        \fi\fi\fi\fi\fi\fi
        \mtc@CkStr{\mlttitle}\if@mtc@FE \let\do@mtilt\e@mti\relax\fi
        \raggedright
        \parskip=\z@%
        \reset@font\mltfont%
        \parindent=\z@%
        \nopagebreak[4]%
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent
        \ifx\mlt@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\mtifont\do@mtilt{\mtc@v\mlttitle}\\
        \end{tabular}%
      \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\mtifont\do@mtilt{\mtc@v\mlttitle}\\\hline
        \end{tabular}%
      \fi
```

\mtc@zrule
\mtc@BBR
\mtcindent
mtc@verse

We forbid a page break after the title and the top rule, then set some layout parameters and begin an `mtc@verse` environment:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\\mtc@BBR
        \leftmargin\mtcindent
        \rightmargin\mtcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
        \begin{mtc@verse}%
```

\c@tocdepth
\c@minilotdepth
\\
\mtc@BBR

We force the effective depth of the mini-table (`\c@tocdepth`) to the required depth (`\c@minilotdepth`), so the printing is done inside the `mtc@verse` environment, where `tocdepth` has been forced to `minilotdepth`, to print only the entries whose level is low enough. then inhibit a page break. The blank line is necessary to avoid a parasite negative indentation.

```
        \@ifundefined{c@lotdepth}{}{\c@tocdepth=\c@minilotdepth}
        \ifnum\c@tocdepth<1\relax\c@tocdepth=1\fi
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\mtc@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \mlt@setform \mtcsetformat \mtc@strut

We test the presence of leaders and page numbers, then print the minilot by inputing the minilot file. But before reading the minilot file, we must call the hook macro (asked for by Donald Arseneau for his `notoccite` package [2]) `\mtc@hook@beforeinputfile` and the macro `\mlt@setform` which adjusts some layout parameters (defined by the user via `\mtcsetformat`). We work in a group to keep local some macro redefinitions.

```
\begingroup
  \makeatletter
  \@ifundefined{mlt@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \mlt@setform
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-\baselineskip}
  \leavevmode\mtc@strut
  \global\@nobreakfalse\endgroup
```

mtc@verse \mtc@bottom@rule \\ samepage \afterminilot

We close the `mtc@verse` environment, add the bottomrule (while preventing a page break), then close the `samepage` environment, and call `\afterminilot`. The blank line (`\\`) is essential.

```
        \end{mtc@verse}%
        \kern-1.\baselineskip%
        \nopagebreak[4]\mlt@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
        \par\pagebreak[1]\vspace*{-1ex}\afterminilot\fi}%
```

5.36 Patching the `\chapter` command, continued

\l@xchapter \@dottedtocline \l@chapter \xchapter

First, we define `\l@xchapter` which is like `\l@chapter`, but with a huge depth, to inhibit its printing (except if you cheat):

```
\def\l@xchapter{\@dottedtocline{\@M}{1em}{2.3em}}
\def\xchapter{xchapter}
```

\@chapter
\sv@chapter
\addcontentsline
\ignorespaces

Then we patch `\@chapter` (the non-starred branch of `\chapter`) to add pseudo-chapter entries in the LOF and the LOT (these entries will be used by the `\dominiXXX` commands to split the LOF and the LOT into slices).

```
\let\sv@chapter\@chapter
\def\@chapter[#1]#2{\sv@chapter[{#1}]{#2}\relax%
   \addcontentsline{lof}{xchapter}{#1}%
   \addcontentsline{lot}{xchapter}{#1}%
   \ignorespaces}
```

\mtc@schapter
\@schapter
\addtocontents
\chapterbegin
\chapterend

We also patch `\@schapter` (the starred branch of `\chapter`) to add marks in the TOC to delimit chapters; these marks will be used by the `\dominiXXX` commands to take slices from the LOF and the LOT; as they are defined as `\relax`, they should not perturbate other packages.

```
\let\mtc@schapter\@schapter
\def\@schapter{\addtocontents{toc}{\protect\chapterend}\mtc@schapter}
\def\@schapter{\addtocontents{@@@}{\protect\chapterbegin}\mtc@schapter}
\let\chapterbegin\relax
\let\chapterend\relax
```

5.37 The `\addstarred...` commands

\addstarredsection
\addstarredchapter
\addstarredpart
\chapter
\section
\part
\addst@rred

If the command `\chapter` is undefined, we define the command `\addstarredsection` (only if `\section` is defined). If the command `\chapter` is defined, we define the command `\addstarredchapter`. If the command `\part` is defined, we define the command `\addstarredpart`. We use the utility command `\addst@rred` defined in section 5.31 on page 91.

```
\@ifundefined{chapter}%
   {\@ifundefined{section}%
      {}{\def\addstarredsection#1{\addst@rred{section}{#1}}}}%
   {\def\addstarredchapter#1{\addst@rred{chapter}{#1}}}
\@ifundefined{part}%
   {}{\def\addstarredpart#1{\addst@rred{part}{#1}}}
```

5.38 TOC entries without leaders

\@Undottedtocline
\coffeefont

We define two internal macros to format TOC entries without leaders. The macro `\@Undottedtocline` prints no page number, but `\@Undottedtoclinep` prints it.

```
\def\@Undottedtocline#1#2#3#4#5{%
  \ifnum #1>\c@tocdepth \else
```

```
    \vskip \z@ \@plus.2\p@
    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
     \parindent #2\relax\@afterindenttrue
     \interlinepenalty\@M
     \leavevmode
     \@tempdima #3\relax
     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
     {\coffeefont #4}\nobreak
     \nobreak\null
     \par}%
  \fi}
```

\@Undottedtoclinep
\coffeefont

The same but with the page number:

```
\def\@Undottedtoclinep#1#2#3#4#5{%
  \ifnum #1>\c@tocdepth \else
    \vskip \z@ \@plus.2\p@
    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
     \parindent #2\relax\@afterindenttrue
     \interlinepenalty\@M
     \leavevmode
     \@tempdima #3\relax
     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
     {#4}\nobreak
        \hfill
     \nobreak\null
     \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
     \par}%
  \fi}
```

5.39 Mini-tables with or without leaders

\minitoc@
\minilof@
\minilot@
\ifundottedmtc
\@dottedtocline
\@Undottedtoclinep
\sv@minitoc@
\sv@minilof@
\sv@minilot@

This code creates the flag, then patches each mini-table command (its internal part). We alter the commands `\minitoc@`, `\minilof@`, etc., to test the flag `\ifundottedmtc` and, if true, replace locally `\@dottedtocline` by its dotless version `\@Undottedtoclinep`. Of course, we must also test the availability of the `\chapter`, `\part` and `\section` commands, to avoid to define many unnecessary commands.

```
\newif\ifundottedmtc\undottedmtcfalse
\@ifundefined{chapter}{}{%
   \let\sv@minitoc@\minitoc@
   \def\minitoc@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
      \sv@minitoc@[#1]}}%
   \let\sv@minilof@\minilof@
   \def\minilof@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
      \sv@minilof@[#1]}}%
   \let\sv@minilot@\minilot@
```

```
  \def\minilot@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
    \sv@minilot@[#1]}}}
```

\sv@parttoc@ \sv@partlof@ \sv@partlot@ \ifundottedmtc \parttoc@ \partlof@ \partlot@

For the part level:

```
\@ifundefined{part}{}{%
  \let\sv@parttoc@\parttoc@
  \def\parttoc@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
    \sv@parttoc@[#1]}}%
  \let\sv@partlof@\partlof@
  \def\partlof@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
    \sv@partlof@[#1]}}%
  \let\sv@partlot@\partlot@
  \def\partlot@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
    \sv@partlot@[#1]}}}
```

\sv@secttoc@ \sv@sectlof@ \sv@sectlot@ \ifundottedmtc \secttoc@ \sectlof@ \sectlot@

For the section level:

```
\@ifundefined{chapter}{%
  \@ifundefined{section}{}{%
    \let\sv@secttoc@\secttoc@
    \def\secttoc@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
      \sv@secttoc@[#1]}}%
    \let\sv@sectlof@\sectlof@
    \def\sectlof@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
      \sv@sectlof@[#1]}}%
    \let\sv@sectlot@\sectlot@
    \def\sectlot@[#1]{{\ifundottedmtc\let\@dottedtocline\@Undottedtoclinep\fi
      \sv@sectlot@[#1]}}}}{}
```

5.40 The \dominitoc command and its siblings

\dominitoc \dominilof \dominilot \contentsline \chapbegin \starchapter

The three commands \dominitoc, \dominilof and \dominilot are, of course, very similar. They take the \jobname.toc file (resp. the \jobname.lof and \jobname.lot files) produced by the previous LaTeX run and cut it in slices (one slice per chapter or starred chapter) into the \jobname.mtc⟨N⟩ files (resp. the \jobname.mlf⟨N⟩ and \jobname.mlt⟨N⟩ files), using specific lines in the \jobname.toc (resp. \jobname.lof and \jobname.lot) file. These lines are essentially chapter-level entry commands (like \contentsline{chapter}..., \contentsline{xchapter}..., \contentsline{starchapter}..., \chapbegin delimiting chapters in the TOC (or in the LOF or the LOT). Analog part-level lines delimit parts, hence also chapters.

\dominitoc \dominitoc@ \@@dominitoc \if@dominitoc@used@

As \dominitoc has an optional argument, whose default value is "l" (left), it calls \dominitoc@ with a argument delimited by brackets.

The macros are \dominitoc (user interface), which calls \dominitoc@[l] (or with the optional argument of \dominitoc). Then \dominitoc@[l] processes its argument and calls \@@dominitoc. \@@dominitoc calls \@dominitoc (passing \jobname as argument) then close the minitoc file written. \@dominitoc reset to zero the counter of mini-tables, calls \MTC@next#1.toc (where #1 is the value of \jobname, then reset again to zero the counter of mini-tables. Each call to \dominitoc@ (*i.e.* to dominitoc) sets the flag \@dominitoc@used@true. This will be used later for a hint (which detects that you have correctly called \minitoc *after* \dominitoc and that both or neither have been called). See section 5.76.2.2 on page 198. The code is similar for \dominilof and \dominilot.

\dominitoc
\@dominitoc
\MTC@next

The \dominitoc command extracts information from the .toc file and create the minitocs files, with the adequate extension.

```
\def\@dominitoc#1{{%
  \makeatletter
  \setcounter{mtc}{0}
  \MTC@next#1.toc\relax\\}\setcounter{mtc}{0}}
\def\dominitoc{\@ifnextchar[{\dominitoc@}{\dominitoc@[l]}}
```

\dominilof
\@dominilof
\MLF@next

The \dominilof command extracts information from the .lof file and create the minilofs files, with the adequate extension.

```
\def\@dominilof#1{{%
  \makeatletter
  \setcounter{mtc}{0}
  \MLF@next#1.lof\relax\\}\setcounter{mtc}{0}}
\def\dominilof{\@ifnextchar[{\dominilof@}{\dominilof@[l]}}
```

\dominilot
\@dominilot
\MLT@next

The \dominilot command extracts information from the .lot file and create the minilots files, with the adequate extension.

```
%    \begin{macrocode}
\def\@dominilot#1{{%
  \makeatletter
  \setcounter{mtc}{0}
  \MLT@next#1.lot\relax\\}\setcounter{mtc}{0}}
\def\dominilot{\@ifnextchar[{\dominilot@}{\dominilot@[l]}}
```

\if@dominitoc@used@
\df@mtitc
\e@mti
\n@mti
\c@mti
\l@mti
\r@mti
\@@dominitoc

Some code to flag the use of the command and manage the position of the minitoc title:

```
\def\dominitoc@[#1]{%
\global\@dominitoc@used@true
\if #1e\let\df@mtitc\e@mti%
\else\if #1n\let\df@mtitc\n@mti%
\else\if #1c\let\df@mtitc\c@mti%
\else\if #1l\let\df@mtitc\l@mti%
```

```
\else\if #1r\let\df@mtitc\r@mti%
\fi\fi\fi\fi\fi%
\@@dominitoc}
```

\if@dominilof@used@ \df@mtilf \e@mti \n@mti \c@mti \l@mti \r@mti \@@dominilof

Some code to flag the use of the command and manage the position of the minilof title:

```
\def\dominilof@[#1]{%
\global\@dominilof@used@true
\if #1e\let\df@mtilf\e@mti%
\else\if #1n\let\df@mtilf\n@mti%
\else\if #1c\let\df@mtilf\c@mti%
\else\if #1l\let\df@mtilf\l@mti%
\else\if #1r\let\df@mtilf\r@mti%
\fi\fi\fi\fi\fi%
\@@dominilof}
```

\if@dominilot@used@ \df@mtilt \e@mti \n@mti \c@mti \l@mti \r@mti \@@dominilot

Some code to flag the use of the command and manage the position of the minilot title:

```
\def\dominilot@[#1]{%
\global\@dominilot@used@true
\if #1e\let\df@mtilt\e@mti%
\else\if #1n\let\df@mtilt\n@mti%
\else\if #1c\let\df@mtilt\c@mti%
\else\if #1l\let\df@mtilt\l@mti%
\else\if #1r\let\df@mtilt\r@mti%
\fi\fi\fi\fi\fi%
\@@dominilot}
```

\@@dominitoc \@@dominilof \@@dominilot \tf@mtc

These macros invoke the `\@domini...` macros to create the mini-table file, then close the file descriptor.

```
\def\@@dominitoc{\@dominitoc{\jobname}\immediate\closeout\tf@mtc}
\def\@@dominilof{\@dominilof{\jobname}\immediate\closeout\tf@mtc}
\def\@@dominilot{\@dominilot{\jobname}\immediate\closeout\tf@mtc}
```

5.40.1 Analysis and splitting of the TOC file

This is done via a loop managed by the following macros [6]:

\MTC@next \MTC@list \MTC@loop

Processes the next entry in the list and removes it from the head of the list:

[6] This code is derived from the `xr` package [12], by David Carlisle, with his permission. The strings "HO", "DV", and "BJ" in the comments denote modifications made by Heiko Oberdiek, Didier Verna, and Bernd Jaehne for the support of `hyperref`, essentially by adding an argument to some macros, to use the hyperlink argument in the contents lines.

```
\def\MTC@next#1\relax#2\\{%
  \edef\MTC@list{#2}%
  \MTC@loop{#1}%
}
```

\MTC@toc
\MTC@list
\MTC@explist

Check if the list is empty:

```
\def\MTC@toc{%
  \ifx\MTC@list\@empty\else\expandafter\MTC@explist\fi
}
```

\MTC@contentsline
\arabic
\chapter
\themtc
\tf@mtc

The macro \MTC@contentsline analyses the lines read from the TOC file and detects interesting keywords. If \chapter is found, the mtc counter (which simulates the chapter counter, but is absolute) is incremented and a new minitoc file is created.

```
\def\MTC@contentsline#1#2#3#4{% %%HO/BJ: 4 instead of 3 parameters
  \gdef\themtc{\arabic{mtc}}% %%HO: space removed
  \expandafter\ifx\csname #1\endcsname\chapter
    \stepcounter{mtc}%
```

\if@longextensions@
\themtc
\mtcname
\tf@mtc
\closeout
\openout

We test if long or short extensions are used, to build the name of the mini-table file, then open it (after closing the file descriptor):

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.mtc\themtc\@gobble}%
      \def\mtcname{\jobname.mtc\themtc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.M\themtc\@gobble}%
      \def\mtcname{\jobname.M\themtc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\mtcname
  \fi
```

\mtc@toks

Now, we filter the relevant contents lines, the token register \mtc@toks is used as a verbatim memory.

```
  \mtc@toks{\noexpand\leavevmode #2}%
```

\MTC@WriteContentsline
\section
\subsection
\subsubsection
\paragraph
\subparagraph

Each interesting contents line is copied, with a font command added before it. We begin with the standatd sectionning commands, below \chapter:

```
  \expandafter\ifx\csname #1\endcsname\section
    \MTC@WriteContentsline{#1}{mtcS}{#3}{#4}%
```

```
\fi
\expandafter\ifx\csname #1\endcsname\subsection
  \MTC@WriteContentsline{#1}{mtcSS}{#3}{#4}%
\fi
\expandafter\ifx\csname #1\endcsname\subsubsection
  \MTC@WriteContentsline{#1}{mtcSSS}{#3}{#4}%
\fi
\expandafter\ifx\csname #1\endcsname\paragraph
  \MTC@WriteContentsline{#1}{mtcP}{#3}{#4}%
\fi
\expandafter\ifx\csname #1\endcsname\subparagraph
  \MTC@WriteContentsline{#1}{mtcSP}{#3}{#4}%
\fi
```

\coffee
\MTC@WriteCoffeeline

A coffee break contents line is written for `\coffee`:

```
\expandafter\ifx\csname #1\endcsname\coffee
  \MTC@WriteCoffeeline{#1}{#3}%
\fi
```

\starchapter
\stepcounter
\if@longextensions@
\mtcname
\themtc
\tf@mtc
\closeout
\openout

If it is `\starchapter` (for a starred chapter), we increment the `mtc` counter, build a new minitoc file name, close the file descriptor and open it with this new file.

```
\expandafter\ifx\csname #1\endcsname\starchapter
  \stepcounter{mtc}%
  \if@longextensions@
    \PackageInfo{minitoc}%
       {Writing\space\jobname.mtc\themtc\@gobble}%
    \def\mtcname{\jobname.mtc\themtc}%
  \else
    \PackageInfo{minitoc}%
       {Writing\space\jobname.M\themtc\@gobble}%
    \def\mtcname{\jobname.M\themtc}%
  \fi
  \immediate\closeout\tf@mtc
  \immediate\openout\tf@mtc=\mtcname
\fi
```

\starsection
\MTC@WriteContentsline
\starsubsection
\starsubsubsection
\starparagraph
\starsubparagraph

For starred sectionning commands lower than `\chapter`, a contents line is written into the minitoc file, with a font command added:

```
\expandafter\ifx\csname #1\endcsname\starsection
  \MTC@WriteContentsline{#1}{mtcS}{#3}{#4}%
\fi
\expandafter\ifx\csname #1\endcsname\starsubsection
  \MTC@WriteContentsline{#1}{mtcSS}{#3}{#4}%
\fi
\expandafter\ifx\csname #1\endcsname\starsubsubsection
  \MTC@WriteContentsline{#1}{mtcSSS}{#3}{#4}%
\fi
```

```
  \expandafter\ifx\csname #1\endcsname\starparagraph
    \MTC@WriteContentsline{#1}{mtcP}{#3}{#4}%
  \fi
  \expandafter\ifx\csname #1\endcsname\starsubparagraph
    \MTC@WriteContentsline{#1}{mtcSP}{#3}{#4}%
  \fi
}
```

\MTC@explist \MTC@next \MTC@list

The loop to read the lines of the TOC file; it expands the list of entries and call \MTC@next to process the first one:

```
\def\MTC@explist{\expandafter\MTC@next\MTC@list\\}
```

\MTC@loop \openin \@inputcheck \MTC@toc \MTC@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\MTC@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1.\MessageBreak MINITOCS NOT PREPARED}%
    \expandafter\MTC@toc
  \else
    \PackageInfo{minitoc}{PREPARING MINITOCS FROM #1}%
    \expandafter\MTC@read
  \fi
}
```

\MTC@read \read \@inputcheck \MTC@line

Read the next entry of the `.toc` file.

```
\def\MTC@read{%
  \read\@inputcheck to\MTC@line
```

\MTC@test \MTC@line \MTC@

The `.....` make sure that \MTC@test has sufficient arguments:

```
  \expandafter\MTC@test\MTC@line.....\MTC@% %%HO: . added
}%
```

\MTC@test \contentsline \MTC@contentsline \mtc@string \@input \MTC@list \chapterend \closeout \tf@mtc \openout \chapterbegin \addtocounter \MTC@toc \MTC@read

The \MTC@test macro finds the “interesting” commands in the TOC file, mainly to delimit chapters:

```
%%HO/BJ: now patch \MTC@test,
%%HO/BJ: call \MTC@contentsline with 4 instead of 3 parameters
```

Look at the first token of the line. If it is an interesting entry, process it. If it is \@input, add the file to the list. Otherwise ignore. Go around the loop if not at end of file. Finally process the next file in the list.

```
\long\def\MTC@test#1#2#3#4#5#6\MTC@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \MTC@contentsline{#2}{#3}{#4}{#5}%
    %%HO/BJ: 4. parameter added by Tony Roberts
    \let\mtc@string\relax
  \else\ifx#1\@input
    \edef\MTC@list{\MTC@list#2\relax}%
  \else\ifx#1\chapterend
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\chapterbegin
    \addtocounter{mtc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck
    \expandafter\MTC@toc
  \else
    \expandafter\MTC@read
  \fi
}%
```

5.41 Mini-lists of figures

The code is similar to the code for mini-tables of contents, but with less commands to recognize.

5.41.1 Analysis and splitting of the list of figures file

\MLF@next
\MLF@list
\MLF@loop

This is done via a loop managed by the following macros:

Processes the next entry in the list and removes it from the head of the list:

```
\def\MLF@next#1\relax#2\\{%
  \edef\MLF@list{#2}%
  \MLF@loop{#1}}
```

\MLF@lof
\MLF@list
\MLF@explist

Checks if the list is empty:

```
\def\MLF@lof{%
  \ifx\MLF@list\@empty\else\expandafter\MLF@explist\fi}
```

\MLF@contentsline \arabic \xchapter

The macro \MLF@contentsline analyses the lines read from the LOF file and detects interesting keywords. If \xchapter is found, mtc is incremented and a new minilof file is created.

```
\def\MLF@contentsline#1#2#3#4{% %%HO: added #4
  \gdef\themtc{\arabic{mtc}}% %%HO: space removed
  \expandafter\ifx\csname #1\endcsname\xchapter
    \stepcounter{mtc}%
```

\if@longextensions@ \themtc \mlfname \closeout \tf@mtc \openout

The name of the minilof file is build from \jobname and a long or short extension:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.mlf\themtc\@gobble}
      \def\mlfname{\jobname.mlf\themtc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.F\themtc\@gobble}
      \def\mlfname{\jobname.F\themtc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\mlfname
  \fi
```

\figure \mtc@toks \MTC@WriteContentsline

If we found a \figure entry, we copy it into the minilof file:

```
  \expandafter\ifx\csname #1\endcsname\figure
    \mtc@toks{\noexpand\leavevmode#2}%
    \MTC@WriteContentsline{#1}{mlf}{#3}{#4}%
  \fi
}
```

\MLF@explist \MLF@next \MLF@list

The loop to read the LOF file; it expands the list of entries and calls \MLF@next to process the first one:

```
\def\MLF@explist{\expandafter\MLF@next\MLF@list\\}
```

\MLF@loop \openin \@inputcheck \MLF@lof \MLF@read

And now, we scan the .lof file:

```
\def\MLF@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1.\MessageBreak MINILOFS NOT PREPARED}%
    \expandafter\MLF@lof
  \else
    \PackageInfo{minitoc}%
       {PREPARING MINILOFS FROM #1}%
    \expandafter\MLF@read\fi}
```

\MLF@read \read \@inputcheck

Read the next entry in the `.lof` file:

```
\def\MLF@read{%
  \read\@inputcheck to\MLF@line
```

\MLF@line \MLF@test \MLF@

The `.....` make sure that `\MLF@test` has sufficient arguments:

```
  \expandafter\MLF@test\MLF@line.....\MLF@% %%HO: . added
  }%
```

\MLF@test

The `\MLF@test` macro finds the "interesting" commands in the LOF file, mainly to delimit chapters.

\contentsline \mtc@string \MLF@contentsline \@input \MLF@list \chapterend \closeout \tf@mtc \openout \chapterbegin \addtocounter \MLF@lof \MLF@read

Look at the first token of the line. If it is an interesting entry, process it. If it is `\@input`, add the file to the list. Otherwise ignore. Go around the loop if not at end of file. Finally process the next file in the list.

```
\long\def\MLF@test#1#2#3#4#5#6\MLF@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \MLF@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
    \let\mtc@string\relax
  \else\ifx#1\@input
     \edef\MLF@list{\MLF@list#2\relax}%
  \else\ifx#1\chapterend
     \immediate\closeout\tf@mtc
     \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\chapterbegin
    \addtocounter{mtc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\MLF@lof
  \else\expandafter\MLF@read\fi}%
```

5.42 Mini-lists of tables

The code is similar to the code for mini-tables of contents, but with less commands to recognize.

5.42.1 Analysis and splitting of the list of tables file

\MLT@next \MLT@list

This is done via a loop managed by the following macros:

\MLT@loop

Processes the next entry in the list and removes it from the head of the list:

```
\def\MLT@next#1\relax#2\\{%
  \edef\MLT@list{#2}%
  \MLT@loop{#1}}
```

\MLT@lot
\MLT@list
\MLT@explist

Checks if the list is empty:

```
\def\MLT@lot{%
  \ifx\MLT@list\@empty\else\expandafter\MLT@explist\fi}
```

\MLT@contentsline
\arabic
\xchapter

The macro `\MLT@contentsline` analyses the lines read from the LOT file and detects interesting keywords. If `\xchapter` is found, `mtc` is incremented and a new minilot file is created.

```
\def\MLT@contentsline#1#2#3#4{% %%HO: added #4
  \gdef\themtc{\arabic{mtc}}% %%HO: space removed
  \expandafter\ifx\csname #1\endcsname\xchapter
    \stepcounter{mtc}%
```

\if@longextensions@
\themtc
\mltname
\closeout
\tf@mtc
\openout

The name of the minilot file is build from `\jobname` and a long or short extension:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.mlt\themtc\@gobble}%
      \def\mltname{\jobname.mlt\themtc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.T\themtc\@gobble}%
      \def\mltname{\jobname.T\themtc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\mltname
  \fi
```

\table
\mtc@toks
\MTC@WriteContentsline

If we found a `\table` entry, we copy it into the minilot file:

```
  \expandafter\ifx\csname #1\endcsname\table
    \mtc@toks{\noexpand\leavevmode#2}%
    \MTC@WriteContentsline{#1}{mlt}{#3}{#4}%
  \fi
}
```

\MLT@explist
\MLT@next
\MLT@list

The loop to read the LOT file; it expands the list of entries and calls `\MLT@next` to process the first one:

```
\def\MLT@explist{\expandafter\MLT@next\MLT@list\\}
```

\MLT@loop \openin \@inputcheck \MLT@lot \MLT@read

And now, we scan the `.lot` file:

```
\def\MLT@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1.\MessageBreak MINILOTS NOT PREPARED}%
    \expandafter\MLT@lot
  \else
    \PackageInfo{minitoc}%
       {PREPARING MINILOTS FROM #1}%
    \expandafter\MLT@read\fi}
```

\MLT@read \read \@inputcheck

Read the next entry in the `.lot` file:

```
\def\MLT@read{%
  \read\@inputcheck to\MLT@line
```

\MLT@line \MLT@test \MLT@

The `.....` make sure that `\MLT@test` has sufficient arguments:

```
  \expandafter\MLT@test\MLT@line.....\MLT@% %%HO: . added
  }%
```

\MLT@test

The `\MLT@test` macro finds the "interesting" commands in the LOT file, mainly to delimit chapters.

\contentsline \mtc@string \MLT@contentsline \@input \MLT@list \chapterend \closeout \tf@mtc \openout \chapterbegin \addtocounter \MLT@lot \MLT@read

Look at the first token of the line. If it is an interesting entry, process it. If it is `\@input`, add the file to the list. Otherwise ignore. Go around the loop if not at end of file. Finally process the next file in the list.

```
\long\def\MLT@test#1#2#3#4#5#6\MLT@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \MLT@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
    \let\mtc@string\relax
  \else\ifx#1\@input
     \edef\MLT@list{\MLT@list#2\relax}%
  \else\ifx#1\chapterend
     \immediate\closeout\tf@mtc
     \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\chapterbegin
    \addtocounter{mtc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\MLT@lot
  \else\expandafter\MLT@read\fi}%
```

Note that we terminate with a closing brace to end the chapter-level macros (end of the *else* branch of a `\@ifundefined{chapter}` alternative).

```
}%
```

5.43 Macro to write a contents line

\mtc@dot \MTC@WriteContentsline \mtc@param \write \tf@mtc \@resetfont \mtc@string \contentsline \mtc@toks

The \MTC@WriteContentsline macro makes the definition of \MTC@contentsline shorter. An extra \edef level is removed (Heiko OBERDIEK):

The arguments of \MTC@WriteContentsline are:

- #1: the #1 argument of \MTC@contentsline;
- #2: font shorthand $\Longrightarrow$ \csname #2font\endcsname;
- #3: the #3 argument of \MTC@contentsline;
- #4: the #4 argument of \MTC@contentsline (hyperlink).

```
\def\mtc@dot{.}
\def\MTC@WriteContentsline#1#2#3#4{%
  \def\mtc@param{#4}%
   \immediate\write\tf@mtc{%
    {\string\reset@font
     \expandafter\string\csname #2font\endcsname
     \string\mtc@string
     \string\contentsline{#1}%
     {\the\mtc@toks}%
     {\string\reset@font
      \expandafter\string\csname #2font\endcsname
      \space #3%
     }%
     \ifx\mtc@dot\mtc@param
     \else
       {#4}% %%HO/BJ: #4 is hyperlink
     \fi
    }%
  }%
}
```

\MTC@WriteCoffeeline \write \tf@mtc \@resetfont \coffeefont \mtc@string \mtc@toks

And the same for a “coffee” line. The arguments of \MTC@WriteCoffeeline are:

- #1: the #1 argument of \MTC@contentsline;
- #2: the #3 argument of \MTC@contentsline.

```
\def\MTC@WriteCoffeeline#1#2#3{%
  \immediate\write\tf@mtc{%
    {\string\reset@font
```

```
        \string\coffeefont
        \string\mtc@string
%% COFFEE \string\textbf{#1}%
        {\the\mtc@toks}%
        {\string\reset@font
         \string\coffeefont
         \space #3%
        }%
      }%
  }%
}
```

5.44 Depth counters for partlofs and partlots

\AtBeginDocument \newcounter \setcounter \c@lofdepth \c@lotdepth

If the counters `lofdepth` and `lotdepth` are defined, we create the corresponding new counters: `partlofdepth` and `partlotdepth`. These counters are initialised to 2. This is done after the loading of the packages, in an `\AtBeginDocument` block:

```
\AtBeginDocument{%
 \@ifundefined{c@lofdepth}{}%
   {\newcounter{partlofdepth}\setcounter{partlofdepth}{2}}%
 \@ifundefined{c@lotdepth}{}%
   {\newcounter{partlotdepth}\setcounter{partlotdepth}{2}}%
}
```

5.45 Part level commands

\xpart \theptc \Thepart \adjustptc \decrementptc \incrementptc \firstpartis \firstpartis \if@firstpartis@used@ \newcounter \setcounter

If `\part` is defined, we define some utility commands, a counter (`ptc`) for the parttocs and related commands (`\theptc`, `\Thepart`, `\adjustptc`, `\decrementptc`, `\incrementptc`), the obsolete command `\firstpartis`, and the depth counter `parttocdepth`.

```
\@ifundefined{part}{}%
{%
\def\xpart{xpart}
\def\Thepart{\arabic{ptc}}
\def\firstpartis#1%
  {\PackageWarning{minitoc}%
     {*** \string\firstpartis \space is an obsolete command ***}%
     \@firstpartis@used@true}
\newcounter{ptc}
\setcounter{ptc}{0}
\newcommand{\adjustptc}[1][1]{\addtocounter{ptc}{#1}}
\def\decrementptc{\addtocounter{ptc}{-1}}
\def\incrementptc{\addtocounter{ptc}{+1}}
```

```
\def\theptc{\arabic{ptc}}
\newcounter{parttocdepth}
\setcounter{parttocdepth}{2}
```

\ptc@rule

But, sometimes, we need to make a difference between book/report and article classes (is `\chapter` defined?), to have a different layout: the definition of `\ptc@rule` is empty except if `\chapter` is undefined. By default, there is no rule before/after parttocs, partlofs, and partlots for books. You should redeclare `\ptc@rule` if you want these rules.

```
\@ifundefined{chapter}%
   {\def\ptc@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}}%
   {\let\ptc@rule\relax}
```

\ptcindent

And we declare the default indentation (both sides) of the parttocs:

```
\newlength\ptcindent
\@ifundefined{chapter}{\ptcindent=24\p@}{\ptcindent=\z@}
```

5.46 Fonts for the parttocs

\ptcfont
\ptcSfont
\ptcSSfont
\ptcSSSfont
\ptcPfont
\ptcSPfont
\plffont
\pltfont
\ptifont

We define the fonts for the parttocs. Note that they are larger if `\chapter` is defined (`book/report`-like document classes) than when it is not (`article`-like document classes):

```
\@ifundefined{chapter}{%
   \def\ptcfont{\small\rmfamily\upshape\mdseries} % the parttoc
   \def\ptcSfont{\small\rmfamily\upshape\bfseries}% (sections)
   \let\ptcSSfont\ptcfont     % (subsections)
   \let\ptcSSSfont\ptcfont    % (subsubsections)
   \let\ptcPfont\ptcfont      % (paragraphs)
   \let\ptcSPfont\ptcfont     % (subparagraphs)
   \let\plffont\ptcfont       % (figures)
   \let\pltfont\ptcfont       % (tables)
   \def\ptifont{\Large\rmfamily\upshape\bfseries}% titles
}%
```

\ptcfont
\ptcCfont
\ptcSfont
\ptcSSfont
\ptcSSSfont
\ptcPfont
\ptcSPfont
\plffont
\pltfont
\ptifont

If `\chapter` is defined, the fonts are larger and `\ptcCfont` must be defined:

```
{%
   \def\ptcfont{\normalsize\rmfamily\upshape\mdseries} % the parttoc
   \def\ptcCfont{\normalsize\rmfamily\upshape\bfseries}% (chapters)
   \def\ptcSfont{\normalsize\rmfamily\upshape\mdseries}% (sections)
   \let\ptcSSfont\ptcfont     % (subsections)
   \let\ptcSSSfont\ptcfont    % (subsubsections)
   \let\ptcPfont\ptcfont      % (paragraphs)
```

```
  \let\ptcSPfont\ptcfont     % (subparagraphs)
  \let\plffont\ptcfont       % (figures)
  \let\pltfont\ptcfont       % (tables)
  \def\ptifont{\LARGE\rmfamily\upshape\bfseries}% titles
}
```

5.47 Default titles for part-level mini-tables

\parttoc \partlof \partlot

We define the default position, the fonts and the layout for titles of the part-level mini-tables (\parttoc, \partlof and \partlot). This formating is different if \chapter is undefined or defined.

\c@pti \l@pti \r@pti \e@pti \n@pti

If \chapter is undefined, the definitions are very simple, for centered, flushleft, flushright or empty titles. Empty titles need a vertical correction (Frank Mittelbach).

```
\@ifundefined{chapter}{%
\def\c@pti#1{\null\hfill #1\hfill\null}
\def\l@pti#1{\null #1\hfill\null}
\def\r@pti#1{\null\hfill #1\null}
\def\e@pti#1{\vspace{-\baselineskip}}
\def\n@pti#1{\vspace{-\baselineskip}}
}%
```

\e@pti \n@pti \if@twocolumn \@topnewpage \@makephead@l \@makephead@r \@makephead@c \@afterheading \ptifont

But, if \chapter is defined, we must simulate the formatting of a chapter head, which is more complex. Empty titles need a vertical correction (Frank Mittelbach).

```
{%
\def\e@pti#1{\vspace{-\baselineskip}}
\def\n@pti#1{\vspace{-\baselineskip}}
```

\if@twocolumn \@topnewpage \@afterheading \ptifont \l@pti \@makephead@l

For a title on the left, we must test if the main text is on two columns:

```
\def\l@pti#1{\if@twocolumn
    \@topnewpage[\@makephead@l{#1}]%
    \else
    \@makephead@l{#1}%
    \@afterheading
    \fi}
\def\@makephead@l#1{%
    \vspace*{50\p@}%
    {\parindent \z@ \raggedright
     \ptifont
     #1\par
     \nobreak
     \vskip 40\p@
    }}
```

\r@pti
\@makephead@l

For a title on the right, we must also test if the main text is on two columns:

```
\def\r@pti#1{\if@twocolumn
    \@topnewpage[\@makephead@r{#1}]%
    \else
    \@makephead@r{#1}%
    \@afterheading
    \fi}
\def\@makephead@r#1{%
    \vspace*{50\p@}%
    {\parindent \z@ \raggedleft
     \ptifont
     #1\par
     \nobreak
     \vskip 40\p@
    }}
```

\c@pti
\@makephead@c

For a centered title, we must also test if the main text is on two columns:

```
\def\c@pti#1{\if@twocolumn
    \@topnewpage[\@makephead@c{#1}]%
    \else
    \@makephead@c{#1}%
    \@afterheading
    \fi}
\def\@makephead@c#1{%
    \vspace*{50\p@}%
    {\parindent \z@ \centering
     \ptifont
     #1\par
     \nobreak
     \vskip 40\p@
    }}%
}
```

\l@pti
\do@ptitc
\df@ptitc
\do@ptilf
\df@ptilf
\do@ptilt
\df@ptilt

By default, titles are on left:

```
\let\do@ptitc\l@pti
\let\df@ptitc\l@pti
\let\do@ptilf\l@pti
\let\df@ptilf\l@pti
\let\do@ptilt\l@pti
\let\df@ptilt\l@pti
```

5.48 The ptc@verse environment

ptc@verse \ptc@verse \endptc@verse \\ \@centercr \list \itemsep \itemindent \listparindent \topsep \parsep \parskip \partopsep \ptcindent \iftightmtc \ifktightmtc

Each parttoc is placed inside a `ptc@verse` environment. This environment is analog to the standard `verse` environment and hence defined via two commands: `\ptc@verse` and `\endptc@verse`. As it is a list environment, we first define (in a local way) `\\`, then call `\list{}` and set some dimensions like `\itemsep`, `\itemindent`, `\listparindent`, `\itemindent`, `\partopsep`, `\topsep`. `\parsep` is set to zero if the `tight` option is active (this reduces the spacing between the lines). `\parskip` is set to zero if the `k-tight` option is active (this reduces the spacing between the lines). Both margins are set to `\ptcindent`. `\endptc@verse` terminates the list and discourages a page break.

```
\def\ptc@verse{\let\\=\@centercr
  \list{}{\itemsep\z@
          \itemindent \z@
          \listparindent \itemindent
          \partopsep\z@
          \iftightmtc \parsep\z@ \fi
          \ifktightmtc \parskip\z@ \fi
          \topsep=1ex
          \leftmargin\ptcindent
          \rightmargin\leftmargin}\item[]}
\def\endptc@verse{\nopagebreak[4]\endlist}
```

5.49 The part level mini-tables: \parttoc, \partlof, and \partlot

\parttoc \partlof \partlot

These commands are essentialy similar to the `\minitoc` command, except that they should be placed after a `\part` command to produce a parttoc, a partlof or a partlot, and the formatting is different and depends of the availability of the `\chapter` command (for the fonts and the horizontal rules). The code is very similar. The `\partlof` and `\partlot` commands are siblings of the `\parttoc` command. Note that `\parttoc`, `\partlof` and `\partlot` use page styles, because `\beforepart...` and `\afterpart...` commands imply usually a `\clear[double]page` command, and hence `\markboth{...}{...}` must be called.

5.49.1 The \parttoc command

\parttoc \parttoc@

This command must be used after `\part` if you need a parttoc (no automatic parttoc). First, `\parttoc` detects the presence of its optional argument, and uses its default value, `d`, if it is missing. Then, `\parttoc@` is called with the effective position as argument:

```
\def\parttoc{\@ifnextchar[{\parttoc@}{\parttoc@[d]}}
```

\parttoc@
\if@parttoc@used@
\if@longextensions@
\@tocfile
\Thepart

The `\parttoc@` macro does the real work. It first sets the flag `\if@parttoc@used@` (for a coherence hint) and checks if long extensions are used or not (to create the name of the parttoc file):

```
\def\parttoc@[#1]{%
\global\@parttoc@used@true
\if@longextensions@%
\def\@tocfile{ptc\Thepart}%
\else
\def\@tocfile{P\Thepart}%
\fi
```

\mtc@CkFile
\if@mtc@FE
\@tocfile

Then, we check the presence of the parttoc file and give a warning if it is not here:

```
        \mtc@CkFile{\jobname.\@tocfile}
        \if@mtc@FE
        \PackageInfo{minitoc}%
           {\jobname.\@tocfile\space is empty}
        \@mtc@empty@parttoc@true
        \else
```

\beforeparttoc

If the parttoc file is present, we can insert it, but we must add some presentation code: first, `\beforeparttoc`, of course:

```
        \beforeparttoc
```

\mtc@markboth
\@mkboth
\thispageparttocstyle
\ptctitle

If `\chapter` is defined, we just set the page marks with the parttoc title and set the page style:

```
        \@ifundefined{chapter}{}{%
        \global\let\mtc@markboth\markboth
        \global\let\@mkboth\markboth
        \thispageparttocstyle
        \mtc@markboth{\uppercase{\ptctitle}}{\uppercase{\ptctitle}}%
        }%
```

\do@ptitc
\e@pti
\n@pti
\c@pti
\l@pti
\r@pti
\df@pti
\mtc@CkStr
\ptctitle
\if@mtc@FE
samepage

A `samepage` environment is begun, then the argument is treated to set the position of the parttoc title. If the title string is empty, this forces the positionning.

```
        \relax\begin{samepage}%
        \if #1e\let\do@ptitc\e@pti
        \else\if #1n\let\do@ptitc\n@pti
        \else\if #1c\let\do@ptitc\c@pti
        \else\if #1l\let\do@ptitc\l@pti
        \else\if #1r\let\do@ptitc\r@pti
        \else\if #1d\let\do@ptitc\df@ptitc
        \fi\fi\fi\fi\fi\fi
        \mtc@CkStr{\ptctitle}\if@mtc@FE \let\do@ptitc\e@pti\relax\fi
```

\raggedright \parskip \ptcfont

We adjust some formatting parameters and avoid a page break between the title and the parttoc, then we set the font:

```
        \raggedright
        \parskip=\z@%
        \reset@font\ptcfont%
        \parindent=\z@%
        \nopagebreak[4]%
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent
        \nopagebreak[4]%
```

\ptc@rule tabular \ptifont \do@ptitc \mtc@v \ptctitle \hline

The parttoc title is set in a `tabular` environment (to inhibit a page break between the title and the top rule), with a rule at its bottom if necessary. This rule is an `\hline`. It is the top rule of the parttoc.

```
        \ifx\ptc@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\ptifont\do@ptitc{\mtc@v\ptctitle}\\
        \end{tabular}%
        \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\ptifont\do@ptitc{\mtc@v\ptctitle}\\\hline
        \end{tabular}%
        \fi
```

\mtc@zrule \mtc@BBR \ptcindent

Then, we adjust the position under the top rule and set the indentation and some formatting parameters:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\*[-\baselineskip]\mtc@BBR
        \leftmargin\ptcindent
        \rightmargin\ptcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
```

ptc@verse \c@tocdepth \c@parttocdepth \mtc@BBR

We enter in a `ptc@verse` environment to format the parttoc. The toc depth is forced (locally) to `parttocdepth`. A little trick is necessary to adjust the position. A blank line is necessary to avoid a negative indentation.

```
        \begin{ptc@verse}\c@tocdepth=\c@parttocdepth%
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\ptc@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \ptc@setform \@tocfile \mtc@strut ptc@verse

If the contents lines must have no numbers, we replace the macro \@dottedtocline with its undotted version. For chapter-level entries, we must invoke \l@chapter ignoring the page number argument. A hook (redefinissable coommand) is added, and the formatting settings coming from \mtcsetformat are activated via \ptc@setform. Then the parttoc file is inserted, followed by a strut, and the ptc@verse environment is terminated.

```
\begingroup
  \makeatletter
  \@ifundefined{ptc@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@ifundefined{ptc@pgno}%
  {\let\l@chapter@SVPN\l@chapter%
    \def\l@chapter##1##2{\l@chapter@SVPN{##1}{\hbox{}}}}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \ptc@setform
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-1\baselineskip}
  \leavevmode\mtc@strut
  \global\@nobreakfalse\endgroup
        \end{ptc@verse}%
```

\ptc@rule \mtc@zrule samepage \afterparttoc

The final part is just to add the bottom rule, if necessary, a possible page break (if \chapter is not defined), and \afterparttoc.

```
        \kern-1.\baselineskip%
        \nopagebreak[4]\ptc@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
        \par\@ifundefined{chapter}{\pagebreak[1]\vspace*{-1ex}}%
        \afterparttoc\fi}%
```

5.49.2 The \partlof command

\partlof \partlof@

This command must be used after \part if you need a partlof (no automatic partlof). First, \partlof detects the presence of its optional argument, and uses its default value, d, if it is missing. Then, \partlof@ is called with the effective position as argument:

```
\def\partlof{\@ifnextchar[{\partlof@}{\partlof@[d]}}
```

\partlof@ \if@partlof@used@ \if@longextensions@ \@tocfile \Thepart

The \partlof@ macro does the real work. It first sets the flag \if@partlof@used@ (for a coherence hint) and checks if long extensions are used or not (to create the name of the partlof file):

```
\def\partlof@[#1]{%
\global\@partlof@used@true
```

```
\if@longextensions@%
\def\@tocfile{plf\Thepart}%
\else
\def\@tocfile{G\Thepart}%
\fi
```

\mtc@CkFile \if@mtc@FE \@tocfile

Then, we check the presence of the partlof file and give a warning if it is not here:

```
        \mtc@CkFile{\jobname.\@tocfile}
        \if@mtc@FE
        \PackageInfo{minitoc}%
           {\jobname.\@tocfile\space is empty}
        \@mtc@empty@partlof@true
        \else
```

\beforepartlof

If the partlof file is present, we can insert it, but we must add some presentation code: first, \beforepartlof, of course:

```
        \beforepartlof
```

\mtc@markboth \@mkboth \thispagepartlofstyle \plftitle

If \chapter is defined, we just set the page marks with the partlof title and set the page style:

```
        \@ifundefined{chapter}{}{%
        \global\let\mtc@markboth\markboth
        \global\let\@mkboth\markboth
        \thispagepartlofstyle
        \mtc@markboth{\uppercase{\plftitle}}{\uppercase{\plftitle}}%
        }%
```

\do@ptilf \e@pti \n@pti \c@pti \l@pti \r@pti \df@pti \mtc@CkStr \plftitle \if@mtc@FE samepage

A samepage environment is begun, then the argument is treated to set the position of the partlof title. If the title string is empty, this forces the positionning.

```
        \relax\begin{samepage}%
        \if #1e\let\do@ptilf\e@pti
        \else\if #1n\let\do@ptilf\n@pti
        \else\if #1c\let\do@ptilf\c@pti
        \else\if #1l\let\do@ptilf\l@pti
        \else\if #1r\let\do@ptilf\r@pti
        \else\if #1d\let\do@ptilf\df@ptilf
        \fi\fi\fi\fi\fi\fi
        \mtc@CkStr{\plftitle}\if@mtc@FE \let\do@ptilf\e@pti\relax\fi
```

\raggedright \parskip \plffont

We adjust some formatting parameters and avoid a page break between the title and the parttoc, then we set the font:

```
        \raggedright
        \parskip=\z@%
        \reset@font\plffont%
```

\plf@rule tabular \ptifont \do@ptilf \mtc@v \plftitle \hline

The parttoc title is set in a `tabular` environment (to inhibit a page break between the title and the top rule), with a rule at its bottom if necessary. This rule is an `\hline`. It is the top rule of the partlof.

```
        \parindent=\z@%
        \nopagebreak[4]%
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent
        \ifx\plf@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\ptifont\do@ptilf{\mtc@v\plftitle}\\
        \end{tabular}%
        \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\ptifont\do@ptilf{\mtc@v\plftitle}\\\hline
        \mtc@hstrut\\
        \end{tabular}%
        \fi
```

\mtc@zrule \mtc@BBR \ptcindent

Then, we adjust the position under the top rule and set the indentation and some formatting parameters:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\*[-\baselineskip]\mtc@BBR
        \leftmargin\ptcindent
        \rightmargin\ptcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
```

ptc@verse \mtc@BBR

We enter in a `ptc@verse` environment to format the partlof. If necessary, the toc depth is forced (locally) to `partlofdepth`. A little trick is necessary to adjust the position. A blank line is necessary to avoid a negative indentation.

```
        \begin{ptc@verse}%
        \@ifundefined{c@lofdepth}{}{\c@tocdepth=\c@partlofdepth}
        \ifnum\c@tocdepth<1\relax\c@tocdepth=1\fi
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\plf@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \plf@setform \@tocfile \mtc@strut ptc@verse

If the contents lines must have no numbers, we replace the macro `\@dottedtocline` with its undotted version. A hook is added, and the formatting settings coming from `\mtcsetformat` are activated via `\plf@setform`. Then the partlof file is inserted, followed by a strut, and the `ptc@verse` environment is terminated.

```
\begingroup
  \makeatletter
  \@ifundefined{plf@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \plf@setform
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-0\baselineskip}
  \leavevmode\mtc@strut
  \global\@nobreakfalse\endgroup
        \end{ptc@verse}%
```

\plf@rule \mtc@zrule samepage \afterpartlof

The final part is just to add the bottom rule, if necessary, a possible page break (if `\chapter` is not defined), and `\afterpartlof`. The blank line (\\) is essential.

```
        \kern-1.\baselineskip%
        \nopagebreak[4]\plf@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
        \par\@ifundefined{chapter}{\pagebreak[1]\vspace*{-1ex}}%
        \afterpartlof\fi}
```

5.49.3 The `\partlot` command

\partlot \partlot@

This command must be used after `\part` if you need a partlot (no automatic partlot). First, `\partlot` detects the presence of its optional argument, and uses its default value, `d`, if it is missing. Then, `\partlot@` is called with the effective position as argument:

```
\def\partlot{\@ifnextchar[{\partlot@}{\partlot@[d]}}
```

\partlot@ \if@partlot@used@ \if@longextensions@ \@tocfile \Thepart

The `\partlot@` macro does the real work. It first sets the flag `\if@partlot@used@` (for a coherence hint) and checks if long extensions are used or not (to create the name of the partlot file):

```
%
\def\partlot@[#1]{%
\global\@partlot@used@true
\if@longextensions@%
\def\@tocfile{plt\Thepart}%
\else
\def\@tocfile{U\Thepart}%
\fi
```

\mtc@CkFile \if@mtc@FE \@tocfile

Then, we check the presence of the partlot file and give a warning if it is not here:

```
       \mtc@CkFile{\jobname.\@tocfile}
       \if@mtc@FE
       \PackageInfo{minitoc}%
          {\jobname.\@tocfile\space is empty}
       \@mtc@empty@partlof@true
       \else
```

\beforepartlot

If the partlot file is present, we can insert it, but we must add some presentation code: first, `\beforepartlot`, of course:

```
       \beforepartlot
```

\mtc@markboth \@mkboth \thispagepartlotstyle \plttitle

If `\chapter` is defined, we just set the page marks with the partlot title and set the page style:

```
       \@ifundefined{chapter}{}{%
       \global\let\mtc@markboth\markboth
       \global\let\@mkboth\markboth
       \thispagepartlotstyle
       \mtc@markboth{\uppercase{\plttitle}}{\uppercase{\plttitle}}%
       }%
```

\do@ptilt \e@pti \n@pti \c@pti \l@pti \r@pti \df@pti \mtc@CkStr \plttitle \if@mtc@FE samepage

A `samepage` environment is begun, then the argument is treated to set the position of the partlof title. If the title string is empty, this forces the positionning.

```
       \relax\begin{samepage}%
       \if #1e\let\do@ptilt\e@pti
       \else\if #1n\let\do@ptilt\n@pti
       \else\if #1c\let\do@ptilt\c@pti
       \else\if #1l\let\do@ptilt\l@pti
       \else\if #1r\let\do@ptilt\r@pti
       \else\if #1d\let\do@ptilt\df@ptilt
       \fi\fi\fi\fi\fi\fi
       \mtc@CkStr{\plttitle}\if@mtc@FE \let\do@ptilt\e@pti\relax\fi
```

\raggedright \parskip \pltfont

We adjust some formatting parameters and avoid a page break between the title and the partlot, then we set the font:

```
       \raggedright
       \parskip=\z@%
       \reset@font\pltfont%
       \parindent=\z@%
       \nopagebreak[4]%
       \kern-0.8\baselineskip\nopagebreak[4]%
       \par\noindent
```

\ptc@rule tabular \ptifont \do@ptilt \mtc@v \plttitle \hline

The partlot title is set in a `tabular` environment (to inhibit a page break between the title and the top rule), with a rule at its bottom if necessary. This rule is an `\hline`. It is the top rule of the partlot.

```
        \ifx\plt@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\ptifont\do@ptilt{\mtc@v\plttitle}\\
        \end{tabular}%
        \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\ptifont\do@ptilt{\mtc@v\plttitle}\\\hline
        \mtc@hstrut\\
        \end{tabular}%
        \fi
```

\mtc@zrule \mtc@BBR \ptcindent

Then, we adjust the position under the top rule and set the indentation and some formatting parameters:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\*[-\baselineskip]\mtc@BBR
        \leftmargin\ptcindent
        \rightmargin\ptcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
```

ptc@verse \mtc@BBR

We enter in a `ptc@verse` environment to format the parttoc. If necessary, the toc depth is forced (locally) to `partlotdepth`. A little trick is necessary to adjust the position.

```
        \begin{ptc@verse}%
        \@ifundefined{c@lotdepth}{}{\c@tocdepth=\c@partlotdepth}
        \ifnum\c@tocdepth<1\relax\c@tocdepth=1\fi
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\plt@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \plt@setform \@tocfile \mtc@strut ptc@verse

If the contents lines must have no numbers, we replace the macro `\@dottedtocline` with its undotted version. A hook is added, and the formatting settings coming from `\mtcsetformat` are activated via `\plt@setform`. Then the partlot file is inserted, followed by a strut, and the `ptc@verse` environment is terminated.

```
\begingroup
  \makeatletter
  \@ifundefined{plt@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \plt@setform
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-0\baselineskip}
  \leavevmode\mtc@strut
```

```
\global\@nobreakfalse\endgroup
      \end{ptc@verse}%
```

\ptc@rule \mtc@zrule samepage \afterpartlot

The final part is just to add the bottom rule, if necessary, a possible page break (if `\chapter` is not defined), and `\afterpartlot`. The blank line (`\\`) is essential.

```
      \kern-1.\baselineskip%
      \nopagebreak[4]\plt@rule\null\leavevmode\\%
      \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
      \par\@ifundefined{chapter}{\pagebreak[1]\vspace*{-1ex}}%
      \afterpartlot\fi}
```

5.50 Auxiliary commands for printing parttocs

\@dottedtocline \l@xpart \l@pchapter \l@psection \pchapter \psection

The following auxiliary commands are used in the printing of parttocs. Note that `\l@xpart` uses a huge depth to inhibit the printing of its contents line (except if you cheat). These command are similar to `\l@subsection`, only the arguments have been altered:

```
\def\l@xpart{\@dottedtocline{\@M}{1.0em}{2.3em}}
\def\l@pchapter{\@dottedtocline{1}{1.0em}{2.3em}}
\def\l@psection{\@dottedtocline{2}{1.0em}{2.3em}}
\def\pchapter{pchapter}
\def\psection{psection}
```

5.51 Patching the `\part` command, continued

\sv@part \mtc@svpart \@part \addtocontents \sv@spart \ptc@spart \@spart \stepcounter \partbegin \partend

We patch both branches of the `\part` command: `\@part` (unstarred `\part`) and `\@spart` (`\part*`). We add the incrementation of the `ptc` counter to both branches. In the unstarred branch, we add `xpart` entries in the TOC, the LOF and the LOT. In the starred branch, we add a `\partbegin` line in the TOC. This command is just a marker and does nothing real (`\relax`).

```
\let\sv@part\mtc@svpart
\def\@part[#1]#2{\sv@part[{#1}]{#2}\relax
\addcontentsline{lof}{xpart}{#1}%
\addcontentsline{lot}{xpart}{#1}%
\addcontentsline{toc}{xpart}{#1}%
\stepcounter{ptc}}
\let\sv@spart\@spart
\def\@spart{\stepcounter{ptc}\sv@spart}
\let\ptc@spart\@spart
\def\@spart{\addtocontents{toc}{\protect\partend}\ptc@spart}
```

```
\def\@spart{\addtocontents{toc}{\protect\partbegin}\ptc@spart}
\let\partend\relax
\let\partbegin\relax
```

5.52 The \doparttoc command and its siblings

\doparttoc \dopartlof \dopartlot

The \doparttoc command works like the \dominitoc command, \dopartlof like \dominilof and \dopartlot like \dominilot.

\@doparttoc \PTC@next \setcounter

The \doparttoc command extracts information from the .toc file and creates the .ptc⟨N⟩ files (.ptc becomes .P on MS-DOS).

```
\def\@doparttoc#1{{%
  \makeatletter
  \setcounter{ptc}{0}%
  \PTC@next#1.toc\relax\\}\setcounter{ptc}{0}}%
```

\@dopartlof \PLF@next \setcounter

The \dopartlof command extracts information from the .lof file and creates the .plf⟨N⟩ files (.plf becomes .G on MS-DOS).

```
\def\@dopartlof#1{{%
  \makeatletter
  \setcounter{ptc}{0}%
  \PLF@next#1.lof\relax\\}\setcounter{ptc}{0}}%
```

\@dopartlot \PLT@next \setcounter

The \dopartlot command extracts information from the .lot file and creates the .plt⟨N⟩ files (.plt becomes .U on MS-DOS).

```
\def\@dopartlot#1{{%
  \makeatletter
  \setcounter{ptc}{0}%
  \PLT@next#1.lot\relax\\}\setcounter{ptc}{0}}%
```

\doparttoc \dopartlof \dopartlot

We define the user macros, who detect the optional argument:

```
\def\doparttoc{\@ifnextchar[{\doparttoc@}{\doparttoc@[l]}}
\def\dopartlof{\@ifnextchar[{\dopartlof@}{\dopartlof@[l]}}
\def\dopartlot{\@ifnextchar[{\dopartlot@}{\dopartlot@[l]}}
```

\doparttoc@ \if@doparttoc@used@ \df@ptitc \e@pti \n@pti \c@pti \l@pti \r@pti

We treat the optional argument of \doparttoc (it becomes the default position for titles of parttocs) and flag this macro as used:

```
\def\doparttoc@[#1]{%
```

```
\global\@doparttoc@used@true
\if #1e\let\df@ptitc\e@pti%
\else\if #1n\let\df@ptitc\n@pti%
\else\if #1c\let\df@ptitc\c@pti%
\else\if #1l\let\df@ptitc\l@pti%
\else\if #1r\let\df@ptitc\r@pti%
\fi\fi\fi\fi\fi%
\@@doparttoc}
```

\dopartlof@ \if@dopartlof@used@ \df@ptilf \e@pti \n@pti \c@pti \l@pti \r@pti

We treat the optional argument of \dopartlof (it becomes the default position for titles of partlofs) and flag this macro as used:

```
\def\dopartlof@[#1]{%
\global\@dopartlof@used@true
\if #1e\let\df@ptilf\e@pti%
\else\if #1n\let\df@ptilf\n@pti%
\else\if #1c\let\df@ptilf\c@pti%
\else\if #1l\let\df@ptilf\l@pti%
\else\if #1r\let\df@ptilf\r@pti%
\fi\fi\fi\fi\fi%
\@@dopartlof}
```

\dopartlot@ \if@dopartlot@used@ \df@ptilt \e@pti \n@pti \c@pti \l@pti \r@pti

We treat the optional argument of \dopartlot (it becomes the default position for titles of partlofs) and flag this macro as used:

```
\def\dopartlot@[#1]{%
\global\@dopartlot@used@true
\if #1e\let\df@ptilt\e@pti%
\else\if #1n\let\df@ptilt\n@pti%
\else\if #1c\let\df@ptilt\c@pti%
\else\if #1l\let\df@ptilt\l@pti%
\else\if #1r\let\df@ptilt\r@pti%
\fi\fi\fi\fi\fi%
\@@dopartlot}
```

\@@doparttoc \@@dopartlof \@@dopartlot \tf@mtc

These macros invoke the \@dopart... commands to create the mini-table file, then close the file descriptor.

```
\def\@@doparttoc{\@doparttoc{\jobname}\immediate\closeout\tf@mtc}
\def\@@dopartlof{\@dopartlof{\jobname}\immediate\closeout\tf@mtc}
\def\@@dopartlot{\@dopartlot{\jobname}\immediate\closeout\tf@mtc}
```

5.52.1 Processing macros for the parttocs

\PTC@next \PTC@list \PTC@loop

Processing the next entry in the list and remove it from the head of the list:

```
\def\PTC@next#1\relax#2\\{%
  \edef\PTC@list{#2}%
  \PTC@loop{#1}}
```

\PTC@toc
\PTC@list
\PTC@explist

Check if the list is empty:

```
\def\PTC@toc{%
  \ifx\PTC@list\@empty\else\expandafter\PTC@explist\fi}
```

\PTC@contentsline
\part
\theptc
\tf@mtc
\ptcname
\MTC@WriteContentsLine

The macro \PTC@contentsline analyses the lines read from the TOC file and detects interesting keywords. If \part is found, ptc is incremented and a new partlof file is created.

```
\def\PTC@contentsline#1#2#3#4{% %%HO/DV: 4 instead of 3 parameters
  \expandafter\ifx\csname #1\endcsname\part
    \stepcounter{ptc}%
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.ptc\theptc\@gobble}%
      \def\ptcname{\jobname.ptc\theptc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.P\theptc\@gobble}%
      \def\ptcname{\jobname.P\theptc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\ptcname
  \fi
  \expandafter\ifx\csname #1\endcsname\starpart\relax
    \stepcounter{ptc}%
```

\if@longextensions@
\ptcname

We test if long or short extensions are used, to build the name of the mini-table file, then open it:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.ptc\theptc}%
      \def\ptcname{\jobname.ptc\theptc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.P\theptc}%
      \def\ptcname{\jobname.P\theptc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\ptcname
  \fi
```

\mtc@toks

The token register `\mtc@toks` is used to pass the entry to `\MTC@WriteContentsline`:

```
 \mtc@toks{\noexpand\leavevmode #2}%
```

\MTC@WriteContentsline \chapter \pchapter \section \coffee \subsection \subsubsection \paragraph \subparagraph

Now, we filter the relevant contents lines; this code extracts and writes info for chapters, sections, etc.:

```
 \expandafter\ifx\csname #1\endcsname\chapter
   \MTC@WriteContentsline{#1}{ptcC}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\pchapter
   \MTC@WriteContentsline{#1}{ptcC}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\section
   \MTC@WriteContentsline{#1}{ptcS}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\coffee
   \MTC@WriteCoffeeline{#1}{#3}%
 \fi
 \expandafter\ifx\csname #1\endcsname\subsection
   \MTC@WriteContentsline{#1}{ptcSS}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\subsubsection
   \MTC@WriteContentsline{#1}{ptcSSS}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\paragraph
   \MTC@WriteContentsline{#1}{ptcP}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\subparagraph
   \MTC@WriteContentsline{#1}{ptcSP}{#3}{#4}%
 \fi
```

\MTC@WriteContentsline \starchapter \starsection \starsubsection \starsubsubsection \starparagraph \starsubparagraph

And for the starred sectionning commands:

```
 \expandafter\ifx\csname #1\endcsname\starchapter
%%HO: the following line should be disabled: \stepcounter{ptc}%
   \MTC@WriteContentsline{#1}{ptcC}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\starsection
   \MTC@WriteContentsline{#1}{ptcS}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\starsubsection
   \MTC@WriteContentsline{#1}{ptcSS}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\starsubsubsection
   \MTC@WriteContentsline{#1}{ptcSSS}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\starparagraph
   \MTC@WriteContentsline{#1}{ptcP}{#3}{#4}%
 \fi
 \expandafter\ifx\csname #1\endcsname\starsubparagraph
```

```
    \MTC@WriteContentsline{#1}{ptcSP}{#3}{#4}%
  \fi
}
```

\PTC@explist \PTC@next \PTC@list

The loop to read the lines of the TOC file; expands the list of entries and call \PTC@next to process the first one:

```
\def\PTC@explist{\expandafter\PTC@next\PTC@list\\}
```

\PTC@loop \PTC@toc \PTC@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\PTC@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1\MessageBreak PARTTOCS NOT PREPARED}%
    \expandafter\PTC@toc
  \else
    \PackageInfo{minitoc}%
       {PREPARING PARTTOCS FROM #1}%
    \expandafter\PTC@read\fi}
```

\PTC@read \PTC@line

Read the next entry of the `.toc` file.

```
\def\PTC@read{%
  \read\@inputcheck to\PTC@line
```

\PTC@test \PTC@line

The `.....` make sure that \PTC@test has sufficient arguments:

```
  \expandafter\PTC@test\PTC@line.....\PTC@% %%HO: . added
  }%
```

\PTC@test

The \PTC@test macro finds the "interesting" commands in the TOC file, mainly to delimit parts:

```
%%HO/BJ: now patch \PTC@test,
%%HO/BJ: call \PTC@contentsline with 4 instead of 3 parameters
```

\PTC@contentsline \mtc@string \PTC@list \PTC@toc \PTC@read \partend

Look at the first token of the line. If it is an interesting entry, process it. If it is \@input, add the file to the list. Otherwise ignore. Go around the loop if not at end of file. Finally process the next file in the list.

```
\long\def\PTC@test#1#2#3#4#5#6\PTC@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \PTC@contentsline{#2}{#3}{#4}{#5}%
```

```
  %%HO/DV: 4 instead of 3 parameters
  \let\mtc@string\relax
\else\ifx#1\@input
   \edef\PTC@list{\PTC@list#2\relax}%
\else\ifx#1\partend
   \immediate\closeout\tf@mtc
   \immediate\openout\tf@mtc=\jobname.mtc
\else\ifx#1\partbegin
   \addtocounter{ptc}{-1}%
\fi\fi\fi\fi
\ifeof\@inputcheck\expandafter\PTC@toc
\else\expandafter\PTC@read\fi}%
```

5.52.2 Processing macros for the partlofs

\PLF@next \PLF@list \PLF@loop

Processing the next entry in the list and remove it from the head of the list:

```
\def\PLF@next#1\relax#2\\{%
  \edef\PLF@list{#2}%
  \PLF@loop{#1}}
```

\PLF@lof \PLF@list \PLF@explist

Check if the list is empty:

```
\def\PLF@lof{%
  \ifx\PLF@list\@empty\else\expandafter\PLF@explist\fi}
```

\PLF@contentsline \part \theptc \tf@mtc \plfname \PLF@WriteContentsLine

The macro `\PLF@contentsline` analyses the lines read from the LOF file and detects interesting keywords. If `\part` is found, `ptc` is incremented and a new partlof file is created.

```
\def\PLF@contentsline#1#2#3#4{% %%HO: #4 added
  \expandafter\ifx\csname #1\endcsname\xpart
    \stepcounter{ptc}%
```

\if@longextensions@ \plfname

We test if long or short extensions are used, to build the name of the mini-table file, then open it:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.plf\theptc}%
      \def\plfname{\jobname.plf\theptc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.G\theptc}%
      \def\plfname{\jobname.G\theptc}%
    \fi
```

```
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\plfname
  \fi
```

\mtc@toks

The token register `\mtc@toks` is used to pass the entry to `\MTC@WriteContentsline`:

\figure
\mtc@toks
\MTC@WriteContentsline

Now, we filter the relevant contents lines:

```
  \expandafter\ifx\csname #1\endcsname\figure
    \mtc@toks{\noexpand\leavevmode#2}%
    \MTC@WriteContentsline{#1}{plf}{#3}{#4}%
  \fi
}
```

\PLF@explist
\PLF@next
\PLF@list

The loop to read the lines of the LOF file; expands the list of entries and call `\PLF@next` to process the first one:

```
\def\PLF@explist{\expandafter\PLF@next\PLF@list\\}
```

\PLF@loop
\PLF@lof
\PLF@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\PLF@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1\MessageBreak PARTLOFS NOT PREPARED}%
    \expandafter\PLF@lof
  \else
    \PackageInfo{minitoc}%
       {PREPARING PARTLOFS FROM #1}%
    \expandafter\PLF@read\fi}
```

\PLF@read
\PLF@line

Read the next entry of the `.lof` file.

```
\def\PLF@read{%
  \read\@inputcheck to\PLF@line
```

\PLF@test
\PLF@line

The `.....` make sure that `\PLF@test` has sufficient arguments:

```
  \expandafter\PLF@test\PLF@line.....\PLF@% %%HO: . added
  }%
```

\PLF@test

The `\PLF@test` macro finds the “interesting” commands in the LOF file, mainly to delimit parts:

```
%%HO/BJ: now patch \PLF@test,
%%HO/BJ: call \PLF@contentsline with 4 instead of 3 parameters
```

\PLF@contentsline
\mtc@string
\PLF@list
\PLF@lof
\PLF@read
\partend

Look at the first token of the line. If it is an interesting entry, process it. If it is `\@input`, add the file to the list. Otherwise ignore. Go around the loop if not at end of file. Finally process the next file in the list.

```
\long\def\PLF@test#1#2#3#4#5#6\PLF@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \PLF@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
    \let\mtc@string\relax
  \else\ifx#1\@input
     \edef\PLF@list{\PLF@list#2\relax}%
  \else\ifx#1\partend
     \immediate\closeout\tf@mtc
     \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\partbegin
     \addtocounter{ptc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\PLF@lof
  \else\expandafter\PLF@read\fi}%
```

5.52.3 Processing macros for the partlots

\PLT@next
\PLT@list
\PLT@loop

Processing the next entry in the list and remove it from the head of the list:

```
\def\PLT@next#1\relax#2\\{%
  \edef\PLT@list{#2}%
  \PLT@loop{#1}}
```

\PLT@lot
\PLT@list
\PLT@explist

Check if the list is empty:

```
\def\PLT@lot{%
  \ifx\PLT@list\@empty\else\expandafter\PLT@explist\fi}
```

\PLT@contentsline
\part
\theptc
\tf@mtc
\pltname
\PLT@WriteContentsLine

The macro `\PLT@contentsline` analyses the lines read from the LOT file and detects interesting keywords. If `\part` is found, `ptc` is incremented and a new partlot file is created.

```
\def\PLT@contentsline#1#2#3#4{% %%HO: #4 added
  \expandafter\ifx\csname #1\endcsname\xpart
    \stepcounter{ptc}%
```

\if@longextensions@ \pltname

We test if long or short extensions are used, to build the name of the mini-table file, then open it:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.plt\theptc}%
      \def\pltname{\jobname.plt\theptc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.U\theptc}%
      \def\pltname{\jobname.U\theptc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\pltname
  \fi
```

\mtc@toks

The token register `\mtc@toks` is used to pass the entry to `\MTC@WriteContentsline`:

\table \mtc@toks \MTC@WriteContentsline

Now, we filter the relevant contents lines:

```
  \expandafter\ifx\csname #1\endcsname\table
    \mtc@toks{\noexpand\leavevmode#2}%
    \MTC@WriteContentsline{#1}{plt}{#3}{#4}%
  \fi
}
```

\PLT@explist \PLT@next \PLT@list

The loop to read the lines of the LOT file; expands the list of entries and call `\PLT@next` to process the first one:

```
\def\PLT@explist{\expandafter\PLT@next\PLT@list\\}
```

\PLT@loop \PLT@lot \PLT@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\PLT@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1\MessageBreak PARTLOTS NOT PREPARED}%
    \expandafter\PLT@lot
  \else
    \PackageInfo{minitoc}%
       {PREPARING PARTLOTS FROM #1}%
    \expandafter\PLT@read\fi}
```

\PLT@read \PLT@line

Read the next entry of the `.lot` file.

```
\def\PLT@read{%
  \read\@inputcheck to\PLT@line
```

\PLT@test \PLT@line

The ..... make sure that \PLT@test has sufficient arguments:

```
  \expandafter\PLT@test\PLT@line.....\PLT@% %%HO: . added
  }%
```

\PLT@test

The \PLT@test macro finds the “interesting” commands in the LOT file, mainly to delimit parts:

```
%%HO/BJ: now patch \PLT@test,
%%HO/BJ: call \PLT@contentsline with 4 instead of 3 parameters
```

\PLT@contentsline \mtc@string \PLT@list \PLT@lot \PLT@read \partend

Look at the first token of the line. If it is an interesting entry, process it. If it is \@input, add the file to the list. Otherwise ignore. Go around the loop if not at end of file. Finally process the next file in the list.

```
\long\def\PLT@test#1#2#3#4#5#6\PLT@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \PLT@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
    \let\mtc@string\relax
  \else\ifx#1\@input
     \edef\PLT@list{\PLT@list#2\relax}%
  \else\ifx#1\partend
     \immediate\closeout\tf@mtc
     \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\partbegin
     \addtocounter{ptc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\PLT@lot
  \else\expandafter\PLT@read\fi}%
```

End of the part level stuff (begun in section 5.45 on page 118):

```
}%
```

5.53 Depth counters for sectlofs and sectlots

\AtBeginDocument \c@lofdepth \c@lotdepth \newcounter \setcounter

If the counters lofdepth and lotdepth are defined, we create new counters for the depths of the corresponding mini-tables: sectlofdepth and sectlotdepth. These counters are initialised to 2. This is done after the loading of the packages, in an \AtBeginDocument block:

```
\AtBeginDocument{%
 \@ifundefined{c@lofdepth}{}%
   {\newcounter{sectlofdepth}\setcounter{sectlofdepth}{2}}%
```

```
\@ifundefined{c@lotdepth}{}%
  {\newcounter{sectlotdepth}\setcounter{sectlotdepth}{2}}%
}
```

5.54 Section-level commands

\if@mtc@chapter@undef@ \if@mtc@section@def@

The section-level commands are defined only if `\chapter` is *not* defined, hence in article-like document classes, and only if `\section` is defined:

```
\if@mtc@chapter@undef@
\if@mtc@section@def@
```

\firstsectionis \adjuststc \decrementstc \incrementstc \stc@rule \stcindent

We define the obsolete command `\firstsectionis` (with its harmless warning), the counter `stc` of secttocs, the `\adjuststc`, `\decrementstc` and `\incrementstc` commands, the depth counter `sectocdepth` and its default value 2 (to include at least the subsections), the horizontal rule `\stc@rule` (rule before/after secttoc/sectlof/sectlot), the indentation (both sides) `\stcindent` for the secttocs (with its default values).

```
\def\firstsectionis#1%
  {\PackageWarning{minitoc}%
     {*** \string\firstsectionis \space is an obsolete command ***}%
     \@firstsectionis@used@true}
\newcounter{stc}\setcounter{stc}{0}%
\newcommand{\adjuststc}[1][1]{\addtocounter{stc}{#1}}%
\def\decrementstc{\addtocounter{stc}{-1}}%
\def\incrementstc{\addtocounter{stc}{+1}}%
\newcounter{secttocdepth}\setcounter{secttocdepth}{2}%
\def\stc@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}%
\newlength\stcindent \stcindent=24\p@
```

5.55 Fonts commands for secttocs and co.

\stcfont \stcSSfont \stcSSSfont \stcPfont \stcSPfont \slffont \sltfont \stifont

We define the fonts commands for the secttocs, sectlofs and sectlots and their titles:

```
\def\stcfont{\small\rmfamily\upshape\mdseries}    % secttoc
\def\stcSSfont{\small\rmfamily\upshape\bfseries}  % (subsections)
\let\stcSSSfont\stcfont    % (subsubsections)
\let\stcPfont\stcfont      % (paragraphs)
\let\stcSPfont\stcfont     % (subparagraphs)
\let\slffont\stcfont       % sectlof (figures)
\let\sltfont\stcfont       % sectlot (tables)
\def\stifont{\large\rmfamily\upshape\bfseries}    % titles
```

5.56 Internal macros for title positionning

Some internal macros for title positionning, from the optional arguments of `\dosecttoc` and `\secttoc` commands (and siblings):

`\l@sti` `\c@sti` `\r@sti` `\e@sti` `\n@sti`

Centering, flushleft, flushright or empty titles (with a vertical correction for empty titles, from Frank Mittelbach):

```
\def\c@sti#1{\null\hfill #1\hfill\null}
\def\l@sti#1{\null #1\hfill\null}
\def\r@sti#1{\null\hfill #1\null}
\def\e@sti#1{\vspace{-\baselineskip}}
\def\n@sti#1{\vspace{-\baselineskip}}
```

`\do@stitc` `\df@stitc` `\do@stilf` `\df@stilf` `\do@stilt` `\df@stilt` `\l@sti`

By default, titles are flushleft.

```
\let\do@stitc\l@sti
\let\df@stitc\l@sti
\let\do@stilf\l@sti
\let\df@stilf\l@sti
\let\do@stilt\l@sti
\let\df@stilt\l@sti
```

5.57 The stc@verse environment

`stc@verse` `\iftightmtc` `\ifktightmtc`

The `stc@verse` environment is a very simple list environment, analog to the standard `verse` environment. Some formatting parameters are adjusted. The `tight/loose` and `k-tight/k-loose` package options are honored.

```
\def\stc@verse{\let\\=\@centercr
  \list{}{\itemsep\z@
          \itemindent \z@
          \listparindent \itemindent
          \partopsep \z@
          \iftightmtc \parsep \z@ \fi
          \ifktightmtc \parskip \z@ \fi
          \topsep \z@
          \leftmargin\stcindent
          \rightmargin\leftmargin}\item[]}
\def\endstc@verse{\nopagebreak[4]\endlist}
```

5.58 The \secttoc, \sectlof, and \sectlot commands

These three commands are very similar.

5.58.1 The \secttoc command

\secttoc
\secttoc@

The \secttoc command must be used after \section if you need a secttoc (no automatic secttoc). Its code is similar to the code of \minitoc (but simpler). First, \secttoc detects the presence of its optional argument, and uses its default value, d, if it is missing. Then, \secttoc@ is called with the effective position as argument:

```
\def\secttoc{\@ifnextchar[{\secttoc@}{\secttoc@[d]}}
```

\secttoc@
\if@secttoc@used@
\if@longextensions@
\@tocfile
\thestc

The \secttoc@ macro does the real work. It first sets the flag \if@secttoc@used@ (for a coherence hint) and checks if long extensions are used or not (to create the name of the secttoc file):

```
\def\secttoc@[#1]{%
\global\@secttoc@used@true
\if@longextensions@%
\def\@tocfile{stc\thestc}%
\else
\def\@tocfile{S\thestc}%
\fi
```

\mtc@CkFile
\if@mtc@FE
\@tocfile

Then, we check the presence and the emptiness of the secttoc file and give a warning if it is not here or is empty:

```
        \mtc@CkFile{\jobname.\@tocfile}
        \if@mtc@FE
        \PackageInfo{minitoc}%
           {\jobname.\@tocfile\space is empty}
        \@mtc@empty@secttoc@true
        \else
```

\beforesecttoc
\thispagesecttocstyle

If the secttoc file is present and not empty, we can insert it, but we must add some presentation code: first, \beforesecttoc, of course, and the page style feature:

```
        \beforesecttoc
        \thispagesecttocstyle
```

\do@stitc \e@sti \n@sti \c@sti \l@sti \r@sti \df@sti \mtc@CkStr \stctitle \if@mtc@FE samepage

We begin a samepage environment, then treat the positionning argument. If the title is empty, we simulate the "e" positionning.

```
%%      \mtc@markboth{\uppercase{\stctitle}}{\uppercase{\stctitle}}%
        \relax\begin{samepage}%
        \if #1e\let\do@stitc\e@sti
        \else\if #1n\let\do@stitc\n@sti
        \else\if #1c\let\do@stitc\c@sti
        \else\if #1l\let\do@stitc\l@sti
        \else\if #1r\let\do@stitc\r@sti
        \else\if #1d\let\do@stitc\df@stitc
        \fi\fi\fi\fi\fi\fi
        \mtc@CkStr{\stctitle}\if@mtc@FE \let\do@stitc\e@sti\relax\fi
```

\raggedright \parskip \stcfont

We adjust some formatting parameters and avoid a page break between the title and the secttoc, then we set the font:

```
        \raggedright
        \parskip=\z@%
        \reset@font\stcfont%
        \parindent=\z@%
        \nopagebreak[4]%
```

\stc@rule tabular \stifont \do@stitc \mtc@v \stctitle \hline

The secttoc title is set in a tabular environment (to inhibit a page break between the title and the top rule), with a rule at its bottom if necessary. This rule is an \hline. It is the top rule of the secttoc.

```
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent
        \nopagebreak[4]%
        \ifx\stc@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\stifont\do@stitc{\mtc@v\stctitle}\\
        \end{tabular}%
        \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\stifont\do@stitc{\mtc@v\stctitle}\\\hline
        \end{tabular}%
        \fi
```

\mtc@zrule \mtc@BBR \stcindent

Then, we adjust the position under the top rule and set the indentation and some formatting parameters:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\\mtc@BBR
        \leftmargin\stcindent
        \rightmargin\stcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
```

stc@verse \c@tocdepth \c@secttocdepth \mtc@BBR

We enter in a stc@verse environment to format the secttoc. The toc depth is forced (locally) to secttocdepth. A little trick is necessary to adjust the position.

```
        \begin{stc@verse}\c@tocdepth=\c@secttocdepth%
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\stc@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \stc@setform \@tocfile \mtc@strut stc@verse

If the contents lines must have no numbers, we replace the macro \@dottedtocline with its undotted version. A hook is added, and the formatting settings coming from \mtcsetformat are activated via \stc@setform. Then the secttoc file is inserted, followed by a strut, and the stc@verse environment is terminated.

```
\begingroup
  \makeatletter
  \stc@setform%
  \@ifundefined{stc@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
  \@fileswfalse\mtc@hook@beforeinputfile
  \stc@setform%
  \@input{\jobname.\@tocfile}
  \vspace{-1ex} \vspace{-\baselineskip}
  \leavevmode\mtc@strut
  \global\@nobreakfalse\endgroup
        \end{stc@verse}%
```

\stc@rule \mtc@zrule samepage \aftersecttoc

The final part is just to add the bottom rule, if necessary, a possible page break and \aftersectoc.

```
        \kern-1.\baselineskip%
        \nopagebreak[4]\stc@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
        \par\pagebreak[1]\vspace*{-1ex}\aftersecttoc\fi}%
```

5.58.2 The \sectlof command

\sectlof \sectlof@

The \sectlof command must be used after \section if you need a sectlof (no automatic sectlof). Its code is similar to the code of \minilof (but simpler). First, \sectlof detects the presence of its optional argument, and uses its default value, d, if it is missing. Then, \sectlof@ is called with the effective position as argument:

```
\def\sectlof{\@ifnextchar[{\sectlof@}{\sectlof@[d]}}
```

\sectlof@ \if@sectlof@used@ \if@longextensions@ \@tocfile \thestc

The \sectlof@ macro does the real work. It first sets the flag \if@sectlof@used@ (for a coherence hint) and checks if long extensions are used or not (to create the name of the sectlof file):

```
\def\sectlof@[#1]{%
\global\@sectlof@used@true
\if@longextensions@%
\def\@tocfile{slf\thestc}%
\else
\def\@tocfile{H\thestc}%
\fi
```

\mtc@CkFile \if@mtc@FE \@tocfile

Then, we check the presence and the emptiness of the sectlof file and give a warning if it is not here or is empty:

```
       \mtc@CkFile{\jobname.\@tocfile}
       \if@mtc@FE
       \PackageInfo{minitoc}%
          {\jobname.\@tocfile\space is empty}
       \@mtc@empty@sectlof@true
       \else
```

\beforesectlof \thispagesectlofstyle

If the sectlof file is present and not empty, we can insert it, but we must add some presentation code: first, \beforesectlof, of course, and the page style feature:

```
       \thispagesectlofstyle
%%       \mtc@markboth{\uppercase{\slftitle}}{\uppercase{\slftitle}}%
       \beforesectlof
```

\do@stilf \e@sti \n@sti \c@sti \l@sti \r@sti \df@sti \mtc@CkStr \slftitle \if@mtc@FE samepage

We begin a samepage environment, then treat the positionning argument. If the title is empty, we simulate the "e" positionning.

```
       \relax\begin{samepage}%
       \if #1e\let\do@stilf\e@sti
       \else\if #1n\let\do@stilf\n@sti
       \else\if #1c\let\do@stilf\c@sti
       \else\if #1l\let\do@stilf\l@sti
       \else\if #1r\let\do@stilf\r@sti
       \else\if #1d\let\do@stilf\df@stilf
       \fi\fi\fi\fi\fi\fi
       \mtc@CkStr{\slftitle}\if@mtc@FE \let\do@stilf\e@sti\relax\fi
```

\raggedright \parskip \slffont

We adjust some formatting parameters and avoid a page break between the title and the sectlof, then we set the font:

```
       \raggedright
       \parskip=\z@%
       \reset@font\slffont%
```

```
        \parindent=\z@%
        \nopagebreak[4]%
```

\slf@rule tabular \stifont \do@stilf \mtc@v \slftitle \hline

The sectlof title is set in a `tabular` environment (to inhibit a page break between the title and the top rule), with a rule at its bottom if necessary. This rule is an `\hline`. It is the top rule of the sectlof.

```
        \kern-0.8\baselineskip\nopagebreak[4]%
        \par\noindent
        \ifx\slf@rule\relax
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \reset@font\stifont\do@stilf{\mtc@v\slftitle}\\
        \end{tabular}%
        \else
        \begin{tabular}{@{}p{\columnwidth}@{}}
        \mtc@hstrut
        \reset@font\stifont\do@stilf{\mtc@v\slftitle}\\\hline
        \end{tabular}%
        \fi
```

\mtc@zrule \mtc@BBR \stcindent

Then, we adjust the position under the top rule and set the indentation and some formatting parameters:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\\mtc@BBR
        \leftmargin\stcindent
        \rightmargin\stcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
```

stc@verse \c@tocdepth \c@sectlofdepth \mtc@BBR

We enter in a `stc@verse` environment to format the sectlof. The toc depth is forced (locally) to `sectlofdepth`. A little trick is necessary to adjust the position.

```
        \begin{stc@verse}%
        \@ifundefined{c@lofdepth}{}{\c@tocdepth=\c@sectlofdepth}
        \ifnum\c@tocdepth<1\relax\c@tocdepth=1\fi
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\slf@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \slf@setform \@tocfile \mtc@strut stc@verse

If the contents lines must have no numbers, we replace the macro `\@dottedtocline` with its undotted version. A hook is added, and the formatting settings coming from `\mtcsetformat` are activated via `\slf@setform`. Then the sectlof file is inserted, followed by a strut, and the `stc@verse` environment is terminated.

```
\begingroup
  \makeatletter
  \@ifundefined{slf@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
```

```
\@fileswfalse\mtc@hook@beforeinputfile
\slf@setform%
\@input{\jobname.\@tocfile}
\global\@nobreakfalse\endgroup
      \end{stc@verse}%
```

\stc@rule \mtc@zrule samepage \aftersecttoc

The final part is just to add the bottom rule, if necessary, a possible page break and `\afterseclof`. The blank line (\\) is essential.

```
      \kern-0.\baselineskip% ABCD
      \nopagebreak[4]\slf@rule\null\leavevmode\\%
      \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
      \par\pagebreak[1]\vspace*{-1ex}\aftersectlof\fi}%
```

5.58.3 The `\sectlot` command

\sectlot \sectlot@

The `\sectlot` command must be used after `\section` if you need a sectlot (no automatic sectlot). Its code is similar to the code of `\minilot` (but simpler). First, `\sectlot` detects the presence of its optional argument, and uses its default value, `d`, if it is missing. Then, `\sectlot@` is called with the effective position as argument:

```
\def\sectlot{\@ifnextchar[{\sectlot@}{\sectlot@[d]}}
```

\sectlot@ \if@sectlot@used@ \if@longextensions@ \@tocfile \thestc

The `\sectlot@` macro does the real work. It first sets the flag `\if@sectlot@used@` (for a coherence hint) and checks if long extensions are used or not (to create the name of the sectlot file):

```
\def\sectlot@[#1]{%
\global\@sectlot@used@true
\if@longextensions@%
\def\@tocfile{slt\thestc}%
\else
\def\@tocfile{I\thestc}%
\fi
```

\mtc@CkFile \if@mtc@FE \@tocfile

Then, we check the presence and the emptiness of the sectlot file and give a warning if it is not here or is empty:

```
      \mtc@CkFile{\jobname.\@tocfile}
      \if@mtc@FE
      \PackageInfo{minitoc}%
         {\jobname.\@tocfile\space is empty}
      \@mtc@empty@sectlot@true
      \else
```

\beforesectlot
\thispagesectlotstyle

If the sectlot file is present and not empty, we can insert it, but we must add some presentation code: first, `\beforesectlot`, of course, and the page style feature:

```
      \thispagesectlotstyle
%%    \mtc@markboth{\uppercase{\slttitle}}{\uppercase{\slttitle}}%
      \beforesectlot
```

\do@stilt
\e@sti
\n@sti
\c@sti
\l@sti
\r@sti
\df@sti
\mtc@CkStr
\slttitle
\if@mtc@FE
samepage

We begin a `samepage` environment, then treat the positionning argument. If the title is empty, we simulate the "`e`" positionning.

```
      \relax\begin{samepage}%
      \if #1e\let\do@stilt\e@sti
      \else\if #1n\let\do@stilt\n@sti
      \else\if #1c\let\do@stilt\c@sti
      \else\if #1l\let\do@stilt\l@sti
      \else\if #1r\let\do@stilt\r@sti
      \else\if #1d\let\do@stilt\df@stilt
      \fi\fi\fi\fi\fi\fi
      \mtc@CkStr{\slttitle}\if@mtc@FE \let\do@stilt\e@sti\relax\fi
```

\raggedright
\parskip
\sltfont

We adjust some formatting parameters and avoid a page break between the title and the sectlot, then we set the font:

```
      \raggedright
      \parskip=\z@%
      \reset@font\sltfont%
      \parindent=\z@%
      \nopagebreak[4]%
```

\stc@rule
tabular
\stifont
\do@stilt
\mtc@v
\slttitle
\hline

The sectlot title is set in a `tabular` environment (to inhibit a page break between the title and the top rule), with a rule at its bottom if necessary. This rule is an `\hline`. It is the top rule of the sectlot.

```
      \kern-0.8\baselineskip\nopagebreak[4]%
      \par\noindent
      \ifx\slt@rule\relax
      \begin{tabular}{@{}p{\columnwidth}@{}}
      \reset@font\stifont\do@stilt{\mtc@v\slttitle}\\
      \end{tabular}%
      \else
      \begin{tabular}{@{}p{\columnwidth}@{}}
      \mtc@hstrut
      \reset@font\stifont\do@stilt{\mtc@v\slttitle}\\\hline
      \end{tabular}%
      \fi
```

\mtc@zrule
\mtc@BBR
\stcindent

Then, we adjust the position under the top rule and set the indentation and some formatting parameters:

```
        \nopagebreak[4]\null\leavevmode\mtc@zrule\\\mtc@BBR
        \leftmargin\stcindent
        \rightmargin\stcindent
        \itemindent=\z@\labelwidth=\z@%
        \labelsep=\z@\listparindent=\z@%
```

stc@verse \c@tocdepth \c@sectlotdepth \mtc@BBR

We enter in a `stc@verse` environment to format the sectlot. The toc depth is forced (locally) to `sectlotdepth`. A little trick is necessary to adjust the position.

```
        \begin{stc@verse}%
        \@ifundefined{c@lotdepth}{}{\c@tocdepth=\c@sectlotdepth}
        \ifnum\c@tocdepth<1\relax\c@tocdepth=1\fi
        \leavevmode\\\mtc@BBR
        \vskip -.5\baselineskip
```

\slt@pgno \@dottedtocline \@undottedtocline \mtc@hook@beforeinputfile \slt@setform \@tocfile \mtc@strut stc@verse

If the contents lines must have no numbers, we replace the macro `\@dottedtocline` with its undotted version. A hook is added, and the formatting settings coming from `\mtcsetformat` are activated via `\slt@setform`. Then the sectlot file is inserted, followed by a strut, and the `stc@verse` environment is terminated.

```
\begingroup
  \makeatletter
  \@ifundefined{slt@pgno}%
  {\let\@dottedtocline\@undottedtocline}{}
   \gdef\thestc{\arabic{stc}}
  \@fileswfalse\mtc@hook@beforeinputfile
  \slt@setform%
  \@input{\jobname.\@tocfile}
  \global\@nobreakfalse\endgroup
        \end{stc@verse}%
```

\stc@rule \mtc@zrule samepage \aftersectlot

The final part is just to add the bottom rule, if necessary, a possible page break and `\afterseclot`.

```
        \kern-0.\baselineskip% ABCD
        \nopagebreak[4]\slt@rule\null\leavevmode\\%
        \vskip-1.0\baselineskip\mtc@zrule\end{samepage}%
        \par\pagebreak[1]\vspace*{-1ex}\aftersectlot\fi}%
```

5.59 Auxiliary internal commands, section level

\l@xsect \@dottedtocline \l@schapter \xsect \schapter

We define auxiliary commands, used for the mini-tables and as delimiters in the TOC file (and LOF and LOT files). The depth of `xsect` is huge to inhibit the printing of its contents line (except if you cheat).

```
\def\l@xsect{\@dottedtocline{\@M}{1.0em}{2.3em}}
\def\l@schapter{\@dottedtocline{1}{1.0em}{2.3em}}
\def\xsect{xsect}
\def\schapter{schapter}
```

5.60 Patching the `\section` command (continued)

\@sect \addcontentsline

We patch the both branches of the `\section` command: `\@sect` for the unstarred version and `\@ssect` for the starred version. First, for the unstarred version (`\@sect`), we add a `xsect` contents line in the LOF and in the LOT. The test `\ifnum #2=1` restricts the action to the section level macros (because `\@sect` is also used by `\subsection` and below, which have no mini-tables).

```
\let\sv@sect\@sect
\gdef\@sect#1#2#3#4#5#6[#7]#8{%
\ifnum #2=1
   \addcontentsline{lof}{xsect}{#7}%
   \addcontentsline{lot}{xsect}{#7}%
\fi
\sv@sect{#1}{#2}{#3}{#4}{#4}{#5}{#6}[{#7}]{#8}}
```

\section \starsection \addcontentsline

If it is a section (unstarred or starred via `\starsection`), we add a `xsect` entry in the LOF and in the LOT.

```
\def\@sect#1#2#3#4#5#6[#7]#8{
\expandafter
\ifx\csname #1\endcsname\section\relax
   \addcontentsline{lof}{xsect}{#7}%
   \addcontentsline{lot}{xsect}{#7}%
\fi
\ifx\csname #1\endcsname\starsection\relax
   \addcontentsline{lof}{xsect}{#7}%
   \addcontentsline{lot}{xsect}{#7}%
\fi
```

\@svsec \refstepcounter \@tempskipa \@hangfrom \addcontentsline \numberline \@svsechd \@xsect

And the remainder of the section header formatting:

```
\ifnum #2>\c@secnumdepth
   \let\@svsec\@empty
```

```
\else
   \refstepcounter{#1}%
   \edef\@svsec{\csname the#1\endcsname\hskip 1em}%
\fi
\@tempskipa #5\relax
\ifdim \@tempskipa>\z@
   \begingroup #6\relax
      \@hangfrom{\hskip #3\relax\@svsec}%
         {\interlinepenalty \@M #8\par}%
   \endgroup
   \csname #1mark\endcsname{#7}\addcontentsline
   {toc}{#1}{\ifnum #2>\c@secnumdepth
             \else
             \protect\numberline{\csname the#1\endcsname}%
             \fi
             #7}%
\else
   \def\@svsechd{#6\hskip #3\relax
   \@svsec #8\csname #1mark\endcsname
   {#7}\addcontentsline
   {toc}{#1}{\ifnum #2>\c@secnumdepth
             \else
                \protect\numberline{\csname the#1\endcsname}
             \fi
   #7}}%
\fi
\@xsect{#5}}
```

\@ssect \sectbegin \sectend \stc@ssect \addtocontents

Then we patch the starred branch (\@ssect). We define also the delimiting commands \sectbegin and \sectend commands.

```
\let\stc@ssect\@ssect
\def\@ssect{\addtocontents{toc}{\protect\sectend}\stc@ssect}
\def\@ssect{\addtocontents{toc}{\protect\sectbegin}\stc@ssect}
\let\sectend\relax
\let\sectbegin\relax
```

5.61 The \dosecttoc command and siblings

The \dosectoc command is very similar to \dominitoc.

\dosecttoc \@dosecttoc \STC@next

The \dosecttoc command extracts information from the .toc file and creates the .stc⟨N⟩ files (.stc becomes .S on MS-DOS).

```
\def\@dosecttoc#1{{%
  \makeatletter
  \setcounter{stc}{0}
  \STC@next#1.toc\relax\\}\setcounter{stc}{0}}
```

\dosectlof \@dosectlof \SLF@next

The `\dosectlof` command extracts information from the `.lof` file and creates the `.slf`⟨*N*⟩ files (`.slf` becomes `.H` on MS-DOS).

```
\def\@dosectlof#1{{%
  \makeatletter
  \setcounter{stc}{0}
  \SLF@next#1.lof\relax\\}\setcounter{stc}{0}}
```

\dosectlot \@dosectlot \PLT@next

The `\dosectlot` command extracts information from the `.lot` file and creates the `.slt`⟨*N*⟩ files (`.slt` becomes `.V` on MS-DOS).

```
\def\@dosectlot#1{{%
  \makeatletter
  \setcounter{stc}{0}
  \SLT@next#1.lot\relax\\}\setcounter{stc}{0}}
```

\dosecttoc \dosectlof \dosectlot

We define the user-level macros, who detect the optional argument:

```
\def\dosecttoc{\@ifnextchar[{\dosecttoc@}{\dosecttoc@[l]}}
\def\dosectlof{\@ifnextchar[{\dosectlof@}{\dosectlof@[l]}}
\def\dosectlot{\@ifnextchar[{\dosectlot@}{\dosectlot@[l]}}
```

\dosecttoc@ \if@dosecttoc@used@ \df@stitc \e@sti \n@sti \c@sti \l@sti \r@sti

We treat the optional argument of `\dosecttoc` (it becomes the default position for titles of secttocs) and flag this macro as used:

```
\def\dosecttoc@[#1]{%
\global\@dosecttoc@used@true
\if #1e\let\df@stitc\e@sti%
\else\if #1n\let\df@stitc\n@sti%
\else\if #1c\let\df@stitc\c@sti%
\else\if #1l\let\df@stitc\l@sti%
\else\if #1r\let\df@stitc\r@sti%
\fi\fi\fi\fi\fi%
\@@dosecttoc}
```

\dosectlof@ \if@dosectlof@used@ \df@stilf \e@sti \n@sti \c@sti \l@sti \r@sti

We treat the optional argument of `\dosectlof` (it becomes the default position for titles of sectlofs) and flag this macro as used:

```
\def\dosectlof@[#1]{%
\global\@dosectlof@used@true
\if #1e\let\df@stilf\e@sti%
\else\if #1n\let\df@stilf\n@sti%
\else\if #1c\let\df@stilf\c@sti%
\else\if #1l\let\df@stilf\l@sti%
\else\if #1r\let\df@stilf\r@sti%
\fi\fi\fi\fi\fi%
\@@dosectlof}
```

\dosectlot@ \if@dosectlot@used@ \df@stilt \e@sti \n@sti \c@sti \l@sti \r@sti

We treat the optional argument of `\dosectlot` (it becomes the default position for titles of sectlofs) and flag this macro as used:

```
\def\dosectlot@[#1]{%
\global\@dosectlot@used@true
\if #1e\let\df@stilt\e@sti%
\else\if #1n\let\df@stilt\n@sti%
\else\if #1c\let\df@stilt\c@sti%
\else\if #1l\let\df@stilt\l@sti%
\else\if #1r\let\df@stilt\r@sti%
\fi\fi\fi\fi\fi%
\@@dosectlot}
```

\@@dosecttoc \@@dosectlof \@@dosectlot \tf@mtc

These macros invoke the `\@dosect...` commands to create the mini-table file, then close the file descriptor.

```
\def\@@dosecttoc{\@dosecttoc{\jobname}\immediate\closeout\tf@mtc}
\def\@@dosectlof{\@dosectlof{\jobname}\immediate\closeout\tf@mtc}
\def\@@dosectlot{\@dosectlot{\jobname}\immediate\closeout\tf@mtc}
```

\STC@next \STC@list \STC@loop

Processing the next entry in the list and remove it from the head of the list:

```
\def\STC@next#1\relax#2\\{%
  \edef\STC@list{#2}%
  \STC@loop{#1}}
```

\STC@toc \STC@list \STC@explist

Check if the list is empty:

```
\def\STC@toc{%
  \ifx\STC@list\@empty\else\expandafter\STC@explist\fi}
```

\STC@contentsline \section \thestc \tf@mtc \stcname \MTC@WriteContentsLine

The macro `\STC@contentsline` analyses the lines read from the TOC file. If `\section` is found, `stc` is incremented and a new secttoc file is created.

```
\def\STC@contentsline#1#2#3#4{% %%HO: #4 added
  \gdef\thestc{\arabic{stc}}% %%HO: space removed
  \expandafter\ifx\csname #1\endcsname\section
    \stepcounter{stc}%
%%    \def\thestc{\arabic{stc}} %% HO: removed
```

\if@longextensions@ \stcname

We test if long or short extensions are used, to build the name of the mini-table file, then open it:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.stc\thestc}%
      \def\stcname{\jobname.stc\thestc}%
```

```
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.S\thestc}%
      \def\stcname{\jobname.S\thestc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\stcname
  \fi
```

\mtc@toks \MTC@WriteCoffeeline \MTC@WriteContentsline \coffee \subsection \subsubsection \paragraph \subparagraph

Now, we filter the relevant contents lines:

```
  \mtc@toks{\noexpand\leavevmode #2}%
  \expandafter\ifx\csname #1\endcsname\coffee
    \MTC@WriteCoffeeline{#1}{#3}%
  \fi
  \expandafter\ifx\csname #1\endcsname\subsection
    \MTC@WriteContentsline{#1}{stcSS}{#3}{#4}%
  \fi
  \expandafter\ifx\csname #1\endcsname\subsubsection
    \MTC@WriteContentsline{#1}{stcSSS}{#3}{#4}%
  \fi
  \expandafter\ifx\csname #1\endcsname\paragraph
    \MTC@WriteContentsline{#1}{stcP}{#3}{#4}%
  \fi
  \expandafter\ifx\csname #1\endcsname\subparagraph
    \MTC@WriteContentsline{#1}{stcSP}{#3}{#4}%
  \fi
```

\starsection \stepcounter \thestc \arabic \if@longextensions@ \stcname \closeout \openout

A starred section terminates the current section and creates a new secttoc file:

```
  \ifx\csname #1\endcsname\starsection
    \stepcounter{stc}%
    \gdef\thestc{\arabic{stc}}
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.stc\thestc}%
      \def\stcname{\jobname.stc\thestc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.S\thestc}%
      \def\stcname{\jobname.S\thestc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\stcname
  \fi
```

\MTC@WriteContentsline \starsubsection \starsubsubsection \starparagraph \starsubparagraph

We process the entries for starred sectionning commands:

```
  \expandafter\ifx\csname #1\endcsname\starsubsection
    \MTC@WriteContentsline{#1}{stcSS}{#3}{#4}%
```

```
  \fi
  \expandafter\ifx\csname #1\endcsname\starsubsubsection
    \MTC@WriteContentsline{#1}{stcSSS}{#3}{#4}%
  \fi
  \expandafter\ifx\csname #1\endcsname\starparagraph
    \MTC@WriteContentsline{#1}{stcP}{#3}{#4}%
  \fi
  \expandafter\ifx\csname #1\endcsname\starsubparagraph
    \MTC@WriteContentsline{#1}{stcSP}{#3}{#4}%
  \fi
}
```

\STC@explist \STC@next \STC@list

The loop to read the lines of the TOC file; expands the list of entries and call `\STC@next` to process the first one.

```
\def\STC@explist{\expandafter\STC@next\STC@list\\}
```

\STC@loop \STC@toc \STC@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\STC@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1\MessageBreak SECTTOCS NOT PREPARED}%
    \expandafter\STC@toc
  \else
    \PackageInfo{minitoc}%
       {PREPARING SECTTOCS FROM #1}%
    \expandafter\STC@read\fi}
```

\STC@read \STC@line

Read the next entry of the `.toc` file.

```
\def\STC@read{%
  \read\@inputcheck to\STC@line
```

\STC@test \STC@line

The `.....` make sure that `\STC@test` has sufficient arguments:

```
  \expandafter\STC@test\STC@line.....\STC@% %%HO: . added
  }%
```

\STC@test \STC@contentsline \mtc@string \STC@list \STC@toc \STC@read \sectend \sectbegin

The `\STC@test` macro finds the "interesting" commands in the TOC file, mainly to delimit sections;

```
\long\def\STC@test#1#2#3#4#5#6\STC@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \STC@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
```

```
    \let\mtc@string\relax
  \else\ifx#1\@input
    \edef\STC@list{\STC@list#2\relax}%
  \else\ifx#1\sectend
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\sectbegin
    \addtocounter{stc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\STC@toc
  \else\expandafter\STC@read\fi}%
```

\SLF@next \SLF@list \SLF@loop

Processing the next entry in the list and remove it from the head of the list:

```
\def\SLF@next#1\relax#2\\{%
  \edef\SLF@list{#2}%
  \SLF@loop{#1}}
```

\SLF@lof \SLF@list \SLF@explist

Check if the list is empty:

```
\def\SLF@lof{%
  \ifx\SLF@list\@empty\else\expandafter\SLF@explist\fi}
```

\SLF@contentsline \section \thestc \tf@mtc \slfname \MTC@WriteContentsLine

The macro `\SLF@contentsline` analyses the lines read from the LOF file. If `\section` is found, `stc` is incremented and a new sectlof file is created.

```
\def\SLF@contentsline#1#2#3#4{% %%HO: #4 added
  \gdef\thestc{\arabic{stc}}% %%HO: space removed
  \expandafter\ifx\csname #1\endcsname\xsect
    \stepcounter{stc}%
%%    \gdef\thestc{\arabic{stc}} %%HO: removed
```

\if@longextensions@ \slfname

We test if long or short extensions are used, to build the name of the mini-table file, then open it:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
        {Writing\space\jobname.slf\thestc}%
      \def\slfname{\jobname.slf\thestc}%
    \else
      \PackageInfo{minitoc}%
        {Writing\space\jobname.H\thestc}%
      \def\slfname{\jobname.H\thestc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\slfname
  \fi
```

\mtc@toks \MTC@WriteContentsline \figure

Now, we filter the relevant contents lines:

```
  \mtc@toks{\noexpand\leavevmode #2}%
  \expandafter\ifx\csname #1\endcsname\figure
    \MTC@WriteContentsline{#1}{slf}{#3}{#4}%
  \fi
}
```

\SLF@explist \SLF@next \SLF@list

The loop to read the lines of the LOF file; expands the list of entries and call `\SLF@next` to process the first one.

```
\def\SLF@explist{\expandafter\SLF@next\SLF@list\\}
```

\SLF@loop \SLF@lof \SLF@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\SLF@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1\MessageBreak SECTLOFS NOT PREPARED}%
    \expandafter\SLF@lof
  \else
    \PackageInfo{minitoc}%
       {PREPARING SECTLOFS FROM #1}%
    \expandafter\SLF@read\fi}
```

Read the next entry of the `.lof` file.

\SLF@read \SLF@test \SLF@line

The `.....` make sure that `\SLF@test` has sufficient arguments:

```
\def\SLF@read{%
  \read\@inputcheck to\SLF@line
  \expandafter\SLF@test\SLF@line.....\SLF@% %%HO: . added
  }%
```

\SLF@test \SLF@contentsline \mtc@string \SLF@list \SLF@lof \SLF@read \sectend \sectbegin

The `\SLF@test` macro finds the “interesting” commands in the LOF file, mainly to delimit sections;

```
\long\def\SLF@test#1#2#3#4#5#6\SLF@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \SLF@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
    \let\mtc@string\relax
  \else\ifx#1\@input
     \edef\SLF@list{\SLF@list#2\relax}%
  \else\ifx#1\sectend
     \immediate\closeout\tf@mtc
     \immediate\openout\tf@mtc=\jobname.mtc
```

```
  \else\ifx#1\sectbegin
     \addtocounter{stc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\SLF@lof
  \else\expandafter\SLF@read\fi}%
```

\SLT@next \SLT@list \SLT@loop

Processing the next entry in the list and remove it from the head of the list:

```
\def\SLT@next#1\relax#2\\{%
  \edef\SLT@list{#2}%
  \SLT@loop{#1}}
```

\SLT@lot \SLT@list \SLT@explist

Check if the list is empty:

```
\def\SLT@lot{%
  \ifx\SLT@list\@empty\else\expandafter\SLT@explist\fi}
```

\SLT@contentsline \section \thestc \tf@mtc \sltname \MTC@WriteContentsLine

The macro `\SLT@contentsline` analyses the lines read from the LOT file. If `\section` is found, `stc` is incremented and a new sectlot file is created.

```
\def\SLT@contentsline#1#2#3#4{% %%HO: #4 added
  \gdef\thestc{\arabic{stc}}% %%HO: space removed
  \expandafter\ifx\csname #1\endcsname\xsect
    \stepcounter{stc}%
%%    \gdef\thestc{\arabic{stc}} %%HO: removed
```

\if@longextensions@ \sltname

We test if long or short extensions are used, to build the name of the mini-table file, then open it:

```
    \if@longextensions@%
      \PackageInfo{minitoc}%
         {Writing\space\jobname.slt\thestc}%
      \def\sltname{\jobname.slt\thestc}%
    \else
      \PackageInfo{minitoc}%
         {Writing\space\jobname.V\thestc}%
      \def\sltname{\jobname.V\thestc}%
    \fi
    \immediate\closeout\tf@mtc
    \immediate\openout\tf@mtc=\sltname
  \fi
```

\mtc@toks \MTC@WriteContentsline \table

Now, we filter the relevant contents lines:

```
  \mtc@toks{\noexpand\leavevmode #2}%
  \expandafter\ifx\csname #1\endcsname\table
    \MTC@WriteContentsline{#1}{slt}{#3}{#4}%
```

```
  \fi
}
```

\SLT@explist \SLT@next \SLT@list

The loop to read the lines of the LOT file; expands the list of entries and call \SLT@next to process the first one.

```
\def\SLT@explist{\expandafter\SLT@next\SLT@list\\}
```

\SLT@loop \SLT@lot \SLT@read

If an entry is found, loop through line by line, looking for interesting entries. Otherwise, process the next entry in the list.

```
\def\SLT@loop#1{\openin\@inputcheck#1\relax
  \ifeof\@inputcheck
    \PackageWarning{minitoc}%
       {No file #1\MessageBreak SECTLOTS NOT PREPARED}%
    \expandafter\SLT@lot
  \else
    \PackageInfo{minitoc}%
       {PREPARING SECTLOTS FROM #1}%
    \expandafter\SLT@read\fi}
```

Read the next entry of the .lot file.

\SLT@read \SLT@test \SLT@line

The ..... make sure that \SLT@test has sufficient arguments:

```
\def\SLT@read{%
  \read\@inputcheck to\SLT@line
  \expandafter\SLT@test\SLT@line.....\SLT@% %%HO: . added
  }%
```

\SLT@test \SLT@contentsline \mtc@string \SLT@list \SLT@lot \SLT@read \sectend \sectbegin

The \SLT@test macro finds the “interesting” commands in the LOT file, mainly to delimit sections;

```
\long\def\SLT@test#1#2#3#4#5#6\SLT@{% %%HO: #6 added
  \ifx#1\contentsline
    \let\mtc@string\string
    \SLT@contentsline{#2}{#3}{#4}{#5}% %%HO: #4 added
    \let\mtc@string\relax
  \else\ifx#1\@input
     \edef\SLT@list{\SLT@list#2\relax}%
  \else\ifx#1\sectend
     \immediate\closeout\tf@mtc
     \immediate\openout\tf@mtc=\jobname.mtc
  \else\ifx#1\sectbegin
     \addtocounter{stc}{-1}%
  \fi\fi\fi\fi
  \ifeof\@inputcheck\expandafter\SLT@lot
  \else\expandafter\SLT@read\fi}%
```

5.62 End of section-level commands

We terminate the *else* branch of the test `\@ifundefined{section}`, the *true* branch of the test `\@ifundefined{chapter}` and add an empty *else* branch to the test `\@ifundefined{chapter}`:

```
\fi% end of \if@mtc@section@def@
\fi% end of \if@mtc@chapter@undef@
```

5.63 Necessary `\l@...` commands

`\l@listof` `\l@starpart` `\l@starchapter` `\l@starsection` `\l@starsubsection` `\l@starsubsubsection` `\l@starparagraph` `\l@starsubparagraph`

Some `\l@...` commands (analog to `\l@section` or `\l@paragraph`) are required to format some entries in the mini-tables, for starred sectionning commands essentially:

```
\@ifundefined{section}{}{\let\l@listof\l@section}
\@ifundefined{chapter}{}{\let\l@listof\l@chapter}
\@ifundefined{part}{}{\let\l@starpart\l@part}
\@ifundefined{chapter}{}{\let\l@starchapter\l@chapter}
\@ifundefined{section}{}{\let\l@starsection\l@section}
\@ifundefined{subsection}{}{\let\l@starsubsection\l@subsection}
\@ifundefined{subsubsection}{}{\let\l@starsubsubsection\l@subsubsection}
\@ifundefined{paragraph}{}{\let\l@starparagraph\l@paragraph}
\@ifundefined{subparagraph}{}{\let\l@starsubparagraph\l@subparagraph}
```

`\noptcrule` `\nomtcrule` `\nostcrule` `\ptcrule` `\mtcrule` `\stcrule` `\noplfrule` `\nomlfrule` `\noslfrule` `\plfrule` `\mlfrule` `\slfrule` `\nopltrule` `\nomltrule` `\nosltrule` `\pltrule` `\mltrule` `\sltrule`

5.64 The horizontal rules and their default values

We define here the various commands to activate ou inhibit the horizontal rules in the various kinds of mini-tables. Each such command is an indirect definition of the corresponding horizontal rule. The rules are .4 pt high horizontal rules.

```
\def\noptcrule{\let\ptc@rule\relax}
\def\nomtcrule{\let\mtc@rule\relax}
\def\nostcrule{\let\stc@rule\relax}
\def\ptcrule{\def\ptc@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\mtcrule{\def\mtc@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\stcrule{\def\stc@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\ptc@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
\def\mtc@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
\def\stc@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
```

```
\def\noplfrule{\let\plf@rule\relax}
\def\nomlfrule{\let\mlf@rule\relax}
\def\noslfrule{\let\slf@rule\relax}
```

```
\def\plfrule{\def\plf@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\mlfrule{\def\mlf@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\slfrule{\def\slf@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\plf@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
\def\mlf@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
\def\slf@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
```

```
\def\nopltrule{\let\plt@rule\relax}
\def\nomltrule{\let\mlt@rule\relax}
\def\nosltrule{\let\slt@rule\relax}
\def\pltrule{\def\plt@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\mltrule{\def\mlt@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\sltrule{\def\slt@rule{\kern-3\p@ \hrule width \columnwidth \kern2.6\p@}}
\def\plt@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
\def\mlt@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
\def\slt@rule{\rule[3\p@]{\columnwidth}{.4\p@}\vspace*{2.6\p@}}
```

5.65 The `\mtcset...` commands

These commands[7] have been introduced to build a nicer user interface, and each of them replaces numerous user commands, offering a rather unified and logical syntax.

5.65.1 Keywords for the `\mtcset...` commands

`\@namedef` `\@nameuse`

We define some common keywords for the `\mtcset...` commands. A keyword is created via the `\@namedef` – `\@nameuse` mechanism the following way:

`\@namedef{mtc@`*family*`@`*name*`}{`*abbreviation*`}`

where *family* is the name of a group of keywords relative to one or several `\mtcset...` macros, *name* is the keyword that the user gives as argument to the `\mtcset...` macro, and *abbreviation* is a string used to build the name of the macro effectively used. As some `\mtcset...` macros have several keyword parameters, this method can reduce the number of macros at the user level, at the cost of few keyword families.

`\@namedef`

We define a family (`typetable`) of keywords for the types of mini-tables:

```
\@namedef{mtc@typetable@parttoc}{ptc}\def\mtc@typetable@parttoc{ptc}
\@namedef{mtc@typetable@partlot}{plt}\def\mtc@typetable@partlot{plf}
\@namedef{mtc@typetable@partlof}{plf}\def\mtc@typetable@partlof{plt}
\@namedef{mtc@typetable@minitoc}{mtc}\def\mtc@typetable@minitoc{mtc}
\@namedef{mtc@typetable@minilof}{mlf}\def\mtc@typetable@minilof{mlf}
```

[7] The general concept of the `\mtcset...` commands was proposed by Benjamin Bayart.

```
\@namedef{mtc@typetable@minilot}{mlt}\def\mtc@typetable@minilot{mlt}
\@namedef{mtc@typetable@secttoc}{stc}\def\mtc@typetable@secttoc{stc}
\@namedef{mtc@typetable@sectlof}{slf}\def\mtc@typetable@sectlof{slf}
\@namedef{mtc@typetable@sectlot}{slt}\def\mtc@typetable@sectlot{slt}
```

\@namedef Then another family (typetitle) for the titles of the mini-tables:

```
\@namedef{mtc@typetitle@parttoc}{pti}\def\mtc@typetitle@parttoc{pti}
\@namedef{mtc@typetitle@partlof}{pti}\def\mtc@typetitle@partlof{pti}
\@namedef{mtc@typetitle@partlot}{pti}\def\mtc@typetitle@partlot{pti}
\@namedef{mtc@typetitle@minitoc}{mti}\def\mtc@typetitle@minitoc{mti}
\@namedef{mtc@typetitle@minilof}{mti}\def\mtc@typetitle@minilof{mti}
\@namedef{mtc@typetitle@minilot}{mti}\def\mtc@typetitle@minilot{mti}
\@namedef{mtc@typetitle@secttoc}{sti}\def\mtc@typetitle@secttoc{sti}
\@namedef{mtc@typetitle@sectlof}{sti}\def\mtc@typetitle@sectlof{sti}
\@namedef{mtc@typetitle@sectlot}{sti}\def\mtc@typetitle@sectlot{sti}
```

\@namedef We define a family (YN) of keywords to recognize the keywords "off" and "on", with their many synonyms [8] and meaning false or true:

```
\@namedef{mtc@YN@off}{no}\def\mtc@YN@off{no}
\@namedef{mtc@YN@OFF}{no}\def\mtc@YN@OFF{no}
\@namedef{mtc@YN@no}{no}\def\mtc@YN@no{no}
\@namedef{mtc@YN@NO}{no}\def\mtc@YN@NO{no}
\@namedef{mtc@YN@n}{no}\def\mtc@YN@n{no}
\@namedef{mtc@YN@N}{no}\def\mtc@YN@N{no}
\@namedef{mtc@YN@false}{no}\def\mtc@YN@false{no}
\@namedef{mtc@YN@FALSE}{no}\def\mtc@YN@FALSE{no}
\@namedef{mtc@YN@faux}{no}\def\mtc@YN@faux{no}
\@namedef{mtc@YN@FAUX}{no}\def\mtc@YN@FAUX{no}
\@namedef{mtc@YN@f}{no}\def\mtc@YN@f{no}
\@namedef{mtc@YN@F}{no}\def\mtc@YN@F{no}
\@namedef{mtc@YN@NON}{no}\def\mtc@YN@NON{no}
\@namedef{mtc@YN@non}{no}\def\mtc@YN@non{no}
\@namedef{mtc@YN@0}{no}\expandafter\def\csname mtc@YN@0\endcsname{no}
\@namedef{mtc@YN@on}{}\def\mtc@YN@on{}
\@namedef{mtc@YN@ON}{}\def\mtc@YN@ON{}
\@namedef{mtc@YN@yes}{}\def\mtc@YN@yes{}
\@namedef{mtc@YN@YES}{}\def\mtc@YN@YES{}
\@namedef{mtc@YN@y}{}\def\mtc@YN@y{}
\@namedef{mtc@YN@Y}{}\def\mtc@YN@Y{}
\@namedef{mtc@YN@true}{}\def\mtc@YN@true{}
\@namedef{mtc@YN@TRUE}{}\def\mtc@YN@TRUE{}
\@namedef{mtc@YN@t}{}\def\mtc@YN@t{}
\@namedef{mtc@YN@T}{}\def\mtc@YN@T{}
\@namedef{mtc@YN@vrai}{}\def\mtc@YN@vrai{}
\@namedef{mtc@YN@VRAI}{}\def\mtc@YN@VRAI{}
\@namedef{mtc@YN@v}{}\def\mtc@YN@v{}
```

[8] This (extreme) case shows the easyness for creating synonyms of frequently used keywords. Note also that when a keyword contains a non-letter character, we must use a hack with \expandafter \csname ... \endcsname.

```
\@namedef{mtc@YN@V}{}\def\mtc@YN@TRUEV
\@namedef{mtc@YN@OUI}{}\def\mtc@YN@OUI{}
\@namedef{mtc@YN@oui}{}\def\mtc@YN@oui{}
\@namedef{mtc@YN@O}{}\def\mtc@YN@O{}
\@namedef{mtc@YN@o}{}\def\mtc@YN@o{}
\@namedef{mtc@YN@1}{}\expandafter\def\csname mtc@YN@1\endcsname{}
```

5.65.2 The `\mtcsetfont` command

`\@namedef` We define the sectionning level keywords (note that `part` is not a member of this family (`sectlevel`), because no contents line for a part can appear in a mini-table, part being the highest sectionning level); "*" represents "any level", and is used to set the global default font for a given kind of mini-table.

```
\@namedef{mtc@sectlevel@chapter}{C}\def\mtc@sectlevel@chapter{C}
\@namedef{mtc@sectlevel@section}{S}\def\mtc@sectlevel@section{S}
\@namedef{mtc@sectlevel@subsection}{SS}\def\mtc@sectlevel@subsection{SS}
\@namedef{mtc@sectlevel@subsubsection}{SSS}\def\mtc@sectlevel@subsubsection{SSS}
\@namedef{mtc@sectlevel@paragraph}{P}\def\mtc@sectlevel@paragraph{P}
\@namedef{mtc@sectlevel@subparagraph}{SP}\def\mtc@sectlevel@subparagraph{SP}
\@namedef{mtc@sectlevel@*}{}\expandafter\def\csname mtc@sectlevel@*\endcsname{}
```

`\mtcsetfont` The `\mtcsetfont` command has the following syntax:

`\mtcsetfont`{*mini-table*}{*level-name*}{*font commands*}

The *mini-table* type is a keyword like `minitoc`, the *level-name* is a sectionning level like `subsection` (no backslash). The *font commands* are a font specification, using NFSS [28] basic commands usually.

`\if@mtc@setfont@` First, we declare a flag, set true:

```
\newif\if@mtc@setfont@\@mtc@setfont@true
```

`\mtcsetfont` Then, we begin the command, which has three arguments:

```
\newcommand{\mtcsetfont}[3]{%
```

`\mtc@mta@abbrev` `\if@mtc@setfont@` `\@nameuse` The two first arguments of this command are keywords. They must be translated into the effective strings. We process the first argument which is a keyword from the `typetable` family. The result is stored in `\mtc@mta@abbrev`. Example: if `#1` is `minitoc`, we get `mtc`.

```
\def\mtc@mta@abbrev{X}
\@mtc@setfont@true
\expandafter\ifx\csname mtc@typetable@#1\endcsname\relax
   \@mtc@setfont@false
   \def\mtc@mta@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsetfont \space has a wrong first argument (#1).
       \MessageBreak
       It should be a mini-table type (parttoc...sectlot)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}
\else
   \edef\mtc@mta@abbrev{\@nameuse{mtc@typetable@#1}}
\fi
```

`\mtc@level@abbrev` `\if@mtc@setfont@` `\@nameuse`

The second argument, a keyword from the family `sectlevel`, is processed the same way and the result is stored into `\mtc@level@abbrev`. Example: if `#2` is `subsection`, we get SS.

```
\def\mtc@level@abbrev{X}
\expandafter\ifx\csname mtc@sectlevel@#2\endcsname\relax
   \@mtc@setfont@false
   \def\mtc@level@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsetfont \space has a wrong second argument (#2).
       \MessageBreak
       It should be a sectionning level (part...subparagraph) or * }%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}
\else
   \edef\mtc@level@abbrev{\@nameuse{mtc@sectlevel@#2}}
\fi
```

`\mtc@tmp@name` `\mtc@mta@abbrev` `\mtc@level@abbrev`

Then, we construct the effective macro to be applied:

```
\def\mtc@tmp@name{\mtc@mta@abbrev\mtc@level@abbrev font}
```

Example: if `#1` is `minitoc` and `#2` is `subsection`, we get `mtcSSfont`, which is the name of the command for the font of a subsection entry in a minitoc (the backslash is missing, but we will use a `\csname` … `\endcsname` pair to apply the constructed command).

`\if@mtc@setfont@`

But all combinaisons are not legal (the level of the entry must be lower than the level of the mini-table), so we must test:

```
\expandafter\ifx\csname #1\endcsname\minitoc
   \expandafter\ifx\csname #2\endcsname\part\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\chapter\@mtc@setfont@false\fi
\fi
\expandafter\ifx\csname #1\endcsname\minilof
```

```
   \expandafter\ifx\csname #2\endcsname\part\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\chapter\@mtc@setfont@false\fi
\fi
\expandafter\ifx\csname #1\endcsname\minilot
   \expandafter\ifx\csname #2\endcsname\part\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\chapter\@mtc@setfont@false\fi
\fi
\expandafter\ifx\csname #1\endcsname\secttoc
   \expandafter\ifx\csname #2\endcsname\part\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\chapter\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\section\@mtc@setfont@false\fi
\fi
\expandafter\ifx\csname #1\endcsname\sectlof
   \expandafter\ifx\csname #2\endcsname\part\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\chapter\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\section\@mtc@setfont@false\fi
\fi
\expandafter\ifx\csname #1\endcsname\sectlot
   \expandafter\ifx\csname #2\endcsname\part\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\chapter\@mtc@setfont@false\fi
   \expandafter\ifx\csname #2\endcsname\section\@mtc@setfont@false\fi
\fi
```

\if@mtc@setfont@
\mtc@tmp@name
\mtc@mta@abbrev
\mtc@level@abbrev
\mtc@toks

If the combinaison is legal, we apply it, *i.e.* we redefine the meaning of the constructed macro with the sequence of commands given as third argument of `\mtcsetfont` and we log that event (we store the third argument in a token register to can print it *verbatim*).

If the combinaison is not legal, an error message is displayed.

```
\if@mtc@setfont@
   \def\mtc@tmp@name{\mtc@mta@abbrev\mtc@level@abbrev font}
   \mtc@toks{#3}
   \PackageInfo{minitoc}%
      {\string\mtcsetfont\space redefines the macro
        \mtc@tmp@name \space as "\the\mtc@toks"}
   \expandafter\def\csname\mtc@tmp@name\endcsname{#3}
\else
   \PackageError{minitoc}%
      {\string\mtcsetfont\space has incompatible\MessageBreak
       first (#1) and second (#2) arguments}{}
\fi
}
```

5.65.3 The `\mtcsettitlefont` command

\mtcsettitlefont

This command is very similar to the `\mtcsetfont` command. Its syntax is almost identical:

\mtcsettitlefont{*mini-table*}{*font commands*}

\if@mtc@settitlefont@

The *mini-table* type is a keyword like minitoc. The *font commands* are a font specification, using NFSS [28] basic commands usually. The difference is the absence of the second keyword argument, because the *font commands* will be applied to the title of each mini-table of the given kind.

First, we declare a flag, set true:

```
\newif\if@mtc@settitlefont@\@mtc@settitlefont@true
```

\mtcsettitlefont

And we begin the definition of the \mtcsettitlefont command, which has two arguments:

```
\newcommand{\mtcsettitlefont}[2]{%
```

\mtc@mtatf@abbrev
\if@mtc@settitlefont@
\@nameuse

We process the first argument, a keyword of the typetitle family, then the result is stored into \mtc@mtatf@abbrev:

```
\def\mtc@mtatf@abbrev{X}
\@mtc@settitlefont@true
\expandafter\ifx\csname mtc@typetitle@#1\endcsname\relax
   \@mtc@settitlefont@false
   \def\mtc@mtatf@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsettitlefont \space has a wrong first argument (#1).
       \MessageBreak
       It should be a mini-table type (parttoc...sectlot)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}
\else
   \edef\mtc@mtatf@abbrev{\@nameuse{mtc@typetitle@#1}}
\fi
```

\if@mtc@settitlefont@
\mtc@tmptf@name
\mtc@mtatf@abbrev
\mtc@toks

Then we build the name of the effective command and apply this command:

```
\if@mtc@settitlefont@
   \def\mtc@tmptf@name{\mtc@mtatf@abbrev font}
   \mtc@toks{#2}
   \PackageInfo{minitoc}%
      {\string\mtcsettitlefont\space redefines the macro
       \mtc@tmptf@name \space as "\the\mtc@toks"}
   \expandafter\def\csname\mtc@tmptf@name\endcsname{#2}
\else
   \PackageError{minitoc}%
      {Illegal type of table (#1)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}{\relax}
```

```
\fi
}
```

5.65.4 The \mtcsettitle command

\mtcsettitle

This command is very similar to the \mtcsettitlefont command. Its syntax is almost identical:

\mtcsettitle{*mini-table*}{*text*}

The *mini-table* type is a keyword like minitoc. The *text* is the text for a mini-table title.

\if@mtc@settitle@

First, we declare a flag, set true:

```
\newif\if@mtc@settitle@\@mtc@settitle@true
```

\mtcsettitle

Then we define the \mtcsettitle command, which has two arguments:

```
\newcommand{\mtcsettitle}[2]{%
```

\mtc@mtati@abbrev
\if@mtc@settitle@
\@nameuse

We process the first argument, which is a keyword of the typetable family. The result is stored in \mtc@mtati@abbrev:

```
\def\mtc@mtati@abbrev{X}
\@mtc@settitle@true
\expandafter\ifx\csname mtc@typetable@#1\endcsname\relax
   \@mtc@settitle@false
   \def\mtc@mtati@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsettitle \space has a wrong first argument (#1).
       \MessageBreak
       It should be a mini-table type (parttoc...sectlot)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}
\else
   \edef\mtc@mtati@abbrev{\@nameuse{mtc@typetable@#1}}
\fi
```

\if@mtc@settitle@
\mtc@tmpti@name
\mtc@mtati@abbrev
\mtc@toks

And we construct the name of the effective macro and apply it:

```
\if@mtc@settitle@
   \def\mtc@tmpti@name{\mtc@mtati@abbrev title}
   \mtc@toks{#2}
   \PackageInfo{minitoc}%
```

```
      {\string\mtcsettitle\space redefines the macro
       \mtc@tmpti@name \space as "\the\mtc@toks"}
   \expandafter\def\csname\mtc@tmpti@name\endcsname{#2}
\else
   \PackageError{minitoc}%
      {Illegal type of table (#1)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}{\relax}
\fi
}
```

5.65.5 The \mtcsetformat command

\@namedef

We define first the keywords (family formatparam) for the three formatting parameters that this command can alter:

```
\@namedef{mtc@formatparam@dotinterval}{dotsep}%
\def\mtc@arg@dotinterval{dotsep}
\@namedef{mtc@formatparam@tocrightmargin}{tocrmarg}%
\def\mtc@arg@tocrightmargin{tocrightmargin}
\@namedef{mtc@formatparam@pagenumwidth}{pnumwidth}%
\def\mtc@arg@pagenumwidth{\mtc@arg@pagenumwidth}
%% \@namedef{mtc@arg@numwidth}{numwidth} %not yet available
%% \def\mtc@arg@numwidth{\mtc@arg@numwidth} %not yet available
```

\AtBeginDocument

The \mtcsetformat command needs an initialization to be done at the beginning of the document, to set the defaults values of the formatting parameters:

```
\AtBeginDocument{%
```

\@pnumwidth
\ptcpnumwidth
\mtcpnumwidth
\stcpnumwidth
\plfpnumwidth
\mlfpnumwidth
\slfpnumwidth
\pltpnumwidth
\mltpnumwidth
\sltpnumwidth

We take, if possible, the default value of \@pnumwith for each type of mini-tables:

```
\@ifundefined{ptcpnumwidth}{\let\ptcpnumwidth\@pnumwidth}{}%
\@ifundefined{stcpnumwidth}{\let\stcpnumwidth\@pnumwidth}{}%
\@ifundefined{mtcpnumwidth}{\let\mtcpnumwidth\@pnumwidth}{}%
\@ifundefined{plfpnumwidth}{\let\plfpnumwidth\@pnumwidth}{}%
\@ifundefined{mlfpnumwidth}{\let\mlfpnumwidth\@pnumwidth}{}%
\@ifundefined{slfpnumwidth}{\let\slfpnumwidth\@pnumwidth}{}%
\@ifundefined{pltpnumwidth}{\let\pltpnumwidth\@pnumwidth}{}%
\@ifundefined{mltpnumwidth}{\let\mltpnumwidth\@pnumwidth}{}%
\@ifundefined{sltpnumwidth}{\let\sltpnumwidth\@pnumwidth}{}%
```

\@tocrmarg \ptctocrmarg \mtclofrmarg \stclotrmarg \plftocrmarg \mlflofrmarg \slflotrmarg \plttocrmarg \mltlofrmarg \sltlotrmarg

We take, if possible, the default value of `\@tocrmarg` for each type of mini-tables:

```
\@ifundefined{ptctocrmarg}{\let\ptctocrmarg\@tocrmarg}{}%
\@ifundefined{mtctocrmarg}{\let\mtctocrmarg\@tocrmarg}{}%
\@ifundefined{stctocrmarg}{\let\stctocrmarg\@tocrmarg}{}%
\@ifundefined{plftocrmarg}{\let\plftocrmarg\@tocrmarg}{}%
\@ifundefined{mlftocrmarg}{\let\mlftocrmarg\@tocrmarg}{}%
\@ifundefined{slftocrmarg}{\let\slftocrmarg\@tocrmarg}{}%
\@ifundefined{plttocrmarg}{\let\plttocrmarg\@tocrmarg}{}%
\@ifundefined{mlttocrmarg}{\let\mlttocrmarg\@tocrmarg}{}%
\@ifundefined{slttocrmarg}{\let\slttocrmarg\@tocrmarg}{}%
```

\@dotsep \ptcdotsep \mtcdotsep \stcdotsep \plfdotsep \mlfdotsep \slfdotsep \pltdotsep \mltdotsep \sltdotsep

We take, if possible, the default value of `\@dotsep` for each type of mini-tables:

```
\@ifundefined{ptcdotsep}{\let\ptcdotsep\@dotsep}{}%
\@ifundefined{mtcdotsep}{\let\mtcdotsep\@dotsep}{}%
\@ifundefined{stcdotsep}{\let\stcdotsep\@dotsep}{}%
\@ifundefined{plfdotsep}{\let\plfdotsep\@dotsep}{}%
\@ifundefined{mlfdotsep}{\let\mlfdotsep\@dotsep}{}%
\@ifundefined{slfdotsep}{\let\slfdotsep\@dotsep}{}%
\@ifundefined{pltdotsep}{\let\pltdotsep\@dotsep}{}%
\@ifundefined{mltdotsep}{\let\mltdotsep\@dotsep}{}%
\@ifundefined{sltdotsep}{\let\sltdotsep\@dotsep}{}%
```

And we terminate the `\AtBeginDocument` block:

```
}
```

\mtcsetformat

The executive part is done via the following macros, which are invoked in the `\mtc@verse`-like environments for each kind of mini-table. These commands activate the values recorded by `\mtcsetformat`.

\ptc@setform ptc@verse

The `\ptc@setform` macro is invoked in `ptc@verse` to set format parameters:

```
\def\ptc@setform{%
\let\@pnumwidth\ptcpnumwidth\relax
\let\@tocrmarg\ptctocrmarg\relax
\let\@dotsep\ptcdotsep\relax
}
```

\mtc@setform mtc@verse

The `\mtc@setform` macro is invoked in `mtc@verse` to set format parameters:

```
\def\mtc@setform{%
\let\@pnumwidth\mtcpnumwidth\relax
\let\@tocrmarg\mtctocrmarg\relax
\let\@dotsep\mtcdotsep\relax
}
```

\stc@setform stc@verse

The \stc@setform macro is invoked in stc@verse to set format parameters:

```
\def\stc@setform{%
\let\@pnumwidth\stcpnumwidth\relax
\let\@tocrmarg\stctocrmarg\relax
\let\@dotsep\stcdotsep\relax
}
```

\plf@setform ptc@verse

The \plf@setform macro is invoked in ptc@verse to set format parameters:

```
\def\plf@setform{%
\let\@pnumwidth\plfpnumwidth\relax
\let\@tocrmarg\plftocrmarg\relax
\let\@dotsep\plfdotsep\relax
}
```

\mlf@setform mtc@verse

The \mlf@setform macro is invoked in mtc@verse to set format parameters:

```
\def\mlf@setform{%
\let\@pnumwidth\mlfpnumwidth\relax
\let\@tocrmarg\mlftocrmarg\relax
\let\@dotsep\mlfdotsep\relax
}
```

\slf@setform stc@verse

The \slf@setform macro is invoked in stc@verse to set format parameters:

```
\def\slf@setform{%
\let\@pnumwidth\slfpnumwidth\relax
\let\@tocrmarg\slftocrmarg\relax
\let\@dotsep\slfdotsep\relax
}
```

\plt@setform ptc@verse

The \plt@setform macro is invoked in ptc@verse to set format parameters:

```
\def\plt@setform{%
\let\@pnumwidth\pltpnumwidth\relax
\let\@tocrmarg\plttocrmarg\relax
\let\@dotsep\pltdotsep\relax
}
```

\mlt@setform mtc@verse

The \mlt@setform macro is invoked in mtc@verse to set format parameters:

```
\def\mlt@setform{%
\let\@pnumwidth\plfpnumwidth\relax
\let\@tocrmarg\plftocrmarg\relax
\let\@dotsep\plfdotsep\relax
}
```

\slt@setform stc@verse

The `\slt@setform` macro is invoked in `stc@verse` to set format parameters:

```
\def\slt@setform{%
\let\@pnumwidth\plfpnumwidth\relax
\let\@tocrmarg\plftocrmarg\relax
\let\@dotsep\plfdotsep\relax
}
```

\if@mtc@setformat@ \mtcsetformat

We now define a flag and the `\mtcsetformat` command, which has the following syntax:

`\mtcsetformat{`*mini-table*`}{`*parameter-name*`}{`*value*`}`

where *mini-table* is a keyword of the `typetable` family, *parameter-name* is a keyword of the `formatparam` family and *value*, the value of this parameter for the given kind of mini-table.

```
\newif\if@mtc@setformat@\@mtc@setformat@true
\newcommand{\mtcsetformat}[3]{%
```

\mtc@mtf@abbrev \mtc@fparam@abbrev

We now process the first argument and store the result in `\mtc@mtf@abbrev`:

```
\def\mtc@mtf@abbrev{X}
\def\mtc@fparam@abbrev{X}
\@mtc@setformat@true
\expandafter\ifx\csname mtc@typetable@#1\endcsname\relax
  \@mtc@setformat@false
  \def\mtc@mtf@abbrev{X}
  \PackageError{minitoc}%
     {*** \string\mtcsetformat \space has a wrong first argument (#1).
     \MessageBreak
     It should be a mini-table type (parttoc...sectlot)}%
     {Correct the source code.\MessageBreak
      Type <return> and rerun LaTeX}
\else
  \edef\mtc@mtf@abbrev{\@nameuse{mtc@typetable@#1}}
\fi
```

\mtc@fparam@abbrev \@nameuse

Then the second argument is processed and the result is stored in `\mtc@fparam@abbrev`:

```
\expandafter\ifx\csname mtc@formatparam@#2\endcsname\relax
 \@mtc@setformat@false
  \def\mtc@fparam@abbrev{X}
  \PackageError{minitoc}%
     {*** \string\mtcsetformat \space has a wrong second argument (#2).
      \MessageBreak
      It should be a formatting param
      (pagenumwidth, tocrightmargin, dotinterval)}%
     {Correct the source code.\MessageBreak
```

```
      Type <return> and rerun LaTeX}
\else
  \edef\mtc@fparam@abbrev{\@nameuse{mtc@formatparam@#2}}
\fi
```

\if@mtc@setformat@
\mtc@tmpfm@name
\mtc@mtf@abbrev
\mtc@fparam@abbrev

The name of the storage macro is build and it receives the third parameter as value:

```
\if@mtc@setformat@
  \def\mtc@tmpfm@name{\mtc@mtf@abbrev\mtc@fparam@abbrev}
  \PackageInfo{minitoc}%
     {\string\mtcsetformat\space redefines
      the macro \mtc@tmpfm@name \space as \string#3}
  \expandafter\def\csname\mtc@tmpfm@name\endcsname{#3}
\else
  \PackageError{minitoc}%
     {\string\mtcsetformat\space has incompatible
      \MessageBreak
      first (#1) and second (#2) arguments}{}
\fi
}
```

5.65.6 The `\mtcsetpagenumbers` command

This command activates of inhibits page numbers in the mini-tables of a given kind. Its syntax is the following:

`\mtcsetpagenumbers{`*mini-table*`}{on|off}`

where *mini-table* is a keyword for a kind of mini-table (`parttoc`, … `sectlot`), or `on` and `off` a keyword to activate (`on`) or inhibit (`off`) the page numbers. `on` and `off` have many synonyms.

\if@mtc@setpagenumbers@
\if@mtc@spn@ok@

We define some flags:

```
\newif\if@mtc@setpagenumbers@ \@mtc@setpagenumbers@false
\newif\if@mtc@spn@ok@
```

\mtcsetpagenumber

Then the `\mtcsetpagenumbers` macro, with two arguments:

```
\newcommand{\mtcsetpagenumbers}[2]{%
```

\mtc@mttpn@abbrev
\mtc@pnsw@abbrev

We process the first argument, a keyword of the `typetable` family, and store the result in `\mtc@mttpn@abbrev`:

```
\def\mtc@mttpn@abbrev{X}
\@mtc@setpagenumbers@true
\def\mtc@pnsw@abbrev{}
\expandafter\ifx\csname mtc@typetable@#1\endcsname\relax
   \@mtc@setpagenumbers@false
   \def\mtc@pnsw@abbrev{X}
   \def\mtc@mttpn@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsetpagenumbers \space has a wrong first argument (#1).}%
      {It should be a mini-table type (parttoc...sectlot)\MessageBreak
       Correct the source code, type <return> and rerun LaTeX}
\else
   \edef\mtc@mttpn@abbrev{\@nameuse{mtc@typetable@#1}}
\fi
```

\if@mtc@spn@ok@
\mtc@pnsw@abbrev
\mtc@mttpn@abbrev
\if@mtc@setpagenumbers@
\mtc@tmppn@name

Then the second argument, a keyword of the YN family, and store the result in \mtc@pnsw@abbrev. The name of the effective macro is build and the macro executed.

```
\@mtc@spn@ok@true
\expandafter\ifx\csname mtc@YN@#2\endcsname\relax
   \@mtc@spn@ok@false
   \def\mtc@pnsw@abbrev{X}
   \def\mtc@mttpn@abbrev{X}
   \@mtc@setpagenumbers@false
   \def\mtc@mttpn@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsetpagenumbers \space has a wrong second argument (#2)}%
      {It should be a boolean value (0/1, yes/no, on/off\MessageBreak
       Correct the source code, type <return> and rerun LaTeX}
\else
   \edef\mtc@pnsw@abbrev{\@nameuse{mtc@YN@#2}}
   \def\mtc@pnsw@abbrevX{X}
   \def\mtc@noX{mtc@noX}
   \def\mtc@tmppn@name{\mtc@pnsw@abbrev\mtc@mttpn@abbrev pagenumbers}
   \expandafter\ifx\csname mtc@\mtc@pnsw@abbrev X\endcsname\mtc@noX
      \PackageInfo{minitoc}%
         {Page numbers are inhibited for the #1s}
   \else
      \PackageInfo{minitoc}%
         {Page numbers are activated for the #1s}
   \fi
   \csname\mtc@tmppn@name\endcsname{}
\fi
}
```

5.65.7 The \mtcsetrules command

This macro is very similar to \mtcsetpagenumbers and its syntax is the same:

\mtcsetrules{*mini-table*}{on|off}

where *mini-table* is a keyword for a kind of mini-table (parttoc, … sectlot), or on and off a keyword to activate (on) or inhibit (off) the horizontal rules. on and off have many synonyms.

Hence the code is similar.

\if@mtc@setrules@
\if@mtc@sru@ok@

We define some flags:

```
\newif\if@mtc@setrules@ \@mtc@setrules@false
\newif\if@mtc@sru@ok@
```

\mtcsetrules

Then the \mtcsetrules macro, which has two arguments:

```
\newcommand{\mtcsetrules}[2]{%
```

\mtc@mttru@abbrev
\if@mtc@setrules@
\mtc@rusw@abbrev
\@nameuse

We process the first argument, a keyword of the typetable family and store the result in \mtc@mttru@abbrev:

```
\def\mtc@mttru@abbrev{X}
\@mtc@setrules@true
\def\mtc@rusw@abbrev{}
\expandafter\ifx\csname mtc@typetable@#1\endcsname\relax
   \@mtc@setrules@false
   \def\mtc@rusw@abbrev{X}
   \def\mtc@mttru@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsetrules \space has a wrong first argument (#1).}%
      {It should be a mini-table type (parttoc...sectlot)
       \MessageBreak
       Correct the source code, type <return> and rerun LaTeX}
\else
   \edef\mtc@mttru@abbrev{\@nameuse{mtc@typetable@#1}}
\fi
```

\if@mtc@sru@ok@
\mtc@rusw@abbrev
\mtc@mttru@abbrev
if@setrules@false
\mtc@noX
\mtc@tmppn@name

Then the second argument, a keyword of the YN family, and store the result in \mtc@rusw@abbrev. The name of the effective macro is build and the macro executed.

```
\@mtc@sru@ok@true
\expandafter\ifx\csname mtc@YN@#2\endcsname\relax
   \@mtc@sru@ok@false
   \def\mtc@rusw@abbrev{X}
   \def\mtc@mttru@abbrev{X}
   \@mtc@setrules@false
   \PackageError{minitoc}%
      {*** \string\mtcsetrules \space has a wrong second argument (#2)}%
      {It should be a boolean value (0/1, yes/no, on/off
```

```
      \MessageBreak
      Correct the source code, type <return> and rerun LaTeX}
\else
   \edef\mtc@rusw@abbrev{\@nameuse{mtc@YN@#2}}
   \def\mtc@rusw@abbrevX{X}
   \def\mtc@noX{mtc@noX}
   \def\mtc@tmppn@name{\mtc@rusw@abbrev\mtc@mttru@abbrev rule}
   \expandafter\ifx\csname mtc@\mtc@rusw@abbrev X\endcsname\mtc@noX
      \PackageInfo{minitoc}%
         {Horizontal rules are inhibited for the #1s}
   \else
      \PackageInfo{minitoc}%
         {Horizontal rules are activated for the #1s}
   \fi
   \csname\mtc@tmppn@name\endcsname{}
\fi
}
```

5.65.8 The \mtcsetfeature command

For this command, we must define three families of keywords, but the third is just used to add the word “style” for the “pagestyle” when “thispage” is used.

A family for the long names of the types of mini-tables:

```
\@namedef{mtc@ltypetable@parttoc}{parttoc}\def\mtc@ltypetable@parttoc{parttoc}
\@namedef{mtc@ltypetable@partlot}{partlot}\def\mtc@ltypetable@partlot{partlot}
\@namedef{mtc@ltypetable@partlof}{partlof}\def\mtc@ltypetable@partlof{partlof}
\@namedef{mtc@ltypetable@minitoc}{minitoc}\def\mtc@ltypetable@minitoc{minitoc}
\@namedef{mtc@ltypetable@minilot}{minilot}\def\mtc@ltypetable@minilot{minilot}
\@namedef{mtc@ltypetable@minilof}{minilof}\def\mtc@ltypetable@minilof{minilof}
\@namedef{mtc@ltypetable@secttoc}{secttoc}\def\mtc@ltypetable@secttoc{secttoc}
\@namedef{mtc@ltypetable@sectlof}{sectlof}\def\mtc@ltypetable@sectlof{sectlof}
\@namedef{mtc@ltypetable@sectlot}{sectlot}\def\mtc@ltypetable@sectlot{sectlot}
```

A family for the type of feature:

```
\@namedef{mtc@featureparam@before}{before}%
   \def\mtc@featureparam@before{before}
\@namedef{mtc@featureparam@after}{after}%
   \def\mtc@featureparam@after{after}
\@namedef{mtc@featureparam@pagestyle}{thispage}%
   \def\mtc@featureparam@pagestyle{thispage}
```

And a family to add “style” if it is a “pagestyle” feature:

```
\@namedef{mtc@ft3@before}{}\expandafter\def\csname mtc@ft3@before\endcsname{}
\@namedef{mtc@ft3@after}{}\expandafter\def\csname mtc@ft3@after\endcsname{}
\@namedef{mtc@ft3@pagestyle}{style}%
```

```
    \expandafter\def\csname mtc@ft3@pagestyle\endcsname{style}
```

The `\mtcsetfeature` command has the following syntax:

`\mtcsetfeature`{*mini-table*}{*feature-name*}{*commands*}

where *mini-table* is a keyword of the `ltypetable` family, *feature-name* is a keyword of the `featureparam` family (but also of the `ft3` family), and *commands* are the commands which constitute the selected feature.

`\if@mtc@setfeature@`
`\mtcsetfeature`

We define a flag and the `\mtcsetfeature` command, with three arguments:

```
\newif\if@mtc@setfeature@\@mtc@setfeature@true
\newcommand{\mtcsetfeature}[3]{%
```

`\mtc@mtfeat@abbrev`
`\mtc@featparam@abbrev`

We process the first argument, a keyword of the `ltypetable` family and store the result in `\mtc@mtfeat@abbrev`:

```
\def\mtc@mtfeat@abbrev{X}
\def\mtc@featparam@abbrev{X}
\@mtc@setfeature@true
\expandafter\ifx\csname mtc@ltypetable@#1\endcsname\relax
  \@mtc@setfeature@false
  \def\mtc@mtfeat@abbrev{X}
  \PackageError{minitoc}%
     {*** \string\mtcsetfeature \space has a wrong first argument (#1).
      \MessageBreak
      It should be a mini-table type (parttoc...sectlot)}%
     {Correct the source code.\MessageBreak
      Type <return> and rerun LaTeX}
\else
  \edef\mtc@mtfeat@abbrev{\@nameuse{mtc@ltypetable@#1}}
\fi
```

`\if@mtc@setfeature@`
`\mtc@featparam@abbrev`
`\mtc@featparam@third`
`\@nameuse`

The second argument is a keyword of the `featureparam` family, the result is stored in `\mtc@featparam@`; and the complement is computed from the first argument, interpreted as a keyword of the `ft3` family and whose result is stored in `\mtc@featparam@third`.

```
\expandafter\ifx\csname mtc@featureparam@#2\endcsname\relax
 \@mtc@setfeature@false
  \def\mtc@featparam@abbrev{X}
  \def\mtc@featparam@third{X}
  \PackageError{minitoc}%
     {*** \string\mtcsetfeature \space has a wrong second argument (#2).
      \MessageBreak
      It should be a feature param (before, after, pagestyle)}%
     {Correct the source code.\MessageBreak
      Type <return> and rerun LaTeX}
```

```
\else
  \edef\mtc@featparam@abbrev{\@nameuse{mtc@featureparam@#2}}
  \edef\mtc@featparam@third{\@nameuse{mtc@ft3@#2}}
\fi
```

\if@mtc@setfeature@
\mtc@tmpfeat@name
\mtc@featparam@abbrev
\mtc@mtfeat@abbrev
\mtc@featparam@third

The name of the effective macro is built by concatenating these three pieces (named `\mtc@featparam@abbrev`, `\mtc@mtfeat@abbrev`, and `\mtc@featparam@third` respectively), then this macro is executed:

```
\if@mtc@setfeature@
  \def\mtc@tmpfeat@name%
     {\mtc@featparam@abbrev\mtc@mtfeat@abbrev\mtc@featparam@third}
  \PackageInfo{minitoc}%
     {\string\mtcsetfeature\space redefines the macro
      "\csname mtc@tmpfeat@name\endcsname" as "\string#3"}
  \expandafter\def\csname\mtc@tmpfeat@name\endcsname{#3}
\else
  \PackageError{minitoc}%
     {\string\mtcsetfeature\space has incompatible\MessageBreak
      first (#1) and second (#2) arguments}{}
\fi}
```

5.65.9 The `\mtcsetdepth` command

This command is very similar to the `\mtcsettitle` command. Its syntax is almost identical:

`\mtcsetdepth`{*mini-table*}{*depth*}

The *mini-table* type is a keyword like `minitoc`. The *depth* is the depth for a mini-table. If it is a mini-table for a list of figures or tables, the corresponding depth counter *must be available*, *i.e.* must have been created (often by an adequate package, like the `subfig` package [13]).

\if@mtc@setdepth@

First, we declare a flag, set true:

```
\newif\if@mtc@setdepth@\@mtc@setdepth@true
```

\mtcsetdepth

Then we define the `\mtcsetdepth` command, with two arguments:

```
\newcommand{\mtcsetdepth}[2]{%
```

\mtc@mtade@abbrev
\if@mtc@setdepth@
\@nameuse

We process the first argument, which is a keyword of the `ltypetable` family. The result is stored in `\mtc@mtade@abbrev`:

```
\def\mtc@mtade@abbrev{X}
\@mtc@setdepth@true
\expandafter\ifx\csname mtc@ltypetable@#1\endcsname\relax
   \@mtc@setdepth@false
   \def\mtc@mtade@abbrev{X}
   \PackageError{minitoc}%
      {*** \string\mtcsetdepth \space has a wrong first argument (#1).
       \MessageBreak
       It should be a mini-table type (parttoc...sectlot)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}
\else
   \edef\mtc@mtade@abbrev{\@nameuse{mtc@ltypetable@#1}}
\fi
```

\if@mtc@setdepth@
\mtc@tmpde@name
\mtc@mtade@abbrev
\mtc@toks
\setcounter

And we construct the name of the effective counter and gave it the value:

```
\if@mtc@setdepth@
   \def\mtc@tmpde@name{\mtc@mtade@abbrev depth}
   \expandafter\noexpand\@ifundefined{c@\mtc@mtade@abbrev depth}%
      {\PackageError{minitoc}%
       {***  \string\mtcsetdepth \space attempt to use\MessageBreak
             an undefined counter (#1depth).}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}}{%
   \mtc@toks{#2} % trick for explicit message using \the.
   \PackageInfo{minitoc}%
      {\string\mtcsetdepth\space redefines the counter
       \mtc@tmpde@name \space as "\the\mtc@toks"}%
    \expandafter\setcounter{\mtc@tmpde@name}{#2}}%
\else
   \PackageError{minitoc}%
      {\string\mtcsetdepth:\space Illegal type of table (#1)}%
      {Correct the source code.\MessageBreak
       Type <return> and rerun LaTeX}{\relax}
\fi}% end of \mtcsettitle
```

5.66 The mtchideinmaintoc environment

\if@mtc@Himtoc@

The flag \if@mtc@Himtoc@ is used to detect an incorrect imbrication of this environment:

```
\newif\if@mtc@Himtoc@ \@mtc@Himtoc@false
```

mtchideinmaintoc \if@mtc@Himtoc@ \mtc@sv@tocdepth \arabic \addtocontents \setcounter

Then we define this environment, which inserts into the TOC file commands of the form `\setcounter{tocdepth}{...}`. It also save and restore the value of the counter `tocdepth`, as the optional argument is the hiding depth of the entries in the main TOC.

```
\newenvironment{mtchideinmaintoc}[1][-1]%
{\if@mtc@Himtoc@\PackageError{minitoc}%
    {Imbrication of mtchideinmaintoc environments}%
    {The hiding in main ToC could be incorrect}\fi
 \global\@mtc@Himtoc@true
 \def\mtc@sv@tocdepth{\arabic{tocdepth}}%
 \addtocontents{toc}{\protect\setcounter{tocdepth}{#1}}}%
{\if@mtc@Himtoc@\else\PackageError{minitoc}%
    {Unbalanced mtchideinmaintoc environment}%
    {The hiding in main ToC could be incorrect}\fi
 \global\@mtc@Himtoc@false
 \addtocontents{toc}{\protect\setcounter{tocdepth}{\mtc@sv@tocdepth}}}
```

\AtBeginDocument \if@mtc@Himlof@ mtchideinmainlof \mtc@sv@lofdepth \mtc@sv@tocdepth \mtc@svf@tocdepth \arabic \addtocontents \setcounter

The `mtchideinmainlof` and `mtchideinmainlot` environments are similar, but we must verify the presence of the associated depth counter, so we have two versions of each of these environments. This must be done *after* the loading of the packages.

First, for the list of figures:

```
\newif\if@mtc@Himlof@ \@mtc@Himlof@false
\AtBeginDocument{%
\@ifundefined{c@lofdepth}{%
   \newenvironment{mtchideinmainlof}[1][-1]%
   {\if@mtc@Himlof@\PackageError{minitoc}%
       {Imbrication of mtchideinmainlof environments}%
       {The hiding in main LoF could be incorrect}\fi
    \global\@mtc@Himlof@true
    \def\mtc@sv@tocdepth{\arabic{tocdepth}}%
    \def\mtc@sv@lofdepth{\arabic{tocdepth}}%
    \addtocontents{lof}{\protect\setcounter{tocdepth}{#1}}}%
   {\if@mtc@Himlof@\else\PackageError{minitoc}%
       {Unbalanced mtchideinmainlof environment}%
       {The hiding in main LoF could be incorrect}\fi
    \global\@mtc@Himtoc@false
    \addtocontents{lof}{\protect\setcounter{tocdepth}{\mtc@sv@lofdepth}}}
}%
{%
   \newenvironment{mtchideinmainlof}[1][-1]%
   {\if@mtc@Himlof@\PackageError{minitoc}%
       {Imbrication of mtchideinmainlof environments}%
       {The hiding in main LoF could be incorrect}\fi
    \global\@mtc@Himlof@true
    \def\mtc@svf@tocdepth{\arabic{tocdepth}}%
    \addtocontents{lof}{\protect\setcounter{tocdepth}{#1}}}%
   {\if@mtc@Himlof@\else\PackageError{minitoc}%
       {Unbalanced mtchideinmainlof environment}%
       {The hiding in main LoF could be incorrect}\fi
    \global\@mtc@Himlof@false
```

```
    \addtocontents{lof}{\protect\setcounter{tocdepth}{\mtc@svf@tocdepth}}}}
```

\AtBeginDocument \if@mtc@Himlot@ mtchideinmainlot \mtc@sv@lotdepth \mtc@sv@tocdepth \mtc@svt@tocdepth \arabic \addtocontents \setcounter

Then for the list of tables:

```
\newif\if@mtc@Himlot@ \@mtc@Himlot@false
\AtBeginDocument{%
\@ifundefined{c@lotdepth}{%
   \newenvironment{mtchideinmainlot}[1][-1]%
   {\if@mtc@Himlot@\PackageError{minitoc}%
       {Imbrication of mtchideinmainlot environments}%
       {The hiding in main LoT could be incorrect}\fi
    \global\@mtc@Himlot@true
    \def\mtc@sv@tocdepth{\arabic{tocdepth}}%
    \def\mtc@sv@lotdepth{\arabic{tocdepth}}%
    \addtocontents{lot}{\protect\setcounter{tocdepth}{#1}}}%
   {\if@mtc@Himlot@\else\PackageError{minitoc}%
       {Unbalanced mtchideinmainlot environment}%
       {The hiding in main LoT could be incorrect}\fi
    \global\@mtc@Himlot@false
    \addtocontents{lot}{\protect\setcounter{tocdepth}{\mtc@sv@lotdepth}}}
}%
{%
   \newenvironment{mtchideinmainlot}[1][-1]%
   {\if@mtc@Himlot@\PackageError{minitoc}%
       {Imbrication of mtchideinmainlot environments}%
       {The hiding in main LoT could be incorrect}\fi
    \global\@mtc@Himlot@true
    \def\mtc@svt@tocdepth{\arabic{tocdepth}}%
    \addtocontents{lot}{\protect\setcounter{tocdepth}{#1}}}%
   {\if@mtc@Himlot@\else\PackageError{minitoc}%
       {Unbalanced mtchideinmainlot environment}%
       {The hiding in main LoT could be incorrect}\fi
    \global\@mtc@Himlot@false
    \addtocontents{lot}{\protect\setcounter{tocdepth}{\mtc@svt@tocdepth}}}}
```

5.67 Fixing the “Index” entry in the TOC

This macro is complex. Its syntax is:

```
\mtcfixindex[part|chapter|section]
```

\@ifundefined \mtc@ixfix@level

Depending of the document class, the “Index” entry in the TOC is treated as a starred chapter or a starred section. Hence we must first determine the default value of the optional argument. The default value is then stored in the macro `\mtc@ixfix@level`. This is done by the following code, which eventually gives a warning message:

```
\@ifundefined{chapter}{%
   \@ifundefined{section}%
      {\PackageWarningNoLine{minitoc}%
       {\string\chapter\space and \string\section\space are undefined.%
        \MessageBreak Cannot use \string\mtcfixindex\space %
           without optional argument [part]}%
       \@ifundefined{part}%
         {\PackageError{minitoc}%
            {But \string\part\space is undefined}%
            {\string\mtcfixindex\space not usable}}%
       {\PackageWarningNoLine{minitoc}%
          {\string\mtcfixindex\space can only be used with the [part]
           optional argument, which becomes the default}%
        \def\mtc@ixfix@level{part}%
       }%
       {\def\mtc@ixfix@level{section}}}}%
{\def\mtc@ixfix@level{chapter}}
```

`\if@mtcfixindex@` `\mtcfixindex` `\addcontentsline`

Then we define a flag and the command `\mtcfixindex`, which adds the necessary lines in the TOC, the LOF and the LOT.

```
\newif\if@mtcfixindex@ \@mtcfixindex@false
\newcommand{\mtcfixindex}[1][\mtc@ixfix@level]{%
   \@mtcfixindex@false
   \expandafter%
    \ifx\csname #1\endcsname\part\relax\@mtcfixindex@true\fi
   \expandafter%
    \ifx\csname #1\endcsname\chapter\relax\@mtcfixindex@true\fi
   \expandafter%
    \ifx\csname #1\endcsname\section\relax\@mtcfixindex@true\fi
   \if@mtcfixindex@
   \addcontentsline{lof}{x\mtc@ixfix@level}{}%
   \addcontentsline{lot}{x\mtc@ixfix@level}{}%
   \csname mtcadd\mtc@ixfix@level\endcsname\relax
   \else
   \PackageError{minitoc}%
      {The optional argument of \string\mtcfixindex\space is wrong}%
      {It must be omitted (\mtc@ixfix@level),
       or be part, chapter or section}%
   \fi
}%
```

5.68 Fixing the “Glossary” entry in the TOC

This macro is complex. Its syntax is:

```
\mtcfixglossary[part|chapter|section]
```

\@ifundefined
\mtc@glofix@level

Depending of the document class, the "Glossary" entry in the TOC is treated as a starred chapter or a starred section. Hence we must first determine the default value of the optional argument. The default value is then stored in the macro \mtc@glofix@level. This is done by the following code, which eventually gives a warning message:

```
\@ifundefined{chapter}{%
   \@ifundefined{section}%
      {\PackageWarningNoLine{minitoc}%
       {\string\chapter\space and \string\section\space are undefined.%
        \MessageBreak Cannot use \string\mtcfixglossary\space %
           without optional argument [part]}%
       \@ifundefined{part}%
         {\PackageError{minitoc}%
            {But \string\part\space is undefined}%
            {\string\mtcfixglossary\space not usable}}%
       {\PackageWarningNoLine{minitoc}%
          {\string\mtcfixglossary\space can only be used with the [part]
           optional argument, which becomes the default}%
        \def\mtc@glofix@level{part}%
       }%
       {\def\mtc@glofix@level{section}}}}%
{\def\mtc@glofix@level{chapter}}
```

\if@mtcfixglossary@
\mtcfixglossary
\addcontentsline

Then we define a flag (\if@mtcfixglossary@) and the command \mtcfixglossary, which adds the necessary lines in the TOC, the LOF and the LOT.

```
\newif\if@mtcfixglossary@ \@mtcfixglossary@false
\newcommand{\mtcfixglossary}[1][\mtc@glofix@level]{%
   \@mtcfixglossary@false
   \expandafter%
    \ifx\csname #1\endcsname\part\relax\@mtcfixglossary@true\fi
   \expandafter%
    \ifx\csname #1\endcsname\chapter\relax\@mtcfixglossary@true\fi
   \expandafter%
    \ifx\csname #1\endcsname\section\relax\@mtcfixglossary@true\fi
   \if@mtcfixglossary@
   \addcontentsline{lof}{x\mtc@glofix@level}{}%
   \addcontentsline{lot}{x\mtc@glofix@level}{}%
   \csname mtcadd\mtc@glofix@level\endcsname\relax
   \else
   \PackageError{minitoc}%
      {The optional argument of \string\mtcfixglossary\space is wrong}%
      {It must be omitted (\mtc@glofix@level),
       or be part, chapter or section}%
   \fi
}%
```

5.69 The \mtcselectlanguage command

\mtcselectlanguage
\if@mtc@insellang@
\IfFileExists
\@input

This command loads a minitoc language definition file *language*`.mld` to set the language-dependent titles for the mini-tables. But first, we verify that this file exists. The flag `\if@mtc@insellang@` is true while we are in this macro.

```
\newif\if@mtc@insellang@ \@mtc@insellang@false
\def\mtcselectlanguage#1{%
     \InputIfFileExists{#1.mld}%
       {\PackageInfo{minitoc}{#1 language selected.%
         \MessageBreak}}%
       {\PackageError{minitoc}%
          {#1 is not a known language, \MessageBreak
           #1.mld not found. \MessageBreak
           Command ignored}%
          {See the minitoc documentation. \MessageBreak
           Correct the source using a valid language name. \MessageBreak
           Press RETURN}}%
}
```

5.70 The \mtcloadmlo internal command

\mtcloadmlo
\if@mtc@insellang@
\IfFileExists
\@input

This command loads a minitoc language object file *language*`.mlo` to set the language-dependent titles for the mini-tables when exotic characters are needed. This command is used only in some `.mld` files when the title strings can not be generated by the normal processing of `minitoc.dtx`. The `.mlo` files are generated by `filecontents` environments in the `minitoc.ins` file. But first, we verify that this `.mlo` file exists.

This command should not be directly invoked by the user. This is verified via the flag `\if@mtc@insellang@`.

```
\def\mtcloadmlo#1{%
     \if@mtc@insellang@
     \InputIfFileExists{#1.mlo}%
       {\PackageInfo{minitoc}{#1 minitoc language object selected.%
         \MessageBreak}}%
       {\PackageError{minitoc}%
          {#1 is not a known minitoc language object file (.mlo), \MessageBreak
           #1.mlo not found. \MessageBreak
           Command ignored}%
          {See the minitoc documentation. \MessageBreak
           Correct the source using a valid language name. \MessageBreak
           Press RETURN}}%
     \else
       \PackageError{minitoc}%
           {You are using the \string\mtcloadmlo\space command\MessageBreak
            outside of a .mld file}%
           {It will be ignored}
```

```
        \@mtc@insellang@false
      \fi
}
```

5.71 The "coffee breaks"

\addcoffeeline
\addtocontents
\coffeeline
\l@coffee
\@Undottedtocline

For the `minutes` package [30] (by Knut LICKERT), we need some commands to insert special entries, undotted, in the TOC to mark "coffee breaks" in a conference. Hence we define `\addcoffeeline`, `\coffeeline` and `\l@coffee`, and internal commands analog to the standard internal commands to format the TOC.

```
\def\addcoffeeline#1#2#3{%
  \addtocontents{#1}{\protect\coffeeline{#2}{#3}{\null}}}
\def\coffeeline#1{\csname l@#1\endcsname}
\newcommand*\l@coffee{\@Undottedtocline{1}{1.5em}{2.3em}}
```

5.72 Initialization of counters

\AtBeginDocument
\@ifundefined
\setcounter

At the beginning of the document, we initialize the absolute counters for parts, chapters and sections, if they are defined.

```
\AtBeginDocument{%
\@ifundefined{c@ptc}{}{\setcounter{ptc}{0}}
\@ifundefined{c@mtc}{}{\setcounter{mtc}{0}}
\@ifundefined{c@stc}{}{\setcounter{stc}{0}}}
```

5.73 Declarations for simple options

These options are just setting a flag.

5.73.1 Options `tight` and `loose`, `k-tight` and `k-loose`

\DeclareOption
\iftightmtc
\kiftightmtc

These options influence the interline separation in the mini-tables.

```
\DeclareOption{tight}{\tightmtctrue}
\DeclareOption{loose}{\tightmtcfalse} % default
\DeclareOption{k-tight}{\ktightmtctrue}
\DeclareOption{k-loose}{\ktightmtcfalse} % default
```

5.73.2 Options checkfiles and nocheckfiles

\DeclareOption \if@mtc@checkfiles

These options activate or inhibit the checking for empty mini-table files.

```
\DeclareOption{checkfiles}{\@mtc@checkfilestrue} % default
\DeclareOption{nocheckfiles}{\@mtc@checkfilesfalse}
```

5.73.3 Options dotted and undotted

\DeclareOption \ifundottedmtc

These options activate or inhibit the leaders (lines of dots) in the mini-tables.

```
\DeclareOption{undotted}{\undottedmtctrue}
\DeclareOption{dotted}{\undottedmtcfalse} % default
```

5.73.4 Option notoccite

This option will later load the notoccite package [2].

\DeclareOption \if@mtc@notoccite@

```
\DeclareOption{notoccite}{\@mtc@notoccite@true}
```

5.73.5 Option shortext

\DeclareOption \if@longextensions@

This option forces the use of short extensions.

```
\DeclareOption{shortext}{\@longextensions@false
\PackageWarningNoLine{minitoc}%
   {*** You have forced the use of short extensions ***}}
```

5.74 The insection option

\if@mtc@ss@insection@ \@ifundefined \DeclareOption

This option is available only if \chapter is not defined and \section defined. It is to be revised when chapter/section level commands will ever be allowed together, sometime in the far away future.

```
\newif\if@mtc@ss@insection@ \@mtc@ss@insection@false
\@ifundefined{chapter}{%
   \@ifundefined{section}{}%
```

```
                              {%
                               \DeclareOption{insection}%
                                {\@mtc@ss@insection@true}%
                              }%
}{}
```

5.75 Language options

\DeclareOption \mtcselectlanguage

Each language option reads the corresponding *language*.mld file via the specialized macro \mtcselectlanguage. If the file does not exists, a standard error message is displayed. The language options are (should be) in alphabetical order. Several options could load the same file, but, by convention, there should be a *language*.mld file for each language option, but this file can load another one (as francais.mld loads french.mld).

```
\DeclareOption{acadian}{\mtcselectlanguage{acadian}}%
\DeclareOption{acadien}{\mtcselectlanguage{acadien}}%
\DeclareOption{afrikaan}{\mtcselectlanguage{afrikaan}}%
\DeclareOption{afrikaans}{\mtcselectlanguage{afrikaans}}%
\DeclareOption{american}{\mtcselectlanguage{american}}%
\DeclareOption{arab}{\mtcselectlanguage{arab}}%
\DeclareOption{arabic}{\mtcselectlanguage{arabic}}%
\DeclareOption{armenian}{\mtcselectlanguage{armenian}}%
\DeclareOption{austrian}{\mtcselectlanguage{austrian}}%
\DeclareOption{bahasa}{\mtcselectlanguage{bahasa}}%
\DeclareOption{bangla}{\mtcselectlanguage{bangla}}%
\DeclareOption{basque}{\mtcselectlanguage{basque}}%
\DeclareOption{bicig}{\mtcselectlanguage{bicig}}%
\DeclareOption{brazil}{\mtcselectlanguage{brazil}}%
\DeclareOption{brazilian}{\mtcselectlanguage{brazilian}}%
\DeclareOption{breton}{\mtcselectlanguage{breton}}%
\DeclareOption{british}{\mtcselectlanguage{british}}%
\DeclareOption{bulgarian}{\mtcselectlanguage{bulgarian}}%
\DeclareOption{bulgarianb}{\mtcselectlanguage{bulgarianb}}%
\DeclareOption{buryat}{\mtcselectlanguage{buryat}}%
\DeclareOption{canadian}{\mtcselectlanguage{canadian}}%
\DeclareOption{canadien}{\mtcselectlanguage{canadien}}%
\DeclareOption{castillan}{\mtcselectlanguage{castillan}}%
\DeclareOption{castillian}{\mtcselectlanguage{castillian}}%
\DeclareOption{catalan}{\mtcselectlanguage{catalan}}%
\DeclareOption{chinese1}{\mtcselectlanguage{chinese1}}%
\DeclareOption{chinese2}{\mtcselectlanguage{chinese2}}%
\DeclareOption{croatian}{\mtcselectlanguage{croatian}}%
\DeclareOption{czech}{\mtcselectlanguage{czech}}%
\DeclareOption{danish}{\mtcselectlanguage{danish}}%
\DeclareOption{dutch}{\mtcselectlanguage{dutch}}%
\DeclareOption{english}{\mtcselectlanguage{english}}%
\DeclareOption{esperant}{\mtcselectlanguage{esperant}}%
\DeclareOption{esperanto}{\mtcselectlanguage{esperanto}}%
```

```
\DeclareOption{estonian}{\mtcselectlanguage{estonian}}%
\DeclareOption{ethiopia}{\mtcselectlanguage{ethiopia}}%
\DeclareOption{ethiopian}{\mtcselectlanguage{ethiopian}}%
\DeclareOption{farsi1}{\mtcselectlanguage{farsi1}}%
\DeclareOption{farsi2}{\mtcselectlanguage{farsi2}}%
\DeclareOption{finnish}{\mtcselectlanguage{finnish}}%
\DeclareOption{finnish2}{\mtcselectlanguage{finnish2}}%
\DeclareOption{francais}{\mtcselectlanguage{francais}}%
\DeclareOption{french}{\mtcselectlanguage{french}}%
\DeclareOption{frenchb}{\mtcselectlanguage{frenchb}}%
\DeclareOption{frenchle}{\mtcselectlanguage{frenchle}}%
\DeclareOption{frenchpro}{\mtcselectlanguage{frenchpro}}%
\DeclareOption{galician}{\mtcselectlanguage{galician}}%
\DeclareOption{german}{\mtcselectlanguage{german}}%
\DeclareOption{germanb}{\mtcselectlanguage{germanb}}%
\DeclareOption{greek}{\mtcselectlanguage{greek}}%
\DeclareOption{greek-mono}{\mtcselectlanguage{greek-mono}}%
\DeclareOption{greek-polydemo}{\mtcselectlanguage{greek-polydemo}}%
\DeclareOption{greek-polykatha}{\mtcselectlanguage{greek-polykatha}}%
\DeclareOption{guarani}{\mtcselectlanguage{guarani}}%
\DeclareOption{hangul1}{\mtcselectlanguage{hangul1}}%
\DeclareOption{hangul2}{\mtcselectlanguage{hangul2}}%
\DeclareOption{hangul3}{\mtcselectlanguage{hangul3}}%
\DeclareOption{hangul4}{\mtcselectlanguage{hangul4}}%
\DeclareOption{hanja1}{\mtcselectlanguage{hanja1}}%
\DeclareOption{hanja2}{\mtcselectlanguage{hanja2}}%
\DeclareOption{hebrew}{\mtcselectlanguage{hebrew}}%
\DeclareOption{hungarian}{\mtcselectlanguage{hungarian}}%
\DeclareOption{icelandic}{\mtcselectlanguage{icelandic}}%
\DeclareOption{interlingua}{\mtcselectlanguage{interlingua}}%
\DeclareOption{irish}{\mtcselectlanguage{irish}}%
\DeclareOption{italian}{\mtcselectlanguage{italian}}%
\DeclareOption{japanese}{\mtcselectlanguage{japanese}}%
\DeclareOption{japanese2}{\mtcselectlanguage{japanese2}}%
\DeclareOption{japanese3}{\mtcselectlanguage{japanese3}}%
\DeclareOption{japanese4}{\mtcselectlanguage{japanese4}}%
\DeclareOption{japanese5}{\mtcselectlanguage{japanese5}}%
\DeclareOption{latin}{\mtcselectlanguage{latin}}%
\DeclareOption{latin2}{\mtcselectlanguage{latin2}}%
\DeclareOption{latvian}{\mtcselectlanguage{latvian}}%
\DeclareOption{letton}{\mtcselectlanguage{letton}}%
\DeclareOption{lithuanian}{\mtcselectlanguage{lithuanian}}%
\DeclareOption{lsorbian}{\mtcselectlanguage{lsorbian}}%
\DeclareOption{magyar}{\mtcselectlanguage{magyar}}%
\DeclareOption{magyar2}{\mtcselectlanguage{magyar2}}%
\DeclareOption{malayalam-keli}{\mtcselectlanguage{malayalam-keli}}%
\DeclareOption{malayalam-rachana}{\mtcselectlanguage{malayalam-rachana}}%
\DeclareOption{malayalam-rachana2}{\mtcselectlanguage{malayalam-rachana2}}%
\DeclareOption{mongol}{\mtcselectlanguage{mongol}}%
\DeclareOption{naustrian}{\mtcselectlanguage{naustrian}}%
\DeclareOption{ngerman}{\mtcselectlanguage{ngerman}}%
\DeclareOption{ngermanb}{\mtcselectlanguage{ngermanb}}%
\DeclareOption{norsk}{\mtcselectlanguage{norsk}}%
\DeclareOption{nynorsk}{\mtcselectlanguage{nynorsk}}%
```

```
\DeclareOption{polish}{\mtcselectlanguage{polish}}%
\DeclareOption{polish2}{\mtcselectlanguage{polish2}}%
\DeclareOption{portuges}{\mtcselectlanguage{portuges}}%
\DeclareOption{portuguese}{\mtcselectlanguage{portuguese}}%
\DeclareOption{romanian}{\mtcselectlanguage{romanian}}%
\DeclareOption{russian}{\mtcselectlanguage{russian}}%
\DeclareOption{russianb}{\mtcselectlanguage{russianb}}%
\DeclareOption{russianc}{\mtcselectlanguage{russianc}}%
\DeclareOption{russian2m}{\mtcselectlanguage{russian2m}}%
\DeclareOption{russian2o}{\mtcselectlanguage{russian2o}}%
\DeclareOption{samin}{\mtcselectlanguage{samin}}%
\DeclareOption{scottish}{\mtcselectlanguage{scottish}}%
\DeclareOption{serbian}{\mtcselectlanguage{serbian}}%
\DeclareOption{serbianc}{\mtcselectlanguage{serbianc}}%
\DeclareOption{slovak}{\mtcselectlanguage{slovak}}%
\DeclareOption{slovene}{\mtcselectlanguage{slovene}}%
\DeclareOption{spanish}{\mtcselectlanguage{spanish}}%
\DeclareOption{spanish2}{\mtcselectlanguage{spanish2}}%
\DeclareOption{spanish3}{\mtcselectlanguage{spanish3}}%
\DeclareOption{swedish}{\mtcselectlanguage{swedish}}%
\DeclareOption{thai}{\mtcselectlanguage{thai}}%
\DeclareOption{turkish}{\mtcselectlanguage{turkish}}%
\DeclareOption{UKenglish}{\mtcselectlanguage{UKenglish}}%
\DeclareOption{ukraineb}{\mtcselectlanguage{ukraineb}}%
\DeclareOption{USenglish}{\mtcselectlanguage{USenglish}}%
\DeclareOption{usorbian}{\mtcselectlanguage{usorbian}}%
\DeclareOption{vietnam}{\mtcselectlanguage{vietnam}}%
\DeclareOption{vietnamese}{\mtcselectlanguage{vietnamese}}%
\DeclareOption{welsh}{\mtcselectlanguage{welsh}}%
```

5.76 The hints option

\if@mtc@hints@
\if@mtc@hints@w@
\if@mtc@hints@giben@

First, we define some flags:

- The flag \if@mtc@hints@ is true if the hints option is required (default).
- The flag \if@mtc@hints@w@ is set true if we detect that some sectionning commands have been altered since the loading of the document class.
- The flag \if@mtc@hints@given@ is set true if the hints option detects something curious and writes messages in the .log file.

```
\newif\if@mtc@hints@ \@mtc@hints@true
\newif\if@mtc@hints@w@ \@mtc@hints@w@false
\newif\if@mtc@hints@given@ \@mtc@hints@given@false
```

\DeclareOption
\if@mtc@hints@

We declare the hints (default) and nohints options:

```
\DeclareOption{hints}{\@mtc@hints@true}
\DeclareOption{nohints}{\@mtc@hints@false}
```

\mtc@hints@begindoc \AtBeginDocument

The hints option is made of three parts: the first, \mtc@hints@begindoc, is executed via \AtBeginDocument and looks if some packages or classes are loaded and gives warnings about their compatibility with minitoc.

The second part is made of tiny pieces of code inserted in the minitoc code, to verify that some macros are called in the right order.

\mtc@hints@enddoc \AtEndDocument

The third and last part, \mtc@hints@enddoc, is executed via \AtEndDocument and examines some flags set by the first and the second parts. Then, if necessary, it writes some infos in the .log file and/or warnings on the screen and in the .log file. Note that the hints option does not signal errors, only infos and warnings, so it does not stop the LaTeX run.

5.76.1 First part: \mtc@hints@begindoc

\mtc@hints@begindoc

We declare the first part of the hints option (for \AtBeginDocument):

```
\def\mtc@hints@begindoc{%
\PackageInfo{minitoc(hints)}%
   {***** You requested the hints option *****\MessageBreak
    ***** Some hints are eventually given below *****\@gobble}%
```

5.76.1.1 Hint about the appendix package

\@ifpackageloaded \if@mtc@hints@given@

We test the presence of the appendix package [46]:

```
\@ifpackageloaded{appendix}{%
\@mtc@hints@given@true
\PackageInfo{minitoc(hints)}%
   {--- The appendix package is loaded. \MessageBreak
    See the minitoc package documentation\MessageBreak
    for specific precautions\@gobble}}{}%
```

5.76.1.2 Hint about the tocbibind package

\@ifpackageloaded \if@mtc@hints@given@

We test the presence of the tocbibind package [47]:

```
\@ifpackageloaded{tocbibind}%
   {\@mtc@hints@given@true
   \PackageInfo{minitoc(hints)}%
      {--- The tocbibind package is loaded. \MessageBreak
       See the minitoc package documentation
       for specific precautions\@gobble}}{}%
```

5.76.1.3 Hint about the tocloft package

\@ifpackageloaded
\if@mtc@hints@given@

We test the presence of the tocloft package [44]:

```
\@ifpackageloaded{tocloft}%
   {\@mtc@hints@given@true
    \PackageInfo{minitoc(hints)}%
       {--- The tocloft package is loaded. \MessageBreak
        See the minitoc package documentation
        for specific precautions\@gobble}}{}%
```

5.76.1.4 Hint about the titletoc package

\@ifpackageloaded
\if@mtc@hints@given@

We test the presence of the titletoc package [8], and emit a warning, because this package is incompatible with minitoc:

```
\@ifpackageloaded{titletoc}%
   {\@mtc@hints@given@true
    \PackageWarningNoLine{minitoc(hints)}%
       {--- The titletoc package is loaded. \MessageBreak
        It is incompatible with the minitoc package}}{}%
```

5.76.1.5 Hint about the placeins package

\@ifpackageloaded
\if@mtc@ss@insection@
\@ifpackagewith
\if@mtc@hints@given@

We test if the placeins package [3] is loaded and, if yes, we check that the right options are selected:

```
\@ifpackageloaded{placeins}%
   {\if@mtc@ss@insection@
    \@ifpackagewith{placeins}{section,below}{}%
       {\@mtc@hints@given@true
        \PackageWarningNoLine{minitoc(hints)}%
           {--- The placeins package is loaded without
            \MessageBreak
            section and below options, but minitoc used
            \MessageBreak
            the insection option which implies them.
```

```
          \MessageBreak
          Try to inverse the loading order and use coherent options.
          \MessageBreak
          You may have got a message
            ! LaTeX Error: Option clash for package placeins}%
     }%
   \fi
  \@ifpackagelater{placeins}{2005/04/18}{}{%
     \@mtc@hints@given@true
     \PackageWarningNoLine{minitoc(hints)}%
{--- The placeins package loaded is
 \MessageBreak
 too old. It should have a version date of
 \MessageBreak
 2005/04/18 at least}%
  }%
  }{}%
```

5.76.1.6 Hint about the memoir class

\@ifclassloaded
\if@mtc@hints@given@

We test if the memoir class [48] is loaded:

```
\@ifclassloaded{memoir}%
  {\@mtc@hints@given@true
   \PackageInfo{minitoc(hints)}%
      {--- The memoir class is loaded. \MessageBreak
       See the minitoc package documentation for specific precautions\@gobble}{}%
  }{}%
```

5.76.1.7 Hint about the amsart and amsproc classes

\@ifclassloaded
\if@mtc@hints@given@

We test is the amsart or amsproc class is loaded and emit a warning, because thess classes are incompatible with minitoc:

```
\@ifclassloaded{amsart}%
  {\@mtc@hints@given@true
   \PackageWarningNoLine{minitoc(hints)}%
      {--- The amsart class is loaded. \MessageBreak
       It is incompatible with the minitoc package}}{}%
\@ifclassloaded{amsproc}%
  {\@mtc@hints@given@true
   \PackageWarningNoLine{minitoc(hints)}%
      {--- The amsproc class is loaded. \MessageBreak
       It is incompatible with the minitoc package}}{}%
```

5.76.1.8 Hint about the amsbook class

\@ifclassloaded
\if@mtc@hints@given@

We test if the amsbook class is loaded:

```
\@ifclassloaded{amsbook}%
   {\@mtc@hints@given@true
    \PackageInfo{minitoc(hints)}%
       {--- The amsbook class is loaded. \MessageBreak
        See the minitoc package documentation for specific precautions\@gobble}{}%
   }{}%
```

5.76.1.9 Hint about the abstract package

\@ifpackageloaded
\if@mtc@abstract@loaded@
\@ifpackagewith
\if@mtc@hints@given@

We test the presence of the abstract package [45], then its options:

```
\@ifpackageloaded{abstract}%
   {\@mtc@abstract@loaded@true%
   \@ifpackagewith{abstract}{addtotoc}%
      {\@mtc@hints@given@true
       \PackageInfo{minitoc(hints)}%
          {The ‘‘abstract’’ package has been loaded with\MessageBreak
           the ‘‘addtotoc’’ option.\MessageBreak
           You need to look at the documentation to adjust.}
      }}{}
```

5.76.1.10 Hint about the alteration of the sectionning commands

\part
\chapter
\section
\AtBeginDocument
\mtc@hints@part
\mtc@hints@chapter
\mtc@hints@section

To check if the sectionning commands \part, \chapter or \section have been altered by some package or in the preamble, we compare them (when executing a \AtBeginDocument block) with their saved versions (saved by the minitoc package when it is loaded) \mtc@hints@part, \mtc@hints@chapter and \mtc@hints@section. For each sectionning command, we must perform the comparaison for the command itself, its unstarred branch and its starred branch. But the hyperref package [39] may interfere, hence the formal precautions in the messages.

5.76.1.10.1 Alteration of \part :

\if@mtc@hints@w@
\@ifundefined
\part
\mtc@hints@part
\@part
\mtc@hints@@part
\@spart
\mtc@hints@@spart
\if@mtc@hints@given@
\if@mtc@hyper@used@

We check the alteration of \part, \@part and \@spart:

```
\@mtc@hints@w@false
\@ifundefined{part}{}{\ifx\part\mtc@hints@part\relax
   \else\@mtc@hints@w@true\fi}
\@ifundefined{part}{}{\ifx\@part\mtc@hints@@part\relax
```

```
    \else\@mtc@hints@w@true\fi}
\@ifundefined{part}{}{\ifx\@spart\mtc@hints@@spart\relax
    \else\@mtc@hints@w@true\fi}
\if@mtc@hints@w@\@mtc@hints@given@true%
  \PackageWarningNoLine{minitoc(hints)}%
     {--- The \string\part\space command is altered after minitoc}
  \if@mtc@hyper@used@
     \PackageWarningNoLine{minitoc(hints)}%
        {--- It may be the consequence of loading the ‘‘hyperref’’ package}
  \fi
\fi
```

5.76.1.10.2 Alteration of `\chapter` :

\if@mtc@hints@w@
\@ifundefined
\chapter
\mtc@hints@chapter
\@chapter
\mtc@hints@@chapter
\@schapter
\mtc@hints@@schapter
\if@mtc@hints@given@
\if@mtc@hyper@used@

We check the alteration of `\chapter`, `\@chapter` and `\@schapter`:

```
\@mtc@hints@w@false
\@ifundefined{chapter}{}{\ifx\chapter\mtc@hints@chapter\relax
    \else\@mtc@hints@w@true\fi}%
\@ifundefined{chapter}{}{\ifx\@chapter\mtc@hints@@chapter\relax
    \else\@mtc@hints@w@true\fi}%
\@ifundefined{chapter}{}{\ifx\@schapter\mtc@hints@@schapter\relax
    \else\@mtc@hints@w@true\fi}%
\if@mtc@hints@w@\@mtc@hints@given@true%
  \PackageWarningNoLine{minitoc(hints)}%
     {--- The \string\chapter\space command is altered after minitoc}
  \if@mtc@hyper@used@
     \PackageWarningNoLine{minitoc(hints)}%
        {--- It may be the consequence of loading the ‘‘hyperref’’ package}
  \fi
\fi
```

5.76.1.10.3 Alteration of `\section` :

\if@mtc@hints@w@
\@ifundefined
\section
\mtc@hints@section
\@sect
\mtc@hints@@sect
\@ssect
\mtc@hints@@ssect
\if@mtc@hints@given@
\if@mtc@hyper@used@

We check the alteration of `\section`, `\@sect` and `\@ssect`:

```
\@mtc@hints@w@false
\@ifundefined{chapter}%
   {\@ifundefined{section}{}{\ifx\section\mtc@hints@section\relax\else
      \@mtc@hints@w@true\fi}
    \@ifundefined{section}{}{\ifx\@sect\mtc@hints@@sect\relax\else
      \@mtc@hints@w@true\fi}
    \@ifundefined{section}{}{\ifx\@ssect\mtc@hints@@ssect\relax\else
      \@mtc@hints@w@true\fi}
\if@mtc@hints@w@\@mtc@hints@given@true%
  \PackageWarningNoLine{minitoc(hints)}%
     {--- The \string\section\space command is altered after minitoc}
  \if@mtc@hyper@used@
```

```
    \PackageWarningNoLine{minitoc(hints)}%
       {--- It may be the consequence of loading the ‘‘hyperref’’ package}
\fi
\relax\else\fi}{}%
```

5.76.1.11 Hint about the coherence of the calling sequences of some commands

And finally, we prepare the coherence tests about the calling sequences of triplets of associated commands like \doparttoc, \parttoc and \[fake]listofcontents, and similar: to be able to use \parttoc, a table of contents file must have been created via \[fake]listofcontents and splitted into parttoc files via \doparttoc.

\if@mtc@hints@
\if@mtc@toc@used@
\if@mtc@lof@used@
\if@mtc@lot@used@

Hence we declare some flags:

```
\if@mtc@hints@
\newif\if@mtc@toc@used@ \global\@mtc@toc@used@false
\newif\if@mtc@lof@used@ \global\@mtc@lof@used@false
\newif\if@mtc@lot@used@ \global\@mtc@lot@used@false
```

\mtc@sv@tableofcontents
\tableofcontents
\if@mtc@toc@used@
\mtc@sv@listoffigures
\listoffigures
\if@mtc@lof@used@
\mtc@sv@listoftables
\listoftables
\if@mtc@lot@used@

Then we patch the involved commands to set the corresponding flag when they are used. First, the commands for the main summaries:

```
\let\mtc@sv@tableofcontents\tableofcontents
\def\tableofcontents%
   {\global\@mtc@toc@used@true\mtc@sv@tableofcontents}
\let\mtc@sv@listoffigures\listoffigures
\def\listoffigures%
   {\global\@mtc@lof@used@true\mtc@sv@listoffigures}
\let\mtc@sv@listoftables\listoftables
\def\listoftables%
   {\global\@mtc@lot@used@true\mtc@sv@listoftables}
```

\mtc@sv@faketableofcontents
\faketableofcontents
\if@mtc@toc@used@
\mtc@sv@fakelistoffigures
\fakelistoffigures
\if@mtc@lof@used@
\mtc@sv@fakelistoftables
\fakelistoftables
\if@mtc@lot@used@

Then, their “fake” siblings:

```
\let\mtc@sv@faketableofcontents\faketableofcontents
\def\faketableofcontents%
   {\global\@mtc@toc@used@true\mtc@sv@faketableofcontents}
\let\mtc@sv@fakelistoffigures\fakelistoffigures
\def\fakelistoffigures%
   {\global\@mtc@lof@used@true\mtc@sv@fakelistoffigures}
\let\mtc@sv@fakelistoftables\fakelistoftables
\def\fakelistoftables%
   {\global\@mtc@lot@used@true\mtc@sv@fakelistoftables}
\fi
```

\mtc@hints@begindoc

And the \mtc@hints@begindoc definition is finished (it begins in section 5.76.1 on page 191):

```
}
```

5.76.2 Final part: \mtc@hints@enddoc

\mtc@hints@enddoc
\AtEndDocument

The final part of the hints option is executed via \AtEndDocument. Its code is in the \mtc@hints@enddoc macro. It is a sequence of tests on the packages or classes loaded and the flags set during the first and the second parts of this option. First, we declare the \mtc@hints@enddoc macro:

```
\def\mtc@hints@enddoc{%
```

5.76.2.1 Hint about \sect-lof|lot and the insection option

\if@mtc@sect@floats@
\if@dosectlof@used@
\if@dosectlot@used@
\if@sectlof@used@
\if@sectlot@used@
\if@mtc@section@def@

We look if some section-level lists of figures or tables have been requested.

```
\if@dosectlof@used@\@mtc@sect@floats@true\fi
\if@dosectlot@used@\@mtc@sect@floats@true\fi
\if@sectlof@used@\@mtc@sect@floats@true\fi
\if@sectlot@used@\@mtc@sect@floats@true\fi
\if@mtc@section@def@
```

\if@mtc@placeinsLoaded@
\if@mtc@sect@floats@
\if@mtc@hints@w@
\if@mtc@hints@given@

If yes, we verify that the placeins package [3] has been loaded with the correct options or that the insection option of the minitoc package has been invoked. If not, a warning is given.

```
\if@mtc@placeinsLoaded@
\else
  \if@mtc@sect@floats@%
   \PackageWarningNoLine{minitoc(hints)}%
      {***** You are using \string\dosectlof\space and/or \MessageBreak
       \string\dosectlot, \string\sectlof\space and/or \string\sectlot,
       \MessageBreak
       hence the ‘‘insection’’ package option is recommended. *****}%
   \@mtc@hints@w@true
   \@mtc@hints@given@true
  \fi
\fi
```

5.76.2.2 Final part of the coherence tests

We test if \parttoc has been used without \doparttoc, etc., for each pair of preparation/insertion commands.

\if@mtc@part@def@
\if@parttoc@used@
\if@doparttoc@used@
\if@mtc@hints@given@
\if@partlof@used@
\if@dopartlof@used@
\if@partlot@used@
\if@dopartlot@used@

For the part level commands:

```
\if@mtc@part@def@
 \if@parttoc@used@
   \if@doparttoc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\parttoc\space
       but not \string\doparttoc\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@partlof@used@
   \if@dopartlof@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\partlof\space
       but not \string\dopartlof\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@partlot@used@
   \if@dopartlot@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\partlot\space
       but not \string\dopartlot\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
\fi
```

\if@mtc@chapter@def@
\if@minitoc@used@
\if@dominitoc@used@
\if@mtc@hints@given@
\if@minilof@used@
\if@dominilof@used@
\if@minilot@used@
\if@dominilot@used@

For the chapter level commands:

```
\if@mtc@chapter@def@
 \if@minitoc@used@
   \if@dominitoc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\minitoc\space
       but not \string\dominitoc\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@minilof@used@
   \if@dominilof@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\minilof\space
       but not \string\dominilof\space*****}
   \@mtc@hints@given@true
```

```
  \fi
 \fi
 \if@minilot@used@
  \if@dominilot@used@\else
  \PackageWarningNoLine{minitoc(hints)}%
     {**** You have used \string\minilot\space
      but not \string\dominilot\space*****}
  \@mtc@hints@given@true
  \fi
 \fi
\fi
```

\if@mtc@section@def@
\if@secttoc@used@
\if@dosecttoc@used@
\if@mtc@hints@given@
\if@sectlof@used@
\if@dosectlof@used@
\if@sectlot@used@
\if@dosectlot@used@

For the section level commands:

```
\if@mtc@section@def@
 \if@secttoc@used@
  \if@dosecttoc@used@\else
  \PackageWarningNoLine{minitoc(hints)}%
     {**** You have used \string\secttoc\space
      but not \string\dosecttoc\space*****}
  \@mtc@hints@given@true
  \fi
 \fi
 \if@sectlof@used@
  \if@dosectlof@used@\else
  \PackageWarningNoLine{minitoc(hints)}%
     {**** You have used \string\sectlof\space
      but not \string\dosectlof\space*****}
  \@mtc@hints@given@true
  \fi
 \fi
 \if@sectlot@used@
  \if@dosectlot@used@\else
  \PackageWarningNoLine{minitoc(hints)}%
     {**** You have used \string\sectlot\space
      but not \string\dosectlot\space*****}
  \fi
 \fi
\fi
```

5.76.2.3 Check if the main tables have been prepared (first part)

Now, we test if a \doparttoc macro has been called but without any matching \parttoc, hence its a vain call. We do the same for each analog command.

\if@mtc@part@def@
\if@doparttoc@used@
\if@parttoc@used@
\if@mtc@hints@given@
\if@dopartlof@used@
\if@partlof@used@
\if@dopartlot@used@
\if@partlot@used@

Part level commands:

```
\if@mtc@part@def@
```

```
 \if@doparttoc@used@
   \if@parttoc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\doparttoc\space
       but not \string\parttoc\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@dopartlof@used@
   \if@partlof@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dopartlof\space
       but not \string\partlof\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@dopartlot@used@
   \if@partlot@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dopartlot\space
       but not \string\partlot\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
\fi
```

\if@mtc@chapter@def@
\if@dominitoc@used@
\if@minitoc@used@
\if@mtc@hints@given@
\if@dominilof@used@
\if@minilof@used@
\if@dominilot@used@
\if@minilot@used@

Chapter level commands:

```
\if@mtc@chapter@def@
 \if@dominitoc@used@
   \if@minitoc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dominitoc\space
       but not \string\minitoc\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@dominilof@used@
   \if@minilof@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dominilof\space
       but not \string\minilof\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@dominilot@used@
   \if@minilot@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dominilot\space
       but not \string\minilot\space*****}
   \@mtc@hints@given@true
   \fi
```

```
 \fi
\fi
```

\if@mtc@section@def@
\if@dosecttoc@used@
\if@secttoc@used@
\if@mtc@hints@given@
\if@dosectlof@used@
\if@sectlof@used@
\if@dosectlot@used@
\if@sectlot@used@

Section level commands:

```
\if@mtc@section@def@
 \if@dosecttoc@used@
   \if@secttoc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dosecttoc\space
       but not \string\secttoc\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@dosectlof@used@
   \if@sectlof@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dosectlof\space
       but not \string\sectlof\space*****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@dosectlot@used@
   \if@sectlot@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\dosectlot\space
       but not \string\sectlot\space*****}
   \fi
 \fi
\fi
\fi
```

5.76.2.4 Check if the main tables have been prepared (second part)

Another coherence test verifies that if \parttoc has been called, then the macro \tableofcontents or \faketableofcontents has also been called (to create the necessary contents file); and similar tests are made for the other mini-table commands.

\if@mtc@part@def@
\if@mtc@hints@given@
\if@parttoc@used@
\ifmtc@toc@used@
\if@partlof@used@
\ifmtc@lof@used@
\if@partlot@used@
\ifmtc@lot@used@

Part level commands:

```
\if@mtc@part@def@
 \if@parttoc@used@
   \if@mtc@toc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\parttoc\space but not\MessageBreak
       \string\tableofcontents\space nor
       \string\faketableofcontents\space *****}
   \@mtc@hints@given@true
```

```
    \fi
  \fi
  \if@partlof@used@
    \if@mtc@lof@used@\else
    \PackageWarningNoLine{minitoc(hints)}%
       {**** You have used \string\partlof\space but not\MessageBreak
        \string\listoffigures\space nor
        \string\fakelistoffigures\space *****}
    \@mtc@hints@given@true
    \fi
  \fi
  \if@partlot@used@
    \if@mtc@lot@used@\else
    \PackageWarningNoLine{minitoc(hints)}%
       {**** You have used \string\partlot\space but not\MessageBreak
        \string\listoftables\space nor
        \string\fakelistoftables\space *****}
    \@mtc@hints@given@true
    \fi
  \fi
\fi
```

\if@mtc@chapter@def@
\if@mtc@hints@given@
\if@minitoc@used@
\ifmtc@toc@used@
\if@minilof@used@
\ifmtc@lof@used@
\if@minilot@used@
\ifmtc@lot@used@

Chapter level commands:

```
\if@mtc@chapter@def@
  \if@minitoc@used@
    \if@mtc@toc@used@\else
    \PackageWarningNoLine{minitoc(hints)}%
       {**** You have used \string\minitoc\space but not\MessageBreak
        \string\tableofcontents\space nor
        \string\faketableofcontents\space *****}
    \@mtc@hints@given@true
    \fi
  \fi
  \if@minilof@used@
    \if@mtc@lof@used@\else
    \PackageWarningNoLine{minitoc(hints)}%
       {**** You have used \string\minilof\space but not\MessageBreak
        \string\listoffigures\space nor
        \string\fakelistoffigures\space *****}
    \@mtc@hints@given@true
    \fi
  \fi
  \if@minilot@used@
    \if@mtc@lot@used@\else
    \PackageWarningNoLine{minitoc(hints)}%
       {**** You have used \string\minilot\space but not\MessageBreak
        \string\listoftables\space nor
        \string\fakelistoftables\space *****}
    \@mtc@hints@given@true
    \fi
  \fi
```

```
\fi
```

\if@mtc@section@def@
\if@mtc@hints@given@
\if@secttoc@used@
\ifmtc@toc@used@
\if@sectlof@used@
\ifmtc@lof@used@
\if@sectlot@used@
\ifmtc@lot@used@

Section level commands:

```
\if@mtc@section@def@
 \if@secttoc@used@
   \if@mtc@toc@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\secttoc\space but not\MessageBreak
       \string\tableofcontents\space nor
       \string\faketableofcontents\space *****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@sectlof@used@
   \if@mtc@lof@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\sectlof\space but not\MessageBreak
      \string\listoffigures\space nor
      \string\fakelistoffigures\space *****}
   \@mtc@hints@given@true
   \fi
 \fi
 \if@sectlot@used@
   \if@mtc@lot@used@\else
   \PackageWarningNoLine{minitoc(hints)}%
      {**** You have used \string\sectlot\space but not\MessageBreak
       \string\listoftables\space nor
       \string\fakelistoftables\space *****}
   \@mtc@hints@given@true
   \fi
 \fi
\fi
```

5.76.2.5 Check the number of mini-tables, in case of short extensions

\mtc@hints@checklongext
\if@longextensions@
\if@mtc@part@def@
\value
\if@mtc@hints@given@true
\if@mtc@chapter@def@
\if@mtc@section@def@

If short extensions are used, you can use only 99 mini-tables of each kind. If more are created, the auxiliary files can be overwritten: the hundredth minitoc file \jobname.U100 has its name truncated to \jobname.U10, which is already the tenth minitoc file. Thus, we need a hint to signal this situation. The code is rather simple, but the remedy is bitter and costly: either use a better operating system [9], either redesign the document.

```
\def\mtc@hints@checklongext{%
\if@longextensions@
\else
  \if@mtc@part@def@
    \ifnum 99 < \value{ptc}
```

[9] On the long term, a good investment.

```
      \@mtc@hints@given@true
      \PackageWarningNoLine{minitoc(hints)}%
         {**** You have used short extensions
          and more than 99 parts *****}
    \fi
  \fi
  \if@mtc@chapter@def@
    \ifnum 99 < \value{mtc}
      \@mtc@hints@given@true
      \PackageWarningNoLine{minitoc(hints)}%
         {**** You have used short extensions
          and more than 99 chapters *****}
    \fi
  \else
    \if@mtc@section@def@
      \ifnum 99 < \value{stc}
        \@mtc@hints@given@true
        \PackageWarningNoLine{minitoc(hints)}%
           {**** You have used short extensions
            and more than 99 sections *****}
      \fi
    \fi
  \fi
\fi}
\mtc@hints@checklongext
```

5.76.2.6 Final part of the hint about the sectsty package

\if@mtc@sectstyLoaded@
\if@mtc@sectstyLoaded@a@
\if@mtc@hints@given@

We test if sectsty has been loaded before (correct) or after (incorrect) minitoc. See section 5.9 on page 76.

```
\if@mtc@sectstyLoaded@\else
  \if@mtc@sectstyLoaded@a@
    \PackageWarningNoLine{minitoc(hints)}%
    {**** The sectsty package should be loaded \MessageBreak
          **before** the minitoc package ****}
    \@mtc@hints@given@true
  \fi
\fi
```

5.76.2.7 Check if empty mini-tables have been detected

We test for each kind of mini-tables.

\if@mtc@empty@parttoc@ For parttocs:

```
\if@mtc@empty@parttoc@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty parttocs ****}
   \@mtc@hints@given@true
\fi
```

\if@mtc@empty@partlof@ For partlofs:

```
\if@mtc@empty@partlof@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty partlofs ****}
   \@mtc@hints@given@true
\fi
```

\if@mtc@empty@partlot@ For partlots:

```
\if@mtc@empty@partlot@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty partlots ****}
   \@mtc@hints@given@true
\fi
```

\if@mtc@empty@minitoc@ For minitocs:

```
\if@mtc@empty@minitoc@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty minitocs ****}
   \@mtc@hints@given@true
\fi
```

\if@mtc@empty@minilof@ For minilofs:

```
\if@mtc@empty@minilof@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty minilofs ****}
   \@mtc@hints@given@true
\fi
```

`\if@mtc@empty@minilot@` For minilots:

```
\if@mtc@empty@minilot@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty minilots ****}
   \@mtc@hints@given@true
\fi
```

`\if@mtc@empty@secttoc@` For secttocs:

```
\if@mtc@empty@secttoc@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty secttocs ****}
   \@mtc@hints@given@true
\fi
```

`\if@mtc@empty@sectlof@` For sectlofs:

```
\if@mtc@empty@sectlof@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty sectlofs ****}
   \@mtc@hints@given@true
\fi
```

`\if@mtc@empty@sectlot@` For sectlots:

```
\if@mtc@empty@sectlot@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have attempted to insert empty sectlots ****}
   \@mtc@hints@given@true
\fi
```

5.76.2.8 Check if obsolete commands have been used

This hint is just a reminder if you have used obsolete commands, which are also signalled in the `.log` file.

`\if@firstpartis@used@` Obsolete macro `\firstpartis`:

```
\if@firstpartis@used@
   \PackageWarningNoLine{minitoc(hints)}%
   {**** You have invoked an obsolete command: \string\firstpartis\space ****}
   \@mtc@hints@given@true
\fi
```

\if@firstchapteris@used@

Obsolete macro \firstchapteris:

```
\if@firstchapteris@used@
  \PackageWarningNoLine{minitoc(hints)}%
  {**** You have invoked an obsolete command: \string\firstchapteris\space ****}
  \@mtc@hints@given@true
\fi
```

\if@firstsectionis@used@

Obsolete macro \firstsectionis:

```
\if@firstsectionis@used@
  \PackageWarningNoLine{minitoc(hints)}%
  {**** You have invoked an obsolete command: \string\firstsectionis\space ****}
  \@mtc@hints@given@true
\fi
```

5.76.2.9 Check if some hints have been written

\if@mtc@hints@given@
\mtc@hints@enddoc

We come at the end of the third part of the hints option: if problems have been detected, a warning is displayed; the warning is not displayed but only written in the .log file if no problems have been detected. And we terminate the \mtc@hints@enddoc macro by a closing brace.

```
\if@mtc@hints@given@
  \PackageWarningNoLine{minitoc(hints)}%
    {***** Some hints have been written\MessageBreak
     in the \jobname.log file. *****}
\else
  \PackageInfo{minitoc(hints)}%
    {***** No hints have been written\MessageBreak
     in the \jobname.log file. *****\@gobble}
\fi
}
```

5.77 Processing of options

\ExecuteOptionsenglish

First, we apply the default language option, english:

```
\ExecuteOptions{english}
```

\ProcessOptions*

Then, we execute all requested options: for most options, it is just setting a flag, or loading a file for the language options.

```
\ProcessOptions*
```

We now examine the flags for some options and execute the necessary actions.

5.77.1 Processing the insection option

\if@mtc@ss@insection@
\RequirePackage
\@ifpackageloaded
\if@mtc@placeinsLoaded@

For the insection option, we load the placeins package [3] with its options verbose, section and below; the correct loading is verified:

```
\if@mtc@ss@insection@
   \RequirePackage[section,below,verbose]{placeins}[2005/04/18]%
   \@ifpackageloaded{placeins}%
      {\@mtc@placeinsLoaded@true}%
      {\@mtc@placeinsLoaded@false}%
\fi
```

5.77.2 Processing the notoccite option

\if@mtc@notoccite@
\RequirePackage

For the notoccite option, we just load the notoccite package [2]:

```
\if@mtc@notoccite@
   \RequirePackage{notoccite}%
\fi
```

5.77.3 Processing the hints option

\if@mtc@hints@
\AtBeginDocument
\mtc@hints@begindoc
\AtEndDocument
\mtc@hints@enddoc

For the hints option, we set its first part in a \AtBeginDocument block and its third (last) part in a \AtEndDocument block:

```
\if@mtc@hints@
   \AtBeginDocument{\mtc@hints@begindoc}%
   \AtEndDocument{\mtc@hints@enddoc}%
\fi
```

5.77.4 Saving the sectionning commands

And, at least, we save the definitions of sectionning commands (and of their unstarred and starred branches), for comparaisons (this is a part of the hints option executed in the preamble):

\@ifundefined
\mtc@hints@part
\part
\mtc@hints@@part
\@part
\mtc@hints@@spart
\@spart

For the \part command:

```
\@ifundefined{part}{}{\let\mtc@hints@part\part
                      \let\mtc@hints@@part\@part
                      \let\mtc@hints@@spart\@spart}
```

\@ifundefined
\mtc@hints@chapter
\chapter
\mtc@hints@@chapter
\@chapter
\mtc@hints@@schapter
\@schapter

For the \chapter command:

```
\@ifundefined{chapter}{}{\let\mtc@hints@chapter\chapter
                         \let\mtc@hints@@chapter\@chapter
                         \let\mtc@hints@@schapter\@schapter}
```

\@ifundefined
\mtc@hints@section
\section
\mtc@hints@@sect
\@sect
\mtc@hints@@ssect
\@ssect

For the \section command:

```
\@ifundefined{section}{}{\let\mtc@hints@section\section
                         \let\mtc@hints@@sect\@sect
                         \let\mtc@hints@@ssect\@ssect}
```

And the package is terminated.

```
⟨/minitoc⟩
```

5.78 The fminitoc.dtx file

\jobname
\input

This short file is necessary to create the french documentation. Its rôle is to set \jobname to fminitoc in place of fminitoc. As minitoc.ins generates the files minitoc.lan and fminitoc.lan which sets a language number \LANG, and minitoc.dtx reads \jobname.lan, the documentation can be in several languages (english and french) in minitoc.dtx, the language being selected by \ifcase\LANG\relax ... \or\relax ... \fi constructs. The \relax primitives are necessary to avoid bad surprises.

```
⟨*fminitoc⟩
\ProvidesFile{fminitoc.dtx}[2005/09/16 minitoc french documentation start file]
\input minitoc.dtx
⟨/fminitoc⟩
```

Chapter 6

Commented code of mtcoff.sty

Contents

6.1 Why mtcoff.sty?

The minitoc package [20, 19] requires that the user inserts many commands in the source code of her/his document, and not only into the preamble of the document. Hence the concept of a replacement package, mtcoff (means “minitoc off”), which substitutes to all commands and environments of the minitoc package some alternative commands and environments with the same names and syntaxes, but doing nothing (except emitting some harmless warnings, for special cases). This way, to turn off easily the minitoc package, you just have to write, in the preamble of your document, something like:

```
\usepackage[...options...]{minitoc}
%\usepackage{mtcoff}
```

then the minitoc package is activated with the specified options. If you modify this two lines this way:

```
%\usepackage[...options...]{minitoc}
\usepackage{mtcoff}
```

then the minitoc package is desactivated and all its commands and environments are ignored. This is much easier, faster and safer than commenting out all the commands and environments of minitoc. Moreover, this operation is reversible.

6.2 Identification of the package

\NeedsTeXFormat
\ProvidesPackage

First, we identify the package and check the version of LaTeX [1]:

```
⟨*mtcoff⟩
\NeedsTeXFormat{LaTeX2e}%
\ProvidesPackage{mtcoff}[2005/09/16 v43 The mtcoff package]
```

6.3 Faking counters and dimensions

\count@
\dimen@
\c@minitocdepth
\mtcindent
\mtcskipamount
\bigskipamount
\c@parttocdepth
\mtcindent
\c@secttocdepth
\stcindent
\c@mtc
\c@ptc
\c@stc

As minitoc declares some counters and dimensions registers, we fake them using \count@ or \dimen@. For \mtcskipamount, we use its default definition, \bigskipamount.

```
\let\c@minitocdepth\count@
\let\mtcindent\dimen@
\let\mtcskipamount\bigskipamount
\let\c@parttocdepth\count@
\let\ptcindent\dimen@
\let\c@secttocdepth\count@
\let\stcindent\dimen@
```

[1] This checking is not really useful for the mtcoff package itself, but it is good to check that your version of LaTeX is not too old to support minitoc.

```
\let\c@mtc\count@
\let\c@ptc\count@
\let\c@stc\count@
```

6.4 Faking simple user commands

\mtcskip \faketableofcontents \fakelistoffigures \fakelistoftables \adjustptc \adjustmtc \adjuststc \decrementptc \decrementmtc \decrementstc \incrementptc \incrementmtc \incrementstc

Some user commands are easy to fake:

```
\let\mtcskip\relax
\let\faketableofcontents\relax
\let\fakelistoffigures\relax
\let\fakelistoftables\relax
\newcommand{\adjustptc}[1][1]{\relax}
\newcommand{\adjustmtc}[1][1]{\relax}
\newcommand{\adjuststc}[1][1]{\relax}
\let\decrementptc\relax
\let\decrementmtc\relax
\let\decrementstc\relax
\let\incrementptc\relax
\let\incrementmtc\relax
\let\incrementstc\relax
```

\partend \partbegin \chapterend \chapterbegin \sectend \sectbegin

The following commands are not directly called by the user, in normal circumstances, but must be faked:

```
\let\partend\relax
\let\partbegin\relax
\let\chapterend\relax
\let\chapterbegin\relax
\let\sectend\relax
\let\sectbegin\relax
```

6.5 Faking user commands with one optional argument

\gobbleopt@

The user commands with an optional argument are faked using the internal LaTeX macro `\@ifnextchar` (to get the optional argument) and the new utility command `\gobbleopt@`.

```
\def\gobbleopt@[#1]{\relax}
```

\@ifnextchar \doparttoc \dopartlof \dopartlot \parttoc \partlof \partlot

Commands for part level mini-tables:

```
\def\doparttoc{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\dopartlof{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\dopartlot{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\parttoc{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\partlof{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\partlot{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
```

\@ifnextchar \dominitoc \dominilof \dominilot \minitoc \minilof \minilot

Commands for chapter level mini-tables:

```
\def\dominitoc{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\dominilof{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\dominilot{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\minitoc{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\minilof{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\minilot{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
```

\@ifnextchar \dosecttoc \dosectlof \dosectlot \secttoc \sectlof \sectlot

Commands for section level mini-tables:

```
\def\dosecttoc{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\dosectlof{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\dosectlot{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\secttoc{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\sectlof{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
\def\sectlot{\@ifnextchar[{\gobbleopt@}{\gobbleopt@[l]}}
```

6.6 Disabling the internal commands

\@gobbletwo \tf@mtc \mtc@string \appendixmtc \l@xchapter \xchapter \pchapter \psection \l@xpart \xpart \l@xsect \xsect

We need also to disable some `minitoc` commands, with `\relax` (macros with no argument) or `\@gobbletwo` (macros with two arguments):

```
\let\tf@mtc\count@
\let\mtc@string\relax
\let\appendixmtc\relax
\let\l@xchapter\@gobbletwo
\let\xchapter\relax
\let\pchapter\relax
\let\psection\relax
\let\l@xpart\@gobbletwo
\let\xpart\relax
\let\l@xsect\@gobbletwo
\let\xsect\relax
```

6.7 Disabling the font commands

\empty

We disable the minitoc font commands (like \mtcSSfont) with \empty, because some users might have used:

```
\renewcommand{\mtcSSfont}{...}
```

which will not work if we use \relax here.

\ptcfont \ptcCfont \ptcSfont \ptcSSfont \ptcSSSfont \ptcPfont \ptcSPfont \plffont \pltfont \ptifont

Fonts for part level mini-tables:

```
\let\ptcfont\empty
\let\ptcCfont\empty
\let\ptcSfont\empty
\let\ptcSSfont\empty
\let\ptcSSSfont\empty
\let\ptcPfont\empty
\let\ptcSPfont\empty
\let\plffont\empty
\let\pltfont\empty
\let\ptifont\empty
```

\mtcfont \mtcSfont \mtcSSfont \mtcSSSfont \mtcPfont \mtcSPfont \mlffont \mltfont \mtifont

Fonts for chapter level mini-tables:

```
\let\mtcfont\empty
\let\mtcSfont\empty
\let\mtcSSfont\empty
\let\mtcSSSfont\empty
\let\mtcPfont\empty
\let\mtcSPfont\empty
\let\mlffont\empty
\let\mltfont\empty
\let\mtifont\empty
```

\stcfont \stcSSfont \stcSSSfont \stcPfont \stcSPfont \slffont \sltfont \stifont

Fonts for section level mini-tables:

```
\let\stcfont\empty
\let\stcSSfont\empty
\let\stcSSSfont\empty
\let\stcPfont\empty
\let\stcSPfont\empty
\let\slffont\empty
\let\sltfont\empty
\let\stifont\empty
```

\coffeefont

Font for “coffee” lines:

```
\let\coffeefont\empty
```

6.8 Disabling the \mtcset... commands

\@gobbletwo
\mtcsetdepth
\mtcsetfont
\mtcsettitlefont
\mtcsettitle
\mtcsetformat
\mtcsetfeature
\mtcsetpagenumbers
\mtcsetrules

These commands use two or three mandatory arguments:

```
\let\mtcsetdepth\@gobbletwo
\def\mtcsetfont#1#2#3{\empty}
\let\mtcsettitlefont\@gobbletwo
\let\mtcsettitle\@gobbletwo
\def\mtcsetformat#1#2#3{\empty}
\def\mtcsetfeature#1#2#3{\empty}
\let\mtcsetpagenumbers\@gobbletwo
\let\mtcsetrules\@gobbletwo
```

6.9 Disabling the new \l@... commands

\l@starpart
\l@starchapter
\l@starsection
\l@starsubsection
\l@starsubsubsection
\l@starparagraph
\l@starsubparagraph

The minitoc package defines the \l@starXXX commands to format TOC entries for starred sectionning commands. We reset to the unstarred version, when necessary:

```
\@ifundefined{part}{}{\let\l@starpart\l@part}
\@ifundefined{chapter}{}{\let\l@starchapter\l@chapter}
\@ifundefined{section}{}{\let\l@starsection\l@section}
\@ifundefined{subsection}{}{\let\l@starsubsection\l@subsection}
\@ifundefined{subsubsection}{}{\let\l@starsubsubsection\l@subsubsection}
\@ifundefined{paragraph}{}{\let\l@starparagraph\l@paragraph}
\@ifundefined{subparagraph}{}{\let\l@starsubparagraph\l@subparagraph}
```

6.10 Ignore the obsolete commands

\@gobble
\firstpartis
\firstchapteris
\firstsectionis

We just ignore the obsolete commands (with one mandatory argument):

```
\let\firstpartis\@gobble
\let\firstchapteris\@gobble
\let\firstsectionis\@gobble
```

6.11 Disabling the \mtcselectlanguage command

\@gobble
\mtcselectlanguage

This command has one mandatory argument:

```
\let\mtcselectlanguage\@gobble
```

6.12 Disabling the \mtcloadmlo command

\@gobble
\mtcloadmlo

This command has one mandatory argument:

```
\let\mtcloadmlo\@gobble
```

6.13 Disabling the commands for the horizontal rules

\ptcrule
\noptcrule
\mtcrule
\nomtcrule
\stcrule
\nostcrule
\plfrule
\noplfrule
\mlfrule
\nomlfrule
\slfrule
\noslfrule
\pltrule
\nopltrule
\mltrule
\nomltrule
\sltrule
\nosltrule

These commands have no argument:

```
\let\ptcrule\relax
\let\noptcrule\relax
\let\mtcrule\relax
\let\nomtcrule\relax
\let\stcrule\relax
\let\nostcrule\relax
\let\plfrule\relax
\let\noplfrule\relax
\let\mlfrule\relax
\let\nomlfrule\relax
\let\slfrule\relax
\let\noslfrule\relax
\let\pltrule\relax
\let\nopltrule\relax
\let\mltrule\relax
\let\nomltrule\relax
\let\sltrule\relax
\let\nosltrule\relax
```

6.14 Disabling the commands for the page numbers

\ptcpagenumbers
\noptcpagenumbers
\mtcpagenumbers
\nomtcpagenumbers
\stcpagenumbers
\nostcpagenumbers
\plfpagenumbers
\noplfpagenumbers
\mlfpagenumbers
\nomlfpagenumbers
\slfpagenumbers
\noslfpagenumbers
\pltpagenumbers
\nopltpagenumbers
\mltpagenumbers
\nomltpagenumbers
\sltpagenumbers
\nosltpagenumbers

These commands have no argument:

```
\let\mtcpagenumbers\relax
\let\nomtcpagenumbers\relax
\let\stcpagenumbers\relax
\let\nostcpagenumbers\relax
\let\ptcpagenumbers\relax
\let\noptcpagenumbers\relax
\let\mlfpagenumbers\relax
\let\nomlfpagenumbers\relax
\let\slfpagenumbers\relax
\let\noslfpagenumbers\relax
\let\plfpagenumbers\relax
\let\noplfpagenumbers\relax
\let\mltpagenumbers\relax
\let\nomltpagenumbers\relax
\let\sltpagenumbers\relax
\let\nosltpagenumbers\relax
\let\pltpagenumbers\relax
\let\nopltpagenumbers\relax
```

6.15 Disabling the mini-table features commands

We disable the commands for features (like \beforeparttoc) with \empty, because some users may have used:

```
\renewcommand{\beforeparttoc}{...}
```

which will not work if we use \relax here. These commands have no argument,

\beforeparttoc
\beforepartlof
\beforepartlot
\afterparttoc
\afterpartlof
\afterpartlot
\thispageparttocstyle
\thispagepartlofstyle
\thispagepartlotstyle

Commands for part level mini-tables:

```
\let\beforeparttoc\empty
\let\beforepartlof\empty
\let\beforepartlot\empty
\let\afterparttoc\empty
\let\afterpartlof\empty
\let\afterpartlot\empty
\let\thispageparttocstyle\empty
\let\thispagepartlofstyle\empty
\let\thispagepartlotstyle\empty
```

\beforeminitoc \beforeminilof \beforeminilot \afterminitoc \afterminilof \afterminilot \thispageminitocstyle \thispageminilofstyle \thispageminilotstyle

Commands for chapter level mini-tables:

```
\let\beforeminitoc\empty
\let\beforeminilof\empty
\let\beforeminilot\empty
\let\afterminitoc\empty
\let\afterminilof\empty
\let\afterminilot\empty
\let\thispageminitocstyle\empty
\let\thispageminilofstyle\empty
\let\thispageminilotstyle\empty
```

\beforesecttoc \beforesectlof \beforesectlot \aftersecttoc \aftersectlof \aftersectlot \thispagesecttocstyle \thispagesectlofstyle \thispagesectlotstyle

Commands for section level mini-tables:

```
\let\beforesecttoc\empty
\let\beforesectlof\empty
\let\beforesectlot\empty
\let\aftersecttoc\empty
\let\aftersectlof\empty
\let\aftersectlot\empty
\let\thispagesecttocstyle\empty
\let\thispagesectlofstyle\empty
\let\thispagesectlotstyle\empty
```

6.16 Disabling miscellaneous flags and commands

\if@longextensions@ \iftightmtc \ifktightmtc \ifundottedmtc \l@listof \chapter

There are some flags and commands that it is wise to declare:

```
\newif\if@longextensions@ \@longextensions@true
\newif\iftightmtc \tightmtcfalse
\newif\ifktightmtc \ktightmtcfalse
\newif\ifundottedmtc \undottedmtcfalse
\let\l@listof\chapter
```

6.17 Caution for some commands

\AtBeginDocument \ifmtcoffwarn@

Some `minitoc` commands should eventually be replaced if you decide to *definitely* stop using the `minitoc` package with your document. So we declare a flag and a `\AtEndDocument` block to signal that you have used these commands:

```
\newif\ifmtcoffwarn@ \mtcoffwarn@false
\AtEndDocument{\ifmtcoffwarn@
   \PackageWarningNoLine{mtcoff}%
```

```
     {You should scan (backwards) your .log file to find
      \MessageBreak
      some commands needing to be replaced if you decide to
      \MessageBreak
      DEFINITELY stop using minitoc for this document.
      \MessageBreak
      It is more wise to keep the \string\usepackage\space lines for
      \MessageBreak
      minitoc and mtcoff and to comment out only one of them}
\fi}
```

\mtcaddchapter
\mtcaddsection
\mtcaddpart
\ifmtcoffwarn@
\mtc@ck
\addcontentsline

Then these commands are disabled and they set the flag and give a warning (useful to get the line number):

```
\newcommand{\mtcaddchapter}[1][]{\mtcoffwarn@true
   \PackageWarning{mtcoff}%
   {\protect\mtcaddchapter{...} should be replaced\MessageBreak
    by \protect\addcontentsline{toc}{chapter}{...}\MessageBreak}
\def\mtc@ck{#1}
\ifx\mtc@ck\empty
\else
   \addcontentsline{toc}{chapter}{#1}%
\fi}
\newcommand{\mtcaddsection}[1][]{\mtcoffwarn@true
   \PackageWarning{mtcoff}%
      {\protect\mtcaddsection{...} should be replaced\MessageBreak
       by \protect\addcontentsline{toc}{section}{...}\MessageBreak}
\def\mtc@ck{#1}
\ifx\mtc@ck\empty
\else
     \addcontentsline{toc}{part}{#1}%
\fi}
\newcommand{\mtcaddpart}[1][]{\mtcoffwarn@true
   \PackageWarning{mtcoff}%
      {\protect\mtcaddpart{...} should be replaced\MessageBreak
       by \protect\addcontentsline{toc}{part}{...}\MessageBreak}
\def\mtc@ck{#1}
\ifx\mtc@ck\empty
\else
   \addcontentsline{toc}{part}{#1}%
\fi}
```

6.18 Disabling commands for “coffee”

\addcoffeeline
\coffeeline
\@gobble
\@Undottedtocline
\@Undottedtoclinep

We disable the commands relative to “coffee” lines, and the specific version of contents lines without leaders of dots:

```
\def\addcoffeeline#1#2#3{\relax}
```

```
\let\coffeeline\@gobble
\let\l@coffee\relax
\def\@Undottedtocline#1#2#3#4#5{\relax}
\def\@Undottedtoclinep#1#2#3#4#5{\relax}
```

6.19 Disabling the mtchideinmain... environments

mtchideinmaintoc
mtchideinmainlof
mtchideinmainlot

These environments accept one optional argument:

```
\newenvironment{mtchideinmaintoc}[1][-1]%
   {\empty}{\empty}
\newenvironment{mtchideinmainlof}[1][-1]%
   {\empty}{\empty}
\newenvironment{mtchideinmainlot}[1][-1]%
   {\empty}{\empty}
```

6.20 Disabling the \mtcfixindex command

\mtcficindex

This command accepts one optional argument:

```
\newcommand{\mtcfixindex}[1][]{\relax}
```

6.21 Disabling the \mtcfixglossary command

\mtcficindex

This command accepts one optional argument:

```
\newcommand{\mtcfixglossary}[1][]{\relax}
```

6.22 Disabling the \addstarred... commands

\ifmtcoffwarn@
\addstarredpart
\addstarredchapter
\addstarredsection
\addcontentsline

These commands should be replaced by standard commands, but mtcoff simulates and gives a warning, which will be reminded at the end of document:

```
\def\addstarredpart#1{\mtcoffwarn@true
   \PackageWarning{mtcoff}%
   {\protect\addstarredpart{...} should be replaced by\MessageBreak
```

```
   \protect\addcontentsline{toc}{part}{...}\MessageBreak}
  \addcontentsline{toc}{part}{#1}}
\def\addstarredchapter#1{\mtcoffwarn@true
  \PackageWarning{mtcoff}%
  {\protect\addstarredchapter{...} should be replaced by\MessageBreak
   \protect\addcontentsline{toc}{chapter}{...}\MessageBreak}
  \addcontentsline{toc}{chapter}{#1}}
\def\addstarredsection#1{\mtcoffwarn@true
  \PackageWarning{mtcoff}%
  {\protect\addstarredsection{...} should be replaced by\MessageBreak
   \protect\addcontentsline{toc}{section}{...}\MessageBreak}
  \addcontentsline{toc}{section}{#1}}
```

And the `mtcoff` package is terminated.

```
⟨/mtcoff⟩
```

Chapter 7

Patch for the memoir class

This code must be loaded to fix an incompatibality of the `minitoc` package with recent versions of the `memoir.cls` class.

```
⟨*mtcpatchmem⟩
\NeedsTeXFormat{LaTeX2e}[1996/06/01]%
\ProvidesPackage{mtcpatchmem}%
   [2005/09/16 v43 Package mtcpatchmem]
\PackageInfo{mtcpatchmem}%
   {*** mtcpatchmem package to patch the memoir class ***\@gobble}
\renewcommand{\@m@mchapter}[1][]{%
  \def\ch@pt@c{#1}% capture first optional arg
   \@ifnextchar[{\@chapter}{\@chapter[]}%
}
\def\@chapter[#1]#2{%
% if |\ch@pt@c| is empty, no [ was found at all. Use |#2| as
% entry for all fields.
   \ifx\ch@pt@c\@empty
     \def\f@rtoc{#2}%
     \def\f@rhdr{#2}%
   \else
% otherwise at least one [ was found. If |#1| is empty then only
% one was found.
     \let\f@rtoc\ch@pt@c
     \ifx\@empty#1\@empty
       \let\f@rhdr\ch@pt@c
     \else
       \def\f@rhdr{#1}%
     \fi
   \fi
   \ifnum \c@secnumdepth >\m@ne
     \if@mainmatter
       \refstepcounter{chapter}%
     \fi
   \fi
```

```
  \chaptermark{\f@rhdr}%
  \ifartopt
    \@makechapterhead{#2}%
    \@afterheading
  \else
    \insertchapterspace
    \if@twocolumn
      \@topnewpage[\@makechapterhead{#2}]%
    \else
      \@makechapterhead{#2}%
    \fi
    \@afterheading
  \fi
  \ifnum \c@secnumdepth >\m@ne
    \if@mainmatter
      \ifanappendix
        \addcontentsline{toc}{appendix}{%
          \protect\chapternumberline{\thechapter}\f@rtoc}%
      \else
        \addcontentsline{toc}{chapter}{%
          \protect\chapternumberline{\thechapter}\f@rtoc}%
      \fi
    \else
      \addcontentsline{toc}{chapter}{\f@rtoc}%
    \fi
  \else
    \addcontentsline{toc}{chapter}{\f@rtoc}%
  \fi
  \ifheadnameref\M@gettitle{\f@rhdr}\else\M@gettitle{\f@rtoc}\fi
}
⟨/mtcpatchmem⟩
```

Chapter 8

The minitoc language definition (`.mld`) and object (`.mlo`) files

Contents

This chapter shows the code of each .mld file. A .mld file is a *minitoc language definition* file, which defines the titles of the mini-tables for a given language. It contains often some comments about its origin, if you need further details. It is loaded either via a package option [1] in the \usepackage command for the minitoc package, either via the command:

\mtcselectlanguage \mtcselectlanguage{⟨*language*⟩}

[1] It can also be a global option for the document.

\ptctitle \plftitle \plttitle \mtctitle \mlftitle \mlttitle \stctitle \slftitle \slttitle

Each .mld file must define the nine following commands (for the mini-tables of contents, mini-lists of figures and mini-lists of tables, at the part, chapter and section levels):

- \ptctitle
- \plftitle
- \plttitle
- \mtctitle
- \mlftitle
- \mlttitle
- \stctitle
- \slftitle
- \slttitle

Many .mld files require special fonts adequate for the corresponding language; as this is a language-dependent issue, the user must set up the correct language and font context for each language, like using the babel package [9, 10, 11], the CJK system [29], the ANTOMEGA system [24], the ArabTeX [25, 26], BangTEX [36], ethiop [6], FarsiTEX (http://www.farsitex.org), guarani [7], Malayalam [1], MonTEX [14, 15], or ArmTEX [16] packages. Note that it is often the *english* name of the language which is used to name the corresponding .mld file.

But for some oriental languages[2], the source of the titles use some *exotic encodings*, difficult to maniputale in a .dtx file, the .mld file is then just a wrapper loading a .mlo file[3], not generated by the .dtx files in the current version of minitoc package. The adequate input encoding must be set up by the user *before* loading the .mld file via \mtcselectlanguage.

filecontents

To go around this limitation, the minitoc.ins file uses filecontents environments to generate the .mlo files.

8.1 "Acadian" language: acadian.mld

The acadian language[4] is just french, so we load the french.mld file (see section 8.43 on page 245):

```
⟨*acadian⟩
\ProvidesFile{acadian.mld}[2004/12/14]
\mtcselectlanguage{french}%
⟨/acadian⟩
```

[2] Mainly for chinese, farsi (iranian) hangul (korean), hanja (korean), japanese and thai variants.
[3] The extension .mlo means *minitoc language object*.
[4] Spoken is some parts of the south of the USA, like Louisiane.

8.2 "Acadien" language: acadien.mld

The "acadien" language [5] is just french ("acadien" is the french term for "acadian"), so we load the french.mld file (see section 8.43 on page 245):

```
⟨*acadien⟩
\ProvidesFile{acadien.mld}[2004/12/14]
\mtcselectlanguage{french}%
⟨/acadien⟩
```

8.3 "Afrikaan" language: afrikaan.mld

The titles for the "afrikaan" language [6] are taken from the babel package [10] [10]:

```
⟨*afrikaan⟩
\ProvidesFile{afrikaan.mld}[1999/03/16]
%% Afrikaan(s) titles for minitoc.sty
\def\ptctitle{Inhoudsopgawe}%
\def\plftitle{Lys van figure}%
\def\plttitle{Lys van tabelle}%
%%
\def\mtctitle{Inhoudsopgawe}%
\def\mlftitle{Lys van figure}%
\def\mlttitle{Lys van tabelle}%
%%
\def\stctitle{Inhoudsopgawe}%
\def\slftitle{Lys van figure}%
\def\slttitle{Lys van tabelle}%
⟨/afrikaan⟩
```

8.4 "Afrikaans" language: afrikaans.mld

"Afrikaans" is a synonym of "afrikaan", so we just load afrikaan.mld (see section 8.3):

```
⟨*afrikaans⟩
\ProvidesFile{afrikaans.mld}[2004/12/14]
\mtcselectlanguage{afrikaan}%
⟨/afrikaans⟩
```

[5] Spoken is some parts of the south of the USA, like Louisiane.
[6] Spoken in South Africa, it has dutch origins; compare with section 8.31 on page 240.

8.5 "American" language: american.mld

The "american" language is just like "english"[7], so we just load `english.mld` (see section 8.32 on page 240):

```
⟨*american⟩
\ProvidesFile{american.mld}[2004/12/14]
\mtcselectlanguage{english}%
⟨/american⟩
```

8.6 "Arab" language: arab.mld

The titles for the "arab" language are taken from the ArabTEX package [25, 26], which should be used, with the associated fonts.

```
⟨*arab⟩
\ProvidesFile{arab.mld}[1999/03/16]
%% Arabic titles for minitoc.sty
%% Needs arabic fonts (cf. documentation of arabtex)
%% (strings taken from arabtex; to be used with arabtex)
\def\ptctitle{al-mu.htawayAtu}
\def\plftitle{qA'imaTu a.s-.suwari}
\def\plttitle{qA'imaTu al-^gadAwili}
%%
\def\mtctitle{al-mu.htawayAtu}
\def\mlftitle{qA'imaTu a.s-.suwari}
\def\mlttitle{qA'imaTu al-^gadAwili}
%%
\def\stctitle{al-mu.htawayAtu}
\def\slftitle{qA'imaTu a.s-.suwari}
\def\slttitle{qA'imaTu al-^gadAwili}
⟨/arab⟩
```

8.7 "Arabic" language: arabic.mld

"Arabic" is a synomym for "arab", so we just load `arab.mld` (see section 8.6):

```
⟨*arabic⟩
\ProvidesFile{arabic.mld}[2005/02/09]
\mtcselectlanguage{arab}%
⟨/arabic⟩
```

[7] It should be true for the mini-table titles; the languages themselves have some differences.

8.8 “Armenian” language: armenian.mld

The titles for the “armenian” language are taken from the ArmTEX package [16], which should be used, with the associated fonts.

```
⟨*armenian⟩
\ProvidesFile{armenian.mld}[1999/06/28]
%% Armenian titles for minitoc.sty
\def\ptctitle{Bovandakuthyun}
\def\plftitle{Patkerneri cank}
\def\plttitle{Aghyusakneri cank}
%%
\def\mtctitle{Bovandakuthyun}
\def\mlftitle{Patkerneri cank}
\def\mlttitle{Aghyusakneri cank}
%%
\def\stctitle{Bovandakuthyun}
\def\slftitle{Patkerneri cank}
\def\slttitle{Aghyusakneri cank}
⟨/armenian⟩
```

8.9 “Austrian” language: austrian.mld

For the mini-table titles, the “austrian” language is like the “german” language, so we load german.mld (see section 8.48 on page 247):

```
⟨*austrian⟩
\ProvidesFile{austrian.mld}[2004/12/14]
\mtcselectlanguage{german}%
⟨/austrian⟩
```

8.10 “Bahasa” language: bahasa.mld

The titles of the mini-tables for the “bahasa” language[8] are taken from the babel package [10]. Specific fonts are needed.

```
⟨*bahasa⟩
\ProvidesFile{bahasa.mld}[1999/03/16]
%% Bahasa titles for minitoc.sty
\def\ptctitle{Daftar Isi}%
\def\plftitle{Daftar Gambar}%
\def\plttitle{Daftar Tabel}%
%%
```

[8]Spoken in Indonesia and Malaysia, with different pronunciations but the same writing.

```
\def\mtctitle{Daftar Isi}%
\def\mlftitle{Daftar Gambar}%
\def\mlttitle{Daftar Tabel}%
%%
\def\stctitle{Daftar Isi}%
\def\slftitle{Daftar Gambar}%
\def\slttitle{Daftar Tabel}%
⟨/bahasa⟩
```

8.11 “Bangla” language: bangla.mld

The titles for the “bangla” language [9] are taken from the `bangtex` package [36]; they need specific fonts.

```
⟨*bangla⟩
\ProvidesFile{bangla.mld}[2002/03/14]
%% Bangla titles for minitoc.sty
%% Needs specific fonts
\def\ptctitle{suu\*c*i potRo}
\def\plftitle{cho\*b*ir ta\*l*ika}
\def\plttitle{cho\*k*er ta\*l*ika}
%%
\def\mtctitle{suu\*c*i}
\def\mlftitle{cho\*b*ir ta\*l*ika}
\def\mlttitle{cho\*k*er ta\*l*ika}
%%
\def\stctitle{suu\*c*i}
\def\slftitle{cho\*b*ir ta\*l*ika}
\def\slttitle{cho\*k*er ta\*l*ika}
⟨/bangla⟩
```

8.12 “Basque” language: basque.mld

The titles for the “basque” language [10] are taken from the `babel` package [10]. It seems that special fonts are needed.

```
⟨*basque⟩
\ProvidesFile{basque.mld}[1999/12/06]
%% Basque titles for minitoc.sty
%% Needs special fonts
\def\ptctitle{Gaien Aurkibidea}
\def\plftitle{Irudien Zerrenda}
\def\plttitle{Taulen Zerrenda}
%%
```

[9] Spoken in Bangladesh and some parts of India.

[10] Spoken in the Basque country, in the north of Spain and south-west of France.

```
\def\mtctitle{Gaien Aurkibidea}
\def\mlftitle{Irudien Zerrenda}
\def\mlttitle{Taulen Zerrenda}
%%
\def\stctitle{Gaien Aurkibidea}
\def\slftitle{Irudien Zerrenda}
\def\slttitle{Taulen Zerrenda}
⟨/basque⟩
```

8.13 "Bicig" language: `bicig.mld`

The titles for the "bicig" language [11] are taken from the MonTEX package [14, 15]. This language requires specific fonts. See also section 8.83 on page 266.

```
⟨*bicig⟩
\ProvidesFile{bicig.mld}[1999/03/16]
%% needs mongol fonts
%% Mongol (Bicig) titles for minitoc.sty
\def\ptctitle{\bcg{GarciG}}
\def\plftitle{\bcg{zuraG-un zigsaalt}}
\def\plttitle{\bcg{k"usn"agti"in jagsaalt}}
%%
\def\mtctitle{\bcg{GarciG}}
\def\mlftitle{\bcg{zuraG-un zigsaalt}}
\def\mlttitle{\bcg{k"usn"agti"in jagsaalt}}
%%
\def\stctitle{\bcg{GarciG}}
\def\slftitle{\bcg{zuraG-un zigsaalt}}
\def\slttitle{\bcg{k"usn"agti"in jagsaalt}}
⟨/bicig⟩
```

8.14 "Brazil" language: `brazil.mld`

The titles for the "brazil" language [12] are taken from the `babel` package [10]:

```
⟨*brazil⟩
\ProvidesFile{brazil.mld}[1999/03/16]
%% Portugues (brazil) titles for minitoc.sty
\def\ptctitle{Sum\'ario}%
\def\plftitle{Lista de Figuras}%
\def\plttitle{Lista de Tabelas}%
%%
\def\mtctitle{Sum\'ario}%
```

[11] Bicig is a written form of the mongolian language.

[12] It is the portuguese dialect spoken in Brazil. These titles are *different* in Brazil and in Portugal. See section 8.92 on page 270.

```
\def\mlftitle{Lista de Figuras}%
\def\mlttitle{Lista de Tabelas}%
%%
\def\stctitle{Sum\'ario}%
\def\slftitle{Lista de Figuras}%
\def\slttitle{Lista de Tabelas}%
⟨/brazil⟩
```

8.15 “brazilian” language: brazilian.mld

The “brazilian” language is just like “brazil”, so we just load brazil.mld (see section 8.14 on the page before):

```
⟨*brazilian⟩
\ProvidesFile{brazilian.mld}[2005/07/11]
\mtcselectlanguage{brazil}%
⟨/brazilian⟩
```

8.16 “Breton” language: breton.mld

The titles for the “breton” language [13] are taken from the babel package [10]:

```
⟨*breton⟩
\ProvidesFile{breton.mld}[1999/03/16]
%% Breton titles for minitoc.sty
\def\ptctitle{Taolenn}
\def\plftitle{Listenn ar Figurenno\`u}
\def\plttitle{Listenn an taolenno\`u}
%%
\def\mtctitle{Taolenn}
\def\mlftitle{Listenn ar Figurenno\`u}
\def\mlttitle{Listenn an taolenno\`u}
%%
\def\stctitle{Taolenn}
\def\slftitle{Listenn ar Figurenno\`u}
\def\slttitle{Listenn an taolenno\`u}
⟨/breton⟩
```

[13] Spoken as a local celtic dialect in french Brittany.

8.17 "british" language: british.mld

The "british" language is just like "english", so we just load english.mld (see section 8.32 on page 240):

```
⟨*british⟩
\ProvidesFile{british.mld}[2005/07/11]
\mtcselectlanguage{english}%
⟨/british⟩
```

8.18 "Bulgarian" language: bulgarian.mld

The titles for the "bulgarian" language are taken from the babel package [10]; they require specific cyrillic fonts. See also section 8.19 on the next page.

```
⟨*bulgarian⟩
\ProvidesFile{bulgarian.mld}[2001/02/28]
%% Bulgarian titles for minitoc.sty
%% Needs special fonts
\def\ptctitle{%
   {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
\def\plftitle{%
   {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ %
    \cyrn\cyra\ \cyrf\cyri\cyrg\cyru\cyrr\cyri\cyrt\cyre}}%
\def\plttitle{%
   {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ %
    \cyrn\cyra\ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyri\cyrt\cyre}}%
%%
\def\mtctitle{%
   {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
\def\mlftitle{%
   {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ %
    \cyrn\cyra\ \cyrf\cyri\cyrg\cyru\cyrr\cyri\cyrt\cyre}}%
\def\mlttitle{%
   {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ %
    \cyrn\cyra\ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyri\cyrt\cyre}}%
%%
\def\stctitle{%
   {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
\def\slftitle{%
   {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ %
    \cyrn\cyra\ \cyrf\cyri\cyrg\cyru\cyrr\cyri\cyrt\cyre}}%
\def\slttitle{%
   {\cyr\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\ %
    \cyrn\cyra\ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyri\cyrt\cyre}}%
⟨/bulgarian⟩
```

8.19 "Bulgarianb" language: `bulgarianb.mld`

The titles for the "bulgarianb" (upper bulgarian) language are taken from the `babel` package [10]; they require specific cyrillic fonts. See also section 8.18 on the preceding page.

```
⟨*bulgarianb⟩
\ProvidesFile{bulgariand.mld}[2005/03/24]
%% Upper bulgarian titles for minitoc.sty
%% Needs cyrillic fonts for upper bulgarian
\def\ptctitle{%
 {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
\def\plftitle{% Figuri
  {\cyr \CYRF\cyri\cyrg\cyru\cyrr\cyri}}%
\def\plttitle{% Tablici
  {\cyr \CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyri}}%
%%
\def\mtctitle{% Sydyrzhanie
 {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
\def\mlftitle{% Figurite
  {\cyr \CYRF\cyri\cyrg\cyru\cyrr\cyri}}%
\def\mlttitle{% Tablici
  {\cyr \CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyri}}%
%%
\def\stctitle{% Sydyrzhanie
 {\cyr\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
\def\slftitle{% Figuri
  {\cyr \CYRF\cyri\cyrg\cyru\cyrr\cyri}}%
\def\slttitle{% Tablici
  {\cyr \CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyri}}%
⟨/bulgarianb⟩
```

8.20 "Buryat" language: `buryat.mld`

The titles for the "buryat" language [14] are taken from the MonTEX package [14, 15]. This language requires specific fonts. See also section 8.83 on page 266.

```
⟨*buryat⟩
\ProvidesFile{buryat.mld}[1999/03/16]
%% Buryat titles for minitoc.sty
%% Needs special fonts
\def\ptctitle{{\mnr Gar{\sh}ag}}%
\def\plftitle{{\mnr Zuraga"i jagsaalt}}%
\def\plttitle{{\mnr X"usn"ag"at"a"i jagsaalt}}%
%%
\def\mtctitle{{\mnr Gar{\sh}ag}}%
\def\mlftitle{{\mnr Zuraga"i jagsaalt}}%
```

[14] Spoken in some regions of Mongolia, in the Buryat republic, near Lake Baikal.

```
\def\mlttitle{{\mnr X"usn"ag"at"a"i jagsaalt}}%
%%
\def\stctitle{{\mnr Gar{\sh}ag}}%
\def\slftitle{{\mnr Zuraga"i jagsaalt}}%
\def\slttitle{{\mnr X"usn"ag"at"a"i jagsaalt}}%
⟨/buryat⟩
```

8.21 “Canadian” language: `canadian.mld`

The “canadian” language (note the final “i*a*n”) is just the *english* language spoken in Canada. We just load the file `english.mld` (see section 8.32 on page 240):

```
⟨*canadian⟩
\ProvidesFile{canadian.mld}[2004/12/14]
\mtcselectlanguage{english}%
⟨/canadian⟩
```

8.22 “Canadien” language: `canadien.mld`

The “canadien” language (note the final “i*e*n”) is just the *french* language spoken in Canada. We just load the file `french.mld` (see section 8.43 on page 245):

```
⟨*canadien⟩
\ProvidesFile{canadien.mld}[2004/12/14]
\mtcselectlanguage{french}%
⟨/canadien⟩
```

8.23 “Castillan” language: `castillan.mld`

The “castillan” language is more known as “spanish”, but is spoken mainly in Castile, a part of central Spain. We just load the `spanish.mld` file (see section 8.105 on page 278):

```
⟨*castillan⟩
%% Castillan (spanish) titles for minitoc.sty
\ProvidesFile{castillan.mld}[2004/12/14]
\mtcselectlanguage{spanish}%
⟨/castillan⟩
```

8.24 "Castillian" language: castillian.mld

The "castillian" language is more known as "spanish", but is spoken mainly in Castile, a part of central Spain. "Castillian" is the english name for "castillan". We just load the `spanish.mld` file (see section 8.105 on page 278):

```
⟨*castillian⟩
\ProvidesFile{castillian.mld}[2005/07/01]
%% Castillian (spanish) titles for minitoc.sty
\mtcselectlanguage{spanish}%
⟨/castillian⟩
```

8.25 "Catalan" language: catalan.mld

The titles for the "catalan" language [15] are taken from the `babel` package [10]:

```
⟨*catalan⟩
\ProvidesFile{catalan.mld}[1999/03/16]
%% Catalan titles for minitoc.sty
\def\ptctitle{\'Index}%
\def\plftitle{\'Index de figures}%
\def\plttitle{\'Index de taules}%
%%
\def\mtctitle{\'Index}%
\def\mlftitle{Figures}%
\def\mlttitle{Taules}%
%%
\def\stctitle{\'Index}%
\def\slftitle{Figures}%
\def\slttitle{Taules}%
⟨/catalan⟩
```

8.26 "Chinese1" language: chinese1.mld and chinese1.mlo

There are several variants for the chinese language. The "chinese1" language uses titles taken from the `Bg5.cap` file in the CJK system [29]. Special fonts are needed, of course. See also section 8.27 on the following page.

The titles for the "chinese1" language contain characters that cannot be easily generated, hence we load `chinese1.mlo`.

```
⟨*chinese1⟩
\ProvidesFile{chinese1.mld}[2005/01/28]
```

[15] Spoken in Catalunya, the eastern part of Spain, around Barcelona.

```
%% From the file file Bg5.cap of the CJK package
%%   for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%%   created by Werner Lemberg <wl@gnu.org>
%% Version 4.5.2 (28-Mar-2003)
%%
%% Chinese captions
%%
%% character set: Big 5
%% encoding: Big 5
%%
\mtcloadmlo{chinese1}
⟨/chinese1⟩
```

8.27 "Chinese2" language: chinese2.mld and chinese2.mlo

The "chinese2" language uses titles taken from the Bg5.cpx file in the CJK system [29]. Special fonts are needed, of course. See also section 8.26 on the page before.

The titles for the "chinese2" language contain characters that cannot be easily generated, hence we load chinese2.mlo.

```
⟨*chinese2⟩
\ProvidesFile{chinese2.mld}[2005/01/28]
%% From the file Bg5.cpx of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Chinese captions
%%
%% character set: Big 5
%% encoding: Big 5
%%
%% preprocessed
%%
\mtcloadmlo{chinese2}
⟨/chinese2⟩
```

8.28 "Croatian" language: croatian.mld

The titles for the "croatian" language are taken from the babel package [10]:

```
⟨*croatian⟩
\ProvidesFile{croatian.mld}[1999/03/16]
%% Croatian titles for minitoc.sty
```

```
\def\ptctitle{Sadr\v{z}aj}%
\def\plftitle{Slike}%
\def\plttitle{Tablice}%
%%
\def\mtctitle{Sadr\v{z}aj}%
\def\mlftitle{Slike}%
\def\mlttitle{Tablice}%
%%
\def\stctitle{Sadr\v{z}aj}%
\def\slftitle{Slike}%
\def\slttitle{Tablice}%
⟨/croatian⟩
```

8.29 “Czech” language: czech.mld

The titles for the “czech” language are taken from the babel package [10]:

```
⟨*czech⟩
\ProvidesFile{czech.mld}[1999/03/16]
%% Czech titles for minitoc.sty
%% Warning: defines \w as a ring accent
\def\w#1{\accent'27 #1}
\def\ptctitle{Obsah}%
\def\plftitle{Seznam obr\'azk\w{u}}%
\def\plttitle{Seznam tabulek}%
%%
\def\mtctitle{Obsah}%
\def\mlftitle{Seznam obr\'azk\w{u}}%
\def\mlttitle{Seznam tabulek}%
%%
\def\stctitle{Obsah}%
\def\slftitle{Seznam obr\'azk\w{u}}%
\def\slttitle{Seznam tabulek}%
⟨/czech⟩
```

8.30 “Danish” language: danish.mld

The titles for the “danish” language are taken from the babel package [10]:

```
⟨*danish⟩
\ProvidesFile{danish.mld}[1999/03/16]
%% Danish titles for minitoc.sty
\def\ptctitle{Indholdsfortegnelse}%
\def\plftitle{Figurer}%
\def\plttitle{Tabeller}%
%%
```

```
\def\mtctitle{Indholdsfortegnelse}%
\def\mlftitle{Figurer}%
\def\mlttitle{Tabeller}%
%%
\def\stctitle{Indholdsfortegnelse}%
\def\slftitle{Figurer}%
\def\slttitle{Tabeller}%
⟨/danish⟩
```

8.31 “Dutch” language: dutch.mld

The titles for the “dutch” language are taken from the babel package [10]:

```
⟨*dutch⟩
\ProvidesFile{dutch.mld}[1999/03/16]
% Dutch titles for minitoc.sty
\def\ptctitle{Inhoudsopgave}%
\def\plftitle{Lijst van figuren}%
\def\plttitle{Lijst van tabellen}%
%%
\def\mtctitle{Inhoudsopgave}%
\def\mlftitle{Lijst van figuren}%
\def\mlttitle{Lijst van tabellen}%
%%
\def\stctitle{Inhoudsopgave}%
\def\slftitle{Lijst van figuren}%
\def\slttitle{Lijst van tabellen}%
⟨/dutch⟩
```

8.32 “English” language: english.mld

The titles for the “english” language are taken from the babel package [10]. See also sections 8.5 on page 229, 8.17 on page 234, 8.21 on page 236, 8.111 on page 280, and 8.113 on page 281. *It is the default language.*

```
⟨*english⟩
\ProvidesFile{english.mld}[1999/03/16]
%% English titles for minitoc.sty
\def\ptctitle{Table of Contents}
\def\plftitle{List of Figures}
\def\plttitle{List of Tables}
%%
\def\mtctitle{Contents}
\def\mlftitle{Figures}
\def\mlttitle{Tables}
%%
```

```
\def\stctitle{Contents}
\def\slftitle{Figures}
\def\slttitle{Tables}
⟨/english⟩
```

8.33 "Esperant" language: esperant.mld

The titles for the "esperant" (esperanto) language are taken from the babel package [10].

```
⟨*esperant⟩
\ProvidesFile{esperant.mld}[1999/03/16]
%% Esperanto titles for minitoc.sty
\def\ptctitle{Enhavo}%
\def\plftitle{Listo de figuroj}%
\def\plttitle{Listo de tabeloj}%
%%
\def\mtctitle{Enhavo}%
\def\mlftitle{Listo de figuroj}%
\def\mlttitle{Listo de tabeloj}%
%%
\def\stctitle{Enhavo}%
\def\slftitle{Listo de figuroj}%
\def\slttitle{Listo de tabeloj}%
⟨/esperant⟩
```

8.34 "Esperanto" language: esperanto.mld

The "esperanto" and "esperant" languages are synonyms, so we just load the esperant.mld file (see section 8.33):

```
⟨*esperanto⟩
\ProvidesFile{esperanto.mld}[2004/12/14]
\mtcselectlanguage{esperant}%
⟨/esperanto⟩
```

8.35 "Estonian" language: estonian.mld

The titles for the "estonian" language are taken from the babel package [10]:

```
⟨*estonian⟩
\ProvidesFile{estonian.mld}[1999/03/16]
%% Estonian titles for minitoc.sty
```

```
\def\ptctitle{Sisukord}%
\def\plftitle{Joonised}%
\def\plttitle{Tabelid}%
%%
\def\mtctitle{Sisukord}%
\def\mlftitle{Joonised}%
\def\mlttitle{Tabelid}%
%%
\def\stctitle{Sisukord}%
\def\slftitle{Joonised}%
\def\slttitle{Tabelid}%
⟨/estonian⟩
```

8.36 "Ethiopia" language: ethiopia.mld

The titles for the "ethiopia" language are taken from the ethiop package [6]. Specific fonts are needed.

```
⟨*ethiopia⟩
\ProvidesFile{ethiopia.mld}[1999/03/16]
%% Needs special fonts
\def\ptctitle{yezate}%
\def\plftitle{%
    ya\eth@doaltchar{85}‘elo\eth@doaltchar{109}
    mAwe\eth@doaltchar{187}}%
\def\plttitle{%
    yasane\eth@doaltchar{176}ra\eth@doaltchar{149}
    mAwe\eth@doaltchar{187}}%
%%
\def\mtctitle{yezate}%
\def\mlftitle{%
    ya\eth@doaltchar{85}‘elo\eth@doaltchar{109}
    mAwe\eth@doaltchar{187}}%
\def\mlttitle{%
    yasane\eth@doaltchar{176}ra\eth@doaltchar{149}
    mAwe\eth@doaltchar{187}}%
%%
\def\stctitle{yezate}%
\def\slftitle{%
    ya\eth@doaltchar{85}‘elo\eth@doaltchar{109}
    mAwe\eth@doaltchar{187}}%
\def\slttitle{%
    yasane\eth@doaltchar{176}ra\eth@doaltchar{149}
    mAwe\eth@doaltchar{187}}%
⟨/ethiopia⟩
```

8.37 "Ethiopian" language: ethiopian.mld

The "ethiopian" language is just a synonym for the "ethiopia" language, so we just load the ethiopia.mld file (see section 8.36 on the preceding page):

```
⟨*ethiopian⟩
\ProvidesFile{ethiopian.mld}[2004/12/14]
%% Needs ethopian special fonts
\mtcselectlanguage{ethiopia}%
⟨/ethiopian⟩
```

8.38 "Farsi1" language: farsi1.mld and farsi1.mlo

There are several variants for the farsi language, spoken in Iran. The "farsi1" language uses titles taken from the farsi.sty file in the FarsiTEX system (http://www.farsitex.org), by Dr Mohammad Ghodsi, Roozbeh Pournader, Hassan Abolhassani, and others. Special fonts are needed, of course. See also section 8.39.

The titles for the "farsi1" language contain characters that cannot be easily generated, hence we load farsi1.mlo.

```
⟨*farsi1⟩
\ProvidesFile{farsi1.mld}[2005/09/13]
% From farsi.sty of the FarsiTeX project
% by Dr Mohammad Ghodsi,
% Roozbeh Pournader (roozbeh@sharif.edu),
% Hassan Abolhassani, and others.
% http://www.farsitex.org
\mtcloadmlo{farsi1}
⟨/farsi1⟩
```

8.39 "Farsi2" language: farsi2.mld and farsi2.mlo

There are several variants for the farsi language, spoken in Iran. The "farsi2" language uses titles taken from the farsi.sty file in the FarsiTEX system (http://www.farsitex.org), by Dr Mohammad Ghodsi, Roozbeh Pournader, Hassan Abolhassani, and others. Special fonts are needed, of course. See also section 8.38.

The titles for the "farsi2" language contain characters that cannot be easily generated, hence we load farsi2.mlo.

```
⟨*farsi2⟩
\ProvidesFile{farsi2.mld}[2005/09/13]
% From farsi.sty of the FarsiTeX project
```

```
% by Dr Mohammad Ghodsi,
% Roozbeh Pournader (roozbeh@sharif.edu),
% Hassan Abolhassani, and others.
% http://www.farsitex.org
\mtcloadmlo{farsi2}
⟨/farsi2⟩
```

8.40 "Finnish" language: `finnish.mld`

The titles for the "finnish" language are taken from the `babel` package [10]. See also section 8.41.

```
⟨*finnish⟩
\ProvidesFile{finnish.mld}[1999/03/16]
%% Finnish titles for minitoc.sty
\def\ptctitle{Sis\"alt\"o}%   /* Could be "Sis\"allys" as well */
\def\plftitle{Kuvat}%
\def\plttitle{Taulukot}%
%%
\def\mtctitle{Sis\"alt\"o}%   /* Could be "Sis\"allys" as well */
\def\mlftitle{Kuvat}%
\def\mlttitle{Taulukot}%
%%
\def\stctitle{Sis\"alt\"o}%   /* Could be "Sis\"allys" as well */
\def\slftitle{Kuvat}%
\def\slttitle{Taulukot}%
⟨/finnish⟩
```

8.41 "Finnish2" language: `finnish2.mld`

The titles for the "finnish2" language are taken from a variant proposed in the `babel` package [10]. See also section 8.40.

```
⟨*finnish2⟩
\ProvidesFile{finnish2.mld}[2005/04/06]
%% Finnish titles for minitoc.sty
\def\ptctitle{Sis\"allys}%
\def\plftitle{Kuvat}%
\def\plttitle{Taulukot}%
%%
\def\mtctitle{Sis\"allys}%
\def\mlftitle{Kuvat}%
\def\mlttitle{Taulukot}%
%%
\def\stctitle{Sis\"allys}%
\def\slftitle{Kuvat}%
```

```
\def\slttitle{Taulukot}%
⟨/finnish2⟩
```

8.42 “Francais” language: francais.mld

The “francais”[16] language is a synonym for the “french” language, so we load the file french.mld (see section 8.43):

```
⟨*francais⟩
\ProvidesFile{francais.mld}[2004/12/14]
\mtcselectlanguage{french}%
⟨/francais⟩
```

8.43 “French” language: french.mld

The titles for the “french” language are taken from the babel package [10]. See also sections 8.1 on page 227, 8.2 on page 228, 8.22 on page 236, 8.42, 8.44, 8.45 on the next page, and 8.46 on the following page.

```
⟨*french⟩
\ProvidesFile{french.mld}[2005/06/16]
%%
\def\ptctitle{Table des Mati\‘eres}
\def\plftitle{Liste des Figures}
\def\plttitle{Liste des Tableaux}
%%
%% French titles for minitoc.sty
\def\mtctitle{Sommaire}
\def\mlftitle{Figures}
\def\mlttitle{Tableaux}
%%
\def\stctitle{Sommaire}
\def\slftitle{Figures}
\def\slttitle{Tableaux}
⟨/french⟩
```

8.44 “Frenchb” language: frenchb.mld

The “frenchb” language is a synonym for the “french” language, so we load the french.mld file. See section 8.43.

[16]The right spelling is “français”, but we do not dare using a cedilla in a file name.

```
⟨*frenchb⟩
\ProvidesFile{frenchb.mld}[2003/02/11]
\mtcselectlanguage{french}%
⟨/frenchb⟩
```

8.45 “Frenchle” language: `frenchle.mld`

The “frenchle” language is a synonym for the “french” language, so we load the `french.mld` file. See section 8.43 on the preceding page.

```
⟨*frenchle⟩
\ProvidesFile{frenchle.mld}[2003/02/20]
\mtcselectlanguage{french}%
⟨/frenchle⟩
```

8.46 “Frenchpro” language: `frenchpro.mld`

The “frenchpro” language is a synonym for the “french” language, so we load the `french.mld` file. See section 8.43 on the page before.

```
⟨*frenchpro⟩
\ProvidesFile{frenchpro.mld}[2003/02/20]
\mtcselectlanguage{french}%
⟨/frenchpro⟩
```

8.47 “Galician” language: `galician.mld`

The titles for the “galician” language [17] are taken from the `babel` package [10]:

```
⟨*galician⟩
\ProvidesFile{galician.mld}[1999/03/16]
%% Galician titles for minitoc.sty
%%
\def\ptctitle{\'Indice Xeral}%
\def\plftitle{\'Indice de Figuras}%
\def\plttitle{\'Indice de T\'aboas}%
%%
\def\mtctitle{\'Indice Xeral}%
\def\mlftitle{\'Indice de Figuras}%
\def\mlttitle{\'Indice de T\'aboas}%
%%
```

[17] Spoken in Galice, in the north-west part of Spain, around Santiago de Compostela.

```
\def\stctitle{\'Indice Xeral}%
\def\slftitle{\'Indice de Figuras}%
\def\slttitle{\'Indice de T\'aboas}%
⟨/galician⟩
```

8.48 "German" language: german.mld

The titles for the "german" language are taken from the babel package [10]. See also the section 8.9 on page 230.

```
⟨*german⟩
\ProvidesFile{german.mld}[1999/03/16]
%% German titles for minitoc.sty
\def\ptctitle{Inhaltsangabe}
\def\plftitle{Figuren}
\def\plttitle{Tabellen}
%%
\def\mtctitle{Inhaltsangabe}
\def\mlftitle{Figuren}
\def\mlttitle{Tabellen}
%%
\def\stctitle{Inhaltsangabe}
\def\slftitle{Figuren}
\def\slttitle{Tabellen}
⟨/german⟩
```

8.49 "Germanb" language: germanb.mld

The "germanb" language is a variant for the "german" language. The titles are taken from the babel package [10]:

```
⟨*germanb⟩
\ProvidesFile{germanb.mld}[1999/03/16]
%% German titles (variant) for minitoc.sty
\def\ptctitle{Inhaltsverzeichnis}%    % oder nur: Inhalt
\def\plftitle{Abbildungsverzeichnis}
\def\plttitle{Tabellenverzeichnis}
%%
\def\mtctitle{Inhaltsverzeichnis}%    % oder nur: Inhalt
\def\mlftitle{Abbildungsverzeichnis}
\def\mlttitle{Tabellenverzeichnis}
%%
\def\stctitle{Inhalt}%    % oder nur: Inhalt
\def\slftitle{Abbildungen}
\def\slttitle{Tabellen}
⟨/germanb⟩
```

8.50 “Greek” language: greek.mld

The titles for the “greek” language (modern greek) are taken from the babel package [10]. Greek fonts are required.

```
⟨*greek⟩
\ProvidesFile{greek.mld}[1999/03/16]
%% greek.mld
%% Needs greek fonts.
\def\ptctitle{Perieq'omena}
\def\plftitle{Kat'alogoc Sqhm'atwn}
\def\plttitle{Kat'alogoc Pin'akwn}
%%
\def\mtctitle{Perieq'omena}
\def\mlftitle{Kat'alogoc Sqhm'atwn}
\def\mlttitle{Kat'alogoc Pin'akwn}
%%
\def\stctitle{Perieq'omena}
\def\slftitle{Kat'alogoc Sqhm'atwn}
\def\slttitle{Kat'alogoc Pin'akwn}
⟨/greek⟩
```

8.51 “Greek-mono” language: greek-mono.mld

The titles for the “greek-mono” language [18] are taken from the ANTOMEGA project [24]:

```
⟨*greek – mono⟩
\ProvidesFile{greek-mono.mld}[2005/02/08]
%% from omega-greek.ldf (Antomega project)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
\def\ptctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^03cc^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\plftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^03ac^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^03ac^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\plftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^03ac^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^03ac^^^^03ba^^^^03c9^^^^03bd}}%
%%
\def\mtctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^03cc^^^^03bc%
```

[18] Monotonic greek, from a recent (1982) but strongly contested – and contestable – reform of the greek language.

```
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\mlftitle%{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^03ac^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^03ac^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\mlftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^03ac^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^03ac^^^^03ba^^^^03c9^^^^03bd}}%
%%
\def\stctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^03cc^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\slftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^03ac^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^03ac^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\slftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^03ac^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^03ac^^^^03ba^^^^03c9^^^^03bd}}%
⟨/greek – mono⟩
```

8.52 "Greek-polydemo" language: `greek-polydemo.mld`

The titles for the "greek-polydemo" language [19] are taken from the ANTOMEGA project [24]:

```
⟨*greek – polydemo⟩
\ProvidesFile{greek-polydemo.mld}[2005/02/08]
%% from omega-greek.ldf (Antomega project)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
\def\ptctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^1f79^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\plftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^1f71^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\plttitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^1f71^^^^03ba^^^^03c9^^^^03bd}}%
%%
\def\mtctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^1f79^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\mlftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
```

[19] Polytonic demotic (popular) greek, for classical greek.

```
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^1f71^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\mlttitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^1f71^^^^03ba^^^^03c9^^^^03bd}}%
%%
\def\stctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^1f79^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\slftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^1f71^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\slttitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^1f71^^^^03ba^^^^03c9^^^^03bd}}%
⟨/greek – polydemo⟩
```

8.53 "Greek-polykatha" language: greek-polykatha.mld

The titles for the "greek-polykatha" language[20] are taken from the ANTOMEGA project [24]:

```
⟨*greek – polykatha⟩
\ProvidesFile{greek-polykatha.mld}[2005/02/08]
%% from omega-greek.ldf (Antomega project)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
\def\ptctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^1f79^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\plftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^1f71^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\plttitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^1f71^^^^03ba^^^^03c9^^^^03bd}}%
%%
\def\mtctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^1f79^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\mlftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
```

[20]Polytonic greek, « kathaverousa » (purified) style, a form of the Greek language created during the early 19th century by Adamantios KORAIS, to purify the language from the Byzantine and non-greek vocabulary. It has now been obsoleted by the demotic (popular) greek, but it has left a very noticeable trace in the modern Greek language.

```
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^1f71^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\mlttitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^1f71^^^^03ba^^^^03c9^^^^03bd}}%
%%
\def\stctitle{\localgreek%
{^^^^03a0^^^^03b5^^^^03c1^^^^03b9^^^^03b5^^^^03c7^^^^1f79^^^^03bc%
 ^^^^03b5^^^^03bd^^^^03b1}}%
\def\slftitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c3^^^^03c7^^^^03b7^^^^03bc^^^^1f71^^^^03c4^^^^03c9%
 ^^^^03bd}}%
\def\slttitle{\localgreek%
{^^^^039a^^^^03b1^^^^03c4^^^^1f71^^^^03bb^^^^03bf^^^^03b3^^^^03bf%
 ^^^^03c2 ^^^^03c0^^^^03b9^^^^03bd^^^^1f71^^^^03ba^^^^03c9^^^^03bd}}%
⟨/greek – polykatha⟩
```

8.54 “Guarani” language: guarani.mld

The “guarani” language is the main language spoken in Paraguay. Very often, a mixture of Guarani and Spanish, known as Jopara, is spoken. A special input encoding (win-gn.def) is needed. The titles are taken from the guarani.ldf file by Javier Bezos [7]. A special input encoding (win-gn.def) is needed. These files are available on the CTAN archives.

```
⟨*guarani⟩
\ProvidesFile{guarani.mld}[2005/08/26]
%% Guarani titles for minitoc.sty
%% from guarani.ldf by Javier Bezos.
%% Input encoding win-gn.def is needed.
%%
\def\ptctitle{\'Indice general}%
\def\plftitle{\'Indice de figuras}%
\def\plttitle{\'Indice de cuadros}%
%%
\def\mtctitle{\'Indice general}%
\def\mlftitle{\'Indice de figuras}%
\def\mlttitle{\'Indice de cuadros}%
%%
\def\stctitle{\'Indice general}%
\def\slftitle{\'Indice de figuras}%
\def\slttitle{\'Indice de cuadros}%
⟨/guarani⟩
```

8.55 "Hangul1" language: hangul1.mld and hangul1.mlo

The titles for the "hangul1" language (korean in hangul script, first variant) are taken from the file hangul.cap of the CJK system [29]. Special fonts are needed, of course. See also sections 8.56, 8.57 on the following page, 8.58 on the next page, 8.59 on page 254, and 8.60 on page 255.

The titles for the "hangul1" language contain characters that cannot be easily generated, hence we load hangul1.mlo.

```
⟨*hangul1⟩
\ProvidesFile{hangul1.mld}[2005/01/28]
%% From the file hangul.cap of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Hangul captions
%%
%% character set: KS X 1001:1992 (=KS C 5601-1992)
%% encoding: EUC (=Wansung)
%%
\mtcloadmlo{hangul1}
⟨/hangul1⟩
```

8.56 "Hangul2" language: hangul2.mld and hangul2.mlo

The titles for the "hangul2" language (korean in hangul script, second variant) are taken from the file hangul.cpx of the CJK system [29]. Special fonts are needed, of course. See also sections 8.55, 8.57 on the next page, 8.58 on the following page, 8.59 on page 254, and 8.60 on page 255.

The titles for the "hangul2" language contain characters that cannot be easily generated, hence we load hangul2.mlo.

```
⟨*hangul2⟩
\ProvidesFile{hangul2.mld}[2005/01/28]
%% From the file hangul.cpx of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Hangul captions
%%
%% character set: KS X 1001:1992 (=KS C 5601-1992)
%% encoding: EUC (=Wansung)
```

```
%%
%% preprocessed
%%
\mtcloadmlo{hangul2}
⟨/hangul2⟩
```

8.57 “Hangul3” language: hangul3.mld and hangul3.mlo

The titles for the “hangul3” language (korean in hangul script, third variant) are taken from the file `hangul2.cap` of the CJK system [29]. Special fonts are needed, of course. See also sections 8.55 on the page before, 8.56 on the preceding page, 8.58, 8.59 on the following page, and 8.60 on page 255.

The titles for the “hangul3” language contain characters that cannot be easily generated, hence we load `hangul3.mlo`.

```
⟨*hangul3⟩
\ProvidesFile{hangul3.mld}[2005/01/28]
%% From the file hangul2.cap of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Hangul captions set 2
%%
%% character set: KS X 1001:1992 (=KS C 5601-1992)
%% encoding: EUC (=Wansung)
%%
\mtcloadmlo{hangul3}
⟨/hangul3⟩
```

8.58 “Hangul4” language: hangul4.mld and hangul4.mlo

The titles for the “hangul4” language (korean in hangul script, fourth variant) are taken from the file `hangul2.cpx` of the CJK system [29]. Special fonts are needed, of course. See also sections 8.55 on the preceding page, 8.56 on the page before, 8.57, 8.59 on the next page, and 8.60 on page 255.

The titles for the “hangul4” language contain characters that cannot be easily generated, hence we load `hangul4.mlo`.

```
⟨*hangul4⟩
\ProvidesFile{hangul4.mld}[2005/01/28]
%% From the file hangul2.cpx of the CJK package
```

```
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Hangul captions set 2
%%
%% character set: KS X 1001:1992 (=KS C 5601-1992)
%% encoding: EUC (=Wansung)
%%
%% preprocessed
%%
\mtcloadmlo{hangul4}
⟨/hangul4⟩
```

8.59 "Hanja1" language: hanja1.mld and hanja1.mlo

The titles for the "hanja1" language (korean in the old script hanja, first variant) are taken from the file `hanja.cpx` of the CJK system [29]. Special fonts are needed, of course. See also sections 8.55 on page 252, 8.56 on page 252, 8.57 on the page before, 8.58 on the preceding page, and 8.60 on the next page.

The titles for the "hanja1" language contain characters that cannot be easily generated, hence we load `hanja1.mlo`.

```
⟨*hanja1⟩
\ProvidesFile{hanja1.mld}[2005/01/28]
%% From the file hanja.cpx of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Hanja captions
%%
%% character set: KS X 1001:1992 (=KS C 5601-1992)
%% encoding: EUC (=Wansung)
%%
%% preprocessed
%%
\mtcloadmlo{hanja1}
⟨/hanja1⟩
```

8.60 "Hanja2" language: hanja2.mld and hanja2.mlo

The titles for the "hanja2" language (Korean in the old script hanja, second variant) are taken from the file hanja.cap of the CJK system [29]. Special fonts are needed, of course. See also sections 8.55 on page 252, 8.56 on page 252, 8.57 on page 253, 8.58 on page 253, and 8.59 on the preceding page.

The titles for the "hanja2" language contain characters that cannot be easily generated, hence we load hanja2.mlo.

```
⟨*hanja2⟩
\ProvidesFile{hanja2.mld}[2005/01/28]
%% From the file hanja.cap of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <a7971428@unet.univie.ac.at>
%%
%% Version 4.1.3 (20-Jun-1997)
%%
%% Hanja captions
%%
%% character set: KS X 1001:1992 (=KS C 5601-1992)
%% encoding: EUC (=Wansung)
%%
\mtcloadmlo{hanja2}
⟨/hanja2⟩
```

8.61 "Hebrew" language: hebrew.mld

The titles for the "hebrew" language are taken from the ArabTeX package [25, 26], which should be used, with the associated fonts.

```
⟨*hebrew⟩
\ProvidesFile{hebrew.mld}[2001/02/28]
%% Hebrew titles for minitoc.sty
%% Need hebrew fonts (see arabtex documentation)
\def\ptctitle{\tav\vav\kaf\finalnun\
              \ayin\nun\yod\nun\yod\finalmem}%
\def\plftitle{\resh\shin\yod\mem\tav\
              \alef\yod\vav\resh\yod\finalmem}%
\def\plttitle{\resh\shin\yod\mem\tav\
              \tet\bet\lamed\alef\vav\tav}%
%%
\def\mtctitle{\tav\vav\kaf\finalnun\
              \ayin\nun\yod\nun\yod\finalmem}%
\def\mlftitle{\resh\shin\yod\mem\tav\
              \alef\yod\vav\resh\yod\finalmem}%
\def\mlttitle{\resh\shin\yod\mem\tav\
              \tet\bet\lamed\alef\vav\tav}%
```

```
%%
\def\stctitle{\tav\vav\kaf\finalnun\
              \ayin\nun\yod\nun\yod\finalmem}%
\def\slftitle{\resh\shin\yod\mem\tav\
              \alef\yod\vav\resh\yod\finalmem}%
\def\slttitle{\resh\shin\yod\mem\tav\
              \tet\bet\lamed\alef\vav\tav}%
⟨/hebrew⟩
```

8.62 “Hungarian” language: hungarian.mld

The “hungarian” language is a synonym of the “magyar” language, so we load `magyar.mld`. See section 8.78 on page 263.

```
⟨*hungarian⟩
\ProvidesFile{hungarian.mld}[2004/12/14]
\mtcselectlanguage{magyar}%
⟨/hungarian⟩
```

8.63 “Icelandic” language: icelandic.mld

The titles for the “icelandic” language are taken from the `babel` package [10]:

```
⟨*icelandic⟩
\ProvidesFile{icelandic.mld}[2001/02/28]
%% Icelandic titles for minitoc.sty
%% need inputenc with 8-bits encoding
\def\ptctitle{Efnisyfirlit}%
\def\plftitle{Myndaskrá}%
\def\plttitle{Töfluskrá}%
%%
\def\mtctitle{Efnisyfirlit}%
\def\mlftitle{Myndaskrá}%
\def\mlttitle{Töfluskrá}%
%%
\def\stctitle{Efnisyfirlit}%
\def\slftitle{Myndaskrá}%
\def\slttitle{Töfluskrá}%
⟨/icelandic⟩
```

8.64 "Interlingua" language: interlingua.mld

The titles for the "interlingua" language [21] are taken from the babel package [10]:

```
⟨*interlingua⟩
\ProvidesFile{interlingua.mld}[2001/07/04]
\def\ptctitle{Contento}
\def\plftitle{Lista de Figuras}
\def\plttitle{Lista de Tabellas}
%%
\def\mtctitle{Contento}
\def\mlftitle{Figuras}
\def\mlttitle{Tabellas}
%%
\def\stctitle{Contento}
\def\slftitle{Figuras}
\def\slttitle{Tabellas}
⟨/interlingua⟩
```

8.65 "Irish" language: irish.mld

The titles for the "irish" language are taken from the babel package [10]:

```
⟨*irish⟩
\ProvidesFile{irish.mld}[1999/03/16]
%% Irish titles for minitoc.sty
\def\ptctitle{Cl\'ar\'Abhair}
\def\plftitle{L\'ear\'aid\'{\i}}
\def\plttitle{T\'abla\'{\i}}
%%
\def\mtctitle{Cl\'ar\'Abhair}
\def\mlftitle{L\'ear\'aid\'{\i}}
\def\mlttitle{T\'abla\'{\i}}
%%
\def\stctitle{Cl\'ar\'Abhair}
\def\slftitle{L\'ear\'aid\'{\i}}
\def\slttitle{T\'abla\'{\i}}
⟨/irish⟩
```

8.66 "Italian" language: italian.mld

The titles for the "italian" language are taken from the babel package [10]:

[21] A tentative for an universal language.

```
⟨*italian⟩
\ProvidesFile{italian.mld}[1999/03/16]
%% Italian titles for minitoc.sty
\def\ptctitle{Contenuto}%
\def\plftitle{Elenco delle figure}%
\def\plttitle{Elenco delle tabelle}%
%%
\def\mtctitle{Contenuto}%
\def\mlftitle{Elenco delle figure}%
\def\mlttitle{Elenco delle tabelle}%
%%
\def\stctitle{Contenuto}%
\def\slftitle{Elenco delle figure}%
\def\slttitle{Elenco delle tabelle}%
⟨/italian⟩
```

8.67 “Japanese” language: `japanese.mld` and `japanese.mlo`

They are several variants for the japanese titles. The titles for a first variant of the “japanese” language have been found (by a Google search) on the Web site of Professor Toshiki Kumazawa [22]. See also sections 8.68, 8.69 on the next page, 8.70 on page 260, and 8.71 on page 260.

The titles for the “japanese” language contain characters that cannot be easily generated, hence we load `japanese.mlo`.

```
⟨*japanese⟩
\ProvidesFile{japanese.mld}[1999/03/16]
%% Japanese titles for minitoc.sty
%% Needs japanese fonts (CJK) and special input encoding.
%% From Kumazawa Toshiki
%% kumazawa@biwako.shiga-u.ac.jp
%% http://www.biwako.shiga-u.ac.jp/sensei/kumazawa/tex/minitoc.html
\mtcloadmlo{japanese}
⟨/japanese⟩
```

8.68 “Japanese2” language: `japanese2.mld` and `japanese2.mlo`

The titles for the “japanese2” language (japanese, second variant) are taken from file `JIS.cap` of the CJK system [29]. Special fonts are needed, of course. See also sections 8.67, 8.69 on the next page, 8.70 on page 260, and 8.71 on page 260.

[22] `http:www.biwako.shiga-u.ac.jp/sensei/kumazawa/tex/minitoc.html`

The titles for the “japanese2” language contain characters that cannot be easily generated, hence we load `japanese2.mlo`.

```
⟨*japanese2⟩
\ProvidesFile{japanese2.mld}[2005/01/28]
%% From the file JIS.cap of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Japanese captions
%%
%% character set: JIS X 0208:1997 (or JIS X 0208-1990)
%% encoding: EUC
%%
\mtcloadmlo{japanese2}
⟨/japanese2⟩
```

8.69 “Japanese3” language: `japanese3.mld` and `japanese3.mlo`

The titles for the “japanese3” language (japanese, third variant) are taken from file `JIS.cpx` of the CJK system [29]. Special fonts are needed, of course. See also sections 8.67 on the preceding page, 8.68 on the page before, 8.70 on the following page, and 8.71 on the next page.

The titles for the “japanese3” language contain characters that cannot be easily generated, hence we load `japanese3.mlo`.

```
⟨*japanese3⟩
\ProvidesFile{japanese3.mld}[2005/01/28]
%% From the file JIS.cpx of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Japanese captions
%%
%% character set: JIS X 0208:1997 (or JIS X 0208-1990)
%% encoding: EUC
%%
%% preprocessed
%%
\mtcloadmlo{japanese3}
⟨/japanese3⟩
```

8.70 “Japanese4” language: japanese4.mld and japanese4.mlo

The titles for the “japanese4” language (japanese, fourth version) are taken from file SJIS.cap of the CJK system [29]. Special fonts are needed, of course. See also sections 8.67 on page 258, 8.68 on page 258, 8.69 on the preceding page, and 8.71.

The titles for the “japanese4” language contain characters that cannot be easily generated, hence we load japanese4.mlo.

```
⟨*japanese4⟩
\ProvidesFile{japanese4.mld}[2005/01/28]
% From the file SJIS.cap of the CJK package
% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
% created by Werner Lemberg <wl@gnu.org>
%
% Version 4.5.2 (28-Mar-2003)
%
% Japanese captions
%
% character set: JIS X 0208:1997 (or JIS X 0208-1990)
% encoding: SJIS
%
\mtcloadmlo{japanese4}
⟨/japanese4⟩
```

8.71 “Japanese5” language: japanese5.mld and japanese.mlo

The titles for the “japanese5” (japanese, fifth variant) language are taken from file SJIS.cpx of the CJK system [29]. Special fonts are needed, of course. See also sections 8.67 on page 258, 8.68 on page 258, 8.69 on the preceding page, and 8.70.

The titles for the “japanese5” language contain characters that cannot be easily generated, hence we load japanese5.mlo.

```
⟨*japanese5⟩
\ProvidesFile{japanese5.mld}[[2005/01/28]
%% From the file SJIS.cpx of the CJK package
%% for using Asian logographs (Chinese/Japanese/Korean) with LaTeX2e
%% created by Werner Lemberg <wl@gnu.org>
%%
%% Version 4.5.2 (28-Mar-2003)
%%
%% Japanese captions
%%
%% character set: JIS X 0208:1997 (or JIS X 0208-1990)
%% encoding: SJIS
%%
%% preprocessed
```

```
%%
\mtcloadmlo{japanese5}
⟨/japanese5⟩
```

8.72 "Latin" language: latin.mld

The titles for the "latin" language are taken from the babel package [10]. See also section 8.73.

```
⟨*latin⟩
\ProvidesFile{latin.mld}[2001/02/28]
%% Latin titles for minitoc.sty
\def\ptctitle{Index}%
\def\plftitle{Conspectus descriptionum}%
\def\plttitle{Conspectus tabularum}%
%%
\def\mtctitle{Index}%
\def\mlftitle{Conspectus descriptionum}%
\def\mlttitle{Conspectus tabularum}%
%%
\def\stctitle{Index}%
\def\slftitle{Conspectus descriptionum}%
\def\slttitle{Conspectus tabularum}%
⟨/latin⟩
```

8.73 "Latin2" language: latin2.mld

The titles for the "latin2" language (latin, abbreviated variant) are taken from the babel package [10], but abbreviated. See also section 8.72.

```
⟨*latin2⟩
\ProvidesFile{latin2.mld}[2005/04/06]
%% Latin titles (short) for minitoc.sty
\def\ptctitle{Index}%
\def\plftitle{Conspectus descriptionum}%
\def\plttitle{Conspectus tabularum}%
%%
\def\mtctitle{Index}%
\def\mlftitle{Descriptiones}%
\def\mlttitle{Tabulae}%
%%
\def\stctitle{Index}%
\def\slftitle{Descriptiones}%
\def\slttitle{Tabulae}%
⟨/latin2⟩
```

8.74 “Latvian” language: latvian.mld

The titles for the “latvian” language [23] are taken from the ANTOMEGA project [24]. See also section 8.75:

```
⟨*latvian⟩
\ProvidesFile{latvian.mld}[2005/02/08]
%% from latvian.ldf (Antomega project)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
\def\ptctitle{\locallatvian{Saturs}}%
\def\plftitle{\locallatvian{Att^^^^0113lu saraksts}}%
\def\plttitle{\locallatvian{Tabulu saraksts}}%
%%
\def\mtctitle{\locallatvian{Saturs}}%
\def\mlftitle{\locallatvian{Att^^^^0113lu saraksts}}%
\def\mlttitle{\locallatvian{Tabulu saraksts}}%
%%
\def\stctitle{\locallatvian{Saturs}}%
\def\slftitle{\locallatvian{Att^^^^0113lu saraksts}}%
\def\slttitle{\locallatvian{Tabulu saraksts}}%
⟨/latvian⟩
```

8.75 “Letton” language: letton.mld

The “letton” language is a synonym for the “latvian” language, so we just load latvian.mld. See section 8.74.

```
⟨*letton⟩
\ProvidesFile{letton.mld}[2005/02/08]
\mtcselectlanguage{latvian}%
⟨/letton⟩
```

8.76 “Lithuanian” language: lithuanian.mld

The titles for the “lithuanian” language are taken from the babel package [10]:

```
⟨*lithuanian⟩
\ProvidesFile{lithuanian.mld}[1999/07/29]
%% Lithuanian titles for minitoc.sty
\def\ptctitle{Turinys}
```

[23] “Latvian” is the original name for “letton”.

```
\def\plftitle{Paveiksl\protect\c u s\protect\c ara\protect\v sas}
\def\plttitle{Lentel\protect\.es}
%%
\def\mtctitle{Turinys}
\def\mlftitle{Paveiksl\protect\c u s\protect\c ara\protect\v sas}
\def\mlttitle{Lentel\protect\.es}
%%
\def\stctitle{Turinys}
\def\slftitle{Paveiksl\protect\c u s\protect\c ara\protect\v sas}
\def\slttitle{Lentel\protect\.es}
⟨/lithuanian⟩
```

8.77 "Lsorbian" language: lsorbian.mld

The titles for the "lsorbian" language [24] are taken from the babel package [10]. See also section 8.114 on page 282.

```
⟨*lsorbian⟩
\ProvidesFile{lsorbian.mld}[1999/03/16]
%% Lower sorbian titles for minitoc.sty
\def\ptctitle{Wop\'simje\'se}
\def\plftitle{Zapiswobrazow}
\def\plttitle{Zapistabulkow}
%%
\def\mtctitle{Wop\'simje\'se}
\def\mlftitle{Zapiswobrazow}
\def\mlttitle{Zapistabulkow}
%%
\def\stctitle{Wop\'simje\'se}
\def\slftitle{Zapiswobrazow}
\def\slttitle{Zapistabulkow}
⟨/lsorbian⟩
```

8.78 "Magyar" language: magyar.mld

The titles for the "magyar" language are taken from the babel package [10]. A synonym of "magyar" is "hungarian" (see section 8.62 on page 256). See also section 8.79 on the following page.

```
⟨*magyar⟩
\ProvidesFile{magyar.mld}[1999/03/16]
% Magyar titles for minitoc.sty
```

[24] Lower sorbian. Sorbian, or wendisch, is a member of the west slavic subgroup of indo-european languages spoken in Lower Lusatia in the german *länder* of Saxony and Brandenburg. The Sorbs are descendents of the Wends, the german name for the slavic tribes who occupied the area between the Elbe and Saale rivers in the west and the Odra (Oder) river in the east during the medieval period (VIth century).

```
\def\ptctitle{Tratalom}%
\def\plftitle{\'Abr\'ak}%
\def\plttitle{T\'abl\'azatok}%
%%
\def\mtctitle{Tratalom}%
\def\mlftitle{\'Abr\'ak}%
\def\mlttitle{T\'abl\'azatok}%
%%
\def\stctitle{Tratalom}%
\def\slftitle{\'Abr\'ak}%
\def\slttitle{T\'abl\'azatok}%
⟨/magyar⟩
```

8.79 “Magyar2” language: magyar2.mld

The titles for the “magyar2” language are taken from a variant proposed in the babel package [10]. See also section 8.78 on the page before.

```
⟨*magyar2⟩
\ProvidesFile{magyar.mld}[2005/04/06]
%% Magyar titles for minitoc.sty (variant)
\def\ptctitle{Tratalom}%
\def\plftitle{\'Abr\'ak list\'aja}%
\def\plttitle{T\'abl\'azatok list\'aja}%
%%
\def\mtctitle{Tratalom}%
\def\mlftitle{\'Abr\'ak list\'aja}%
\def\mlttitle{T\'abl\'azatok list\'aja}%
%%
\def\stctitle{Tratalom}%
\def\slftitle{\'Abr\'ak list\'aja}%
\def\slttitle{T\'abl\'azatok list\'aja}%
⟨/magyar2⟩
```

8.80 “Malayalam-keli” language: malayalam-keli.mld

The titles for the “malayalam” language [25], with the “Keli” fonts, are taken from the malayalam package [1]. This language requires specific fonts. See also sections 8.81 on the following page and 8.82 on the next page.

```
⟨*malayalam – keli⟩
\ProvidesFile{malayalam-keli.mld}[2005/06/07]
%
```

[25] The Malayalam language is spoken from the western coast of Malabar to the extreme southern India, mainly in the Kerala state. It is one of the dravidian languages strongly bound to the Tamil language. The alphabet and the script are dated from the 8th or 9th centuries.

```
% Malayalam: Keli fonts
%
\def\ptctitle{\mm \X{\<68>}\X{\<197>}\X{\<83>}\X{\<161>}\<119>}%
\def\plftitle{\mm \X{\<78>\<110>}\X{\<123>\<88>}\X{\<167>}\X{\<196>}}%
\def\plttitle{\mm \X{\<116>\<83>}\X{\<95>\<110>}\X{\<102>\<112>}\X{\<73>}\X{\<196>}}%
%
\def\mtctitle{\mm \X{\<68>}\X{\<197>}\X{\<83>}\X{\<161>}\<119>}%
\def\mlftitle{\mm \X{\<78>\<110>}\X{\<123>\<88>}\X{\<167>}\X{\<196>}}%
\def\mlttitle{\mm \X{\<116>\<83>}\X{\<95>\<110>}\X{\<102>\<112>}\X{\<73>}\X{\<196>}}%
%
\def\stctitle{\mm \X{\<68>}\X{\<197>}\X{\<83>}\X{\<161>}\<119>}%
\def\slftitle{\mm \X{\<78>\<110>}\X{\<123>\<88>}\X{\<167>}\X{\<196>}}%
\def\slttitle{\mm \X{\<116>\<83>}\X{\<95>\<110>}\X{\<102>\<112>}\X{\<73>}\X{\<196>}}%
⟨/malayalam – keli⟩
```

8.81 "Malayalam-rachana" language: malayalam-rachana.mld

The titles for the "malayalam" language, with the traditionnal "Rachana" fonts (old lipi), are taken from the malayalam package [1]. This language requires specific fonts. See also sections 8.80 on the preceding page and 8.82.

```
⟨*malayalam – rachana⟩
\ProvidesFile{malayalam-rachana.mld}[2005/06/07]
%
% Malayalam: Rachana fonts, traditionnal.
%
\def\ptctitle{\mm \X{\<68>}\X{\<201>}\X{\<83>}\X{\<183>}\<119>}%
\def\plftitle{\mm \X{\<78>\<111>}\X{{\C\<94>}}\X{\<186>}\X{\<179>}}%
\def\plttitle{\mm \X{\<117>\<83>}\X{\<95>\<111>}\X{{\F\<59>}}\X{\<73>}\X{\<179>}}%
%
\def\mtctitle{\mm \X{\<68>}\X{\<201>}\X{\<83>}\X{\<183>}\<119>}%
\def\mlftitle{\mm \X{\<78>\<111>}\X{{\C\<94>}}\X{\<186>}\X{\<179>}}%
\def\mlttitle{\mm \X{\<117>\<83>}\X{\<95>\<111>}\X{{\F\<59>}}\X{\<73>}\X{\<179>}}%
%
\def\stctitle{\mm \X{\<68>}\X{\<201>}\X{\<83>}\X{\<183>}\<119>}%
\def\slftitle{\mm \X{\<78>\<111>}\X{{\C\<94>}}\X{\<186>}\X{\<179>}}%
\def\slttitle{\mm \X{\<117>\<83>}\X{\<95>\<111>}\X{{\F\<59>}}\X{\<73>}\X{\<179>}}%
⟨/malayalam – rachana⟩
```

8.82 "Malayalam-rachana2" language: malayalam-rachana2.mld

The titles for the "malayalam" language, with the reformed "Rachana" fonts (new lipi), are taken from the malayalam package [1]. This language requires specific fonts. See also sections 8.80 on the preceding page and 8.81.

```
⟨*malayalam – rachana2⟩
```

```
\ProvidesFile{malayalam-rachana2.mld}[2005/06/07]
%
% Malayalam: Rachana fonts, reformed.
%
\def\ptctitle{\mm \X{\<68>}\X{\<201>}\X{\<83>}\X{\<183>}\<119>}
\def\plftitle{\mm \X{\<78>\<111>}\X{\<125>\<88>}\X{\<186>}\X{\<179>}}
\def\plttitle{\mm \X{\<117>\<83>}\X{\<95>\<111>}\X{\<106>\<113>}\X{\<73>}\X{\<179>}}
%
\def\mtctitle{\mm \X{\<68>}\X{\<201>}\X{\<83>}\X{\<183>}\<119>}
\def\mlftitle{\mm \X{\<78>\<111>}\X{\<125>\<88>}\X{\<186>}\X{\<179>}}
\def\mlttitle{\mm \X{\<117>\<83>}\X{\<95>\<111>}\X{\<106>\<113>}\X{\<73>}\X{\<179>}}
%
\def\stctitle{\mm \X{\<68>}\X{\<201>}\X{\<83>}\X{\<183>}\<119>}
\def\slftitle{\mm \X{\<78>\<111>}\X{\<125>\<88>}\X{\<186>}\X{\<179>}}
\def\slttitle{\mm \X{\<117>\<83>}\X{\<95>\<111>}\X{\<106>\<113>}\X{\<73>}\X{\<179>}}
⟨/malayalam – rachana2⟩
```

8.83 “Mongol” language: mongol.mld

The titles for the “mongol” language are taken from the MonTEX package [14, 15]. This language requires specific fonts. See also sections 8.13 on page 232 and 8.20 on page 235.

```
⟨*mongol⟩
\ProvidesFile{mongol.mld}[1999/03/16]
%% Mongol (xalx) titles for minitoc.sty
%% Needs mongol fonts
\def\ptctitle{{\mnr Garqig}}
\def\plftitle{{\mnr Zurgi"in jagsaalt}}
\def\plttitle{{\mnr X"usn"agti"in jagsaalt}}
%%
\def\mtctitle{{\mnr Garqig}}
\def\mlftitle{{\mnr Zurgi"in jagsaalt}}
\def\mlttitle{{\mnr X"usn"agti"in jagsaalt}}
%%
\def\stctitle{{\mnr Garqig}}
\def\slftitle{{\mnr Zurgi"in jagsaalt}}
\def\slttitle{{\mnr X"usn"agti"in jagsaalt}}
⟨/mongol⟩
```

8.84 “Naustrian” language: naustrian.mld

The “naustrian” language is a synonym of the “ngermanb” language (a revised version of the germanb variant of the german language), so we load ngermanb.mld. See also section 8.86 on the following page.

```
⟨*naustrian⟩
```

```
\ProvidesFile{naustrian.mld}[2004/12/14]
\mtcselectlanguage{ngermanb}%
⟨/naustrian⟩
```

8.85 “Ngerman” language: ngerman.mld

The “ngerman” language is a synonym of the “ngermanb” language[26], so we load `ngermanb.mld`. See also section 8.86.

```
⟨*ngerman⟩
\ProvidesFile{ngerman.mld}[2004/12/14]
\mtcselectlanguage{ngermanb}%
⟨/ngerman⟩
```

8.86 “Ngermanb” language: ngermanb.mld

The titles for the “ngermanb” language[27] are taken from the `babel` package [10]. See also sections 8.84 on the preceding page, and 8.85.

```
⟨*ngermanb⟩
\ProvidesFile{ngermanb.mld}[1999/12/06]
%% New german (B) titles for minitoc.sty
\def\ptctitle{Inhaltsverzeichnis}%    % oder nur: Inhalt
\def\plftitle{Abbildungsverzeichnis}%
\def\plttitle{Tabellenverzeichnis}%
%%
\def\mtctitle{Inhalt}%
\def\mlftitle{Abbildungsverzeichnis}%
\def\mlttitle{Tabellenverzeichnis}%
%%
\def\stctitle{Inhalt}%
\def\slftitle{Abbildungsverzeichnis}%
\def\slttitle{Tabellenverzeichnis}%
⟨/ngermanb⟩
```

[26] A revised version of the `germanb` variant of the german language.

[27] A variant of the german language, with revised spelling.

8.87 “Norsk” language: norsk.mld

The titles for the “norsk” language (or bokmål, lamguage of the kingdom) are taken from the babel package [10], with help from Dag Langmyhr. See also section 8.88.

```
⟨*norsk⟩
\ProvidesFile{norsk.mld}[1999/03/16]
%% Norsk titles for minitoc.sty
%% Thanks to Dag Langmyhr (dag@ifi.uio.no)
\def\ptctitle{Innhold}%
\def\plftitle{Figurer}% or Figurliste
\def\plttitle{Tabeller}% or Tabelliste
%%
\def\mtctitle{Innhold}%
\def\mlftitle{Figurer}% or Figurliste
\def\mlttitle{Tabeller}% or Tabelliste
%%
\def\stctitle{Innhold}%
\def\slftitle{Figurer}% or Figurliste
\def\slttitle{Tabeller}% or Tabelliste
⟨/norsk⟩
```

8.88 “Nynorsk” language: nynorsk.mld

The titles for the “nynorsk” language [28] are taken from the babel package [10], with help from Dag Langmyhr. See also section 8.87.

```
⟨*nynorsk⟩
\ProvidesFile{nynorsk.mld}[1999/03/16]
%% Nynorsk titles for minitoc.sty
%% Thanks to Dag Langmyhr (dag@ifi.uio.no)
\def\mtctitle{Innhald}%
\def\mlftitle{Figurar}% or Figurliste
\def\mlttitle{Tabellar}% or Tabelliste
%%
\def\ptctitle{Innhald}%
\def\plftitle{Figurar}% or Figurliste
\def\plttitle{Tabellar}% or Tabelliste
%%
\def\stctitle{Innhald}%
\def\slftitle{Figurar}% or Figurliste
\def\slttitle{Tabellar}% or Tabelliste
⟨/nynorsk⟩
```

[28] Created around 1800 by Ivar Åssen to make a real independant and national norvegian language, in reaction to danish, from the various dialect spoken in the country.

8.89 "Polish" language: polish.mld

The titles for the "polish" language are taken from the babel package [10]. See also section 8.90.

```
⟨*polish⟩
\ProvidesFile{polish.mld}[1999/03/16]
%% Polish titles for minitoc.sty
\def\ptctitle{Spis rzeczy}%
\def\plftitle{Spis rysunk\'ow}%
\def\plttitle{Spis tablic}%
%%
\def\mtctitle{Spis rzeczy}%
\def\mlftitle{Spis rysunk\'ow}%
\def\mlttitle{Spis tablic}%
%%
\def\stctitle{Spis rzeczy}%
\def\slftitle{Spis rysunk\'ow}%
\def\slttitle{Spis tablic}%
⟨/polish⟩
```

8.90 "Polish2" language: polish2.mld

The titles for the "polish2" language [29] are taken from the ANTOMEGA project [24]. See also section 8.89.

```
⟨*polish2⟩
\ProvidesFile{polish2.mld}[2005/02/08]
%% from omega-polish.ldf (Antomega project)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
\def\ptctitle{\localpolish{Spis tre^^^^00b1ci}}%
\def\plftitle{\localpolish{Spis rysunk^^^^00adw}}%
\def\plttitle{\localpolish{Spis tablic}}%
%%
\def\mtctitle{\localpolish{Spis tre^^^^00b1ci}}%
\def\mlftitle{\localpolish{Spis rysunk^^^^00adw}}%
\def\mlttitle{\localpolish{Spis tablic}}%
%%
\def\stctitle{\localpolish{Spis tre^^^^00b1ci}}%
\def\slftitle{\localpolish{Spis rysunk^^^^00adw}}%
\def\slttitle{\localpolish{Spis tablic}}%
⟨/polish2⟩
```

[29] "Polish2" is a variant of "polish".

8.91 “Portuges” language: portuges.mld

This is another spelling for “portuguese” (see section 8.92), so we just load portuguese.mld:

```
⟨*portuges⟩
\ProvidesFile{portuges.mld}[2005/06/06]
\mtcselectlanguage{portuguese}
⟨/portuges⟩
```

8.92 “Portuguese” language: portuguese.mld

The titles for the “portuguese” language are taken from the babel package [10]. See also section 8.14 on page 232, because the titles are different in Brazil, even if the language is also portuguese.

```
⟨*portuguese⟩
\ProvidesFile{portuguese.mld}[2005/07/08]
%% Portuguese titles for minitoc.sty
\def\ptctitle{Conte\'udo}%
\def\plftitle{Lista de Figuras}%
\def\plttitle{Lista de Tabelas}%
%%
\def\mtctitle{Conte\'udo}%
\def\mlftitle{Lista de Figuras}%
\def\mlttitle{Lista de Tabelas}%
%%
\def\stctitle{Conte\'udo}%
\def\slftitle{Lista de Figuras}%
\def\slttitle{Lista de Tabelas}%
⟨/portuguese⟩
```

8.93 “Romanian” language: romanian.mld

The titles for the “romanian” language are taken from the babel package [10].

```
⟨*romanian⟩
\ProvidesFile{romanian.mld}[1999/03/16]
%% Romanian titles for minitoc.sty
\def\ptctitle{Cuprins}%
\def\plftitle{List\u{a} de figuri}%
\def\plttitle{List\u{a} de tabele}%
%%
\def\mtctitle{Cuprins}%
```

```
\def\mlftitle{List\u{a} de figuri}%
\def\mlttitle{List\u{a} de tabele}%
%%
\def\stctitle{Cuprins}%
\def\slftitle{List\u{a} de figuri}%
\def\slttitle{List\u{a} de tabele}%
⟨/romanian⟩
```

8.94 “Russian” language: russian.mld

The titles for the “russian” language are taken from the babel package [10]. Specific cyrillic fonts are required.

```
⟨*russian⟩
\ProvidesFile{russian.mld}[1999/03/16]
%% Russian titles for minitoc.sty
\def\ptctitle{Oglavlenie}%
\def\plftitle{Pere{\cz}en{\mz} risunkov}%
\def\plttitle{Pere{\cz}en{\mz} tablic}%
%%
\def\mtctitle{Oglavlenie}%
\def\mlftitle{Pere{\cz}en{\mz} risunkov}%
\def\mlttitle{Pere{\cz}en{\mz} tablic}%
%%
\def\stctitle{Oglavlenie}%
\def\slftitle{Pere{\cz}en{\mz} risunkov}%
\def\slttitle{Pere{\cz}en{\mz} tablic}%
⟨/russian⟩
```

8.95 “Russian2m” language: russian2m.mld

The titles for the “russian2m” language (“russian2m” is a modern variant of “russian”) are taken from the ANTOMEGA project [24]. Specific cyrillic fonts are required. See also section 8.94.

```
⟨*russian2m⟩
\ProvidesFile{russian2m.mld}[2005/02/08]
%% from russian2m.ldf (Antomega project, russian modern)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
%% Needs cyrillic fonts
%%
\def\ptctitle{\localrussian%
{^^^^041e^^^^0433^^^^043b^^^^0430^^^^0432^^^^043b^^^^0435^^^^043d%
```

```
 ^^^^0438^^^^0435}}%
\def\plftitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a ^^^^0438^^^^043b%
 ^^^^043b^^^^044e^^^^0441^^^^0442^^^^0440^^^^0430^^^^0446^^^^0438%
 ^^^^0439}}%
\def\plttitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a ^^^^0442^^^^0430%
 ^^^^0431^^^^043b^^^^0438^^^^0446}}%
%%
\def\mtctitle{\localrussian%
{^^^^041e^^^^0433^^^^043b^^^^0430^^^^0432^^^^043b^^^^0435^^^^043d%
 ^^^^0438^^^^0435}}%
\def\mlftitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a ^^^^0438^^^^043b%
 ^^^^043b^^^^044e^^^^0441^^^^0442^^^^0440^^^^0430^^^^0446^^^^0438%
 ^^^^0439}}%
\def\mlttitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a ^^^^0442^^^^0430%
 ^^^^0431^^^^043b^^^^0438^^^^0446}}%
%%
\def\stctitle{\localrussian%
{^^^^041e^^^^0433^^^^043b^^^^0430^^^^0432^^^^043b^^^^0435^^^^043d%
 ^^^^0438^^^^0435}}%
\def\slftitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a ^^^^0438^^^^043b%
 ^^^^043b^^^^044e^^^^0441^^^^0442^^^^0440^^^^0430^^^^0446^^^^0438%
 ^^^^0439}}%
\def\slttitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a ^^^^0442^^^^0430%
 ^^^^0431^^^^043b^^^^0438^^^^0446}}%
⟨/russian2m⟩
```

8.96 "Russian2o" language: russian2o.mld

The titles for the "russian2o" language ("russian2o" is an old variant of "russian") are taken from the ANTOMEGA project [24]. Specific cyrillic fonts are required. See also section 8.94 on the page before.

```
⟨*russian2o⟩
\ProvidesFile{russian2o.mld}[2005/02/08]
%% from russian2o.mld (Antomega project - russian old)
%% Needs Omega
%% Alexej M. Kryokov
%% Dmitry Ivanov
%%
\def\ptctitle{\localrussian%
{^^^^041e^^^^0433^^^^043b^^^^0430^^^^0432^^^^043b^^^^0435^^^^043d%
 ^^^^0456^^^^0435}}%
\def\plftitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a^^^^044a ^^^^0438%
```

```
 ^^^^043b^^^^043b^^^^044e^^^^0441^^^^0442^^^^0440^^^^0430^^^^0446%
 ^^^^0456^^^^0439}}%
\def\plttitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a^^^^044a ^^^^0442%
 ^^^^0430^^^^0431^^^^043b^^^^0438^^^^0446^^^^044a}}%
%%
\def\mtctitle{\localrussian%
{^^^^041e^^^^0433^^^^043b^^^^0430^^^^0432^^^^043b^^^^0435^^^^043d%
 ^^^^0456^^^^0435}}%
\def\mlftitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a^^^^044a ^^^^0438%
 ^^^^043b^^^^043b^^^^044e^^^^0441^^^^0442^^^^0440^^^^0430^^^^0446%
 ^^^^0456^^^^0439}}%
\def\mlttitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a^^^^044a ^^^^0442%
 ^^^^0430^^^^0431^^^^043b^^^^0438^^^^0446^^^^044a}}%
%
\def\stctitle{\localrussian%
{^^^^041e^^^^0433^^^^043b^^^^0430^^^^0432^^^^043b^^^^0435^^^^043d%
 ^^^^0456^^^^0435}}%
\def\slftitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a^^^^044a ^^^^0438%
 ^^^^043b^^^^043b^^^^044e^^^^0441^^^^0442^^^^0440^^^^0430^^^^0446%
 ^^^^0456^^^^0439}}%
\def\slttitle{\localrussian%
{^^^^0421^^^^043f^^^^0438^^^^0441^^^^043e^^^^043a^^^^044a ^^^^0442%
 ^^^^0430^^^^0431^^^^043b^^^^0438^^^^0446^^^^044a}}%
⟨/russian2o⟩
```

8.97 "Russianb" language: russianb.mld

The titles for the "russianb" language ("russianb" is a variant of "russian") are taken from the babel package [10]. Specific cyrillic fonts are required. See also section 8.94 on page 271.

```
⟨*russianb⟩
\ProvidesFile{russianb.mld}[1999/03/16]
%% russianb.mld
\def\ptctitle{%
  {\cyr \CYRO\CYRg\CYRl\CYRa\CYRv\CYRl\CYRe\CYRn\CYRi\CYRe}}%
\def\plftitle{%
  {\cyr \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
    \CYRi\CYRl\CYRl\CYRyu\CYRs\CYRt\CYRr\CYRa\CYRc\CYRi\CYRishrt}}%
\def\plttitle{%
  \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
  \CYRt\CYRa\CYRb\CYRl\CYRi\CYRc}%
}
%%
\def\mtctitle{%
  {\cyr \CYRO\CYRg\CYRl\CYRa\CYRv\CYRl\CYRe\CYRn\CYRi\CYRe}}%
```

```
\def\mlftitle{%
  {\cyr \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
    \CYRi\CYRl\CYRl\CYRyu\CYRs\CYRt\CYRr\CYRa\CYRc\CYRi\CYRishrt}}%
\def\mlttitle{%
  \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
  \CYRt\CYRa\CYRb\CYRl\CYRi\CYRc}%
}
%%
\def\stctitle{%
  {\cyr \CYRO\CYRg\CYRl\CYRa\CYRv\CYRl\CYRe\CYRn\CYRi\CYRe}}%
\def\slftitle{%
  {\cyr \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
    \CYRi\CYRl\CYRl\CYRyu\CYRs\CYRt\CYRr\CYRa\CYRc\CYRi\CYRishrt}}%
\def\slttitle{%
  \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
  \CYRt\CYRa\CYRb\CYRl\CYRi\CYRc}%
}
⟨/russianb⟩
```

8.98 “Russianc” language: russianc.mld

The titles for the “russianc” language (“russianc” is a variant of “russian”, used in the part of Mongolia under russian influence) are taken from the file russian.def in the MonTEX package [14, 15]. Specific cyrillic fonts are required. See also section 8.94 on page 271.

```
⟨*russianc⟩
\ProvidesFile{russianc.mld}[1999/03/16]
%% Needs cyrillic fonts
%% Russian titles for minitoc.sty
%% Needs cyrillic fonts
\def\ptctitle{\xalx{Oglawlenie}}
\def\plftitle{\xalx{Spisok risunkow}}
\def\plttitle{\xalx{Spisok tablic}}
%%
\def\mtctitle{\xalx{Soderjanie}}
\def\mlftitle{\xalx{Spisok risunkow}}
\def\mlttitle{\xalx{Spisok tablic}}
%%
\def\stctitle{\xalx{Soderjanie}}
\def\slftitle{\xalx{Spisok risunkow}}
\def\slttitle{\xalx{Spisok tablic}}
⟨/russianc⟩
```

8.99 "Samin" language: samin.mld

The titles for the "samin" language[30] are taken from the samin.dtx file of the babel package [10]. Specific fonts are required.

```
⟨*samin⟩
\ProvidesFile{samin.mld}[2001/02/28]
%% Samin titles for minitoc.sty
\def\ptctitle{Sisdoallu}%
\def\plftitle{Govvosat}%
\def\plttitle{Tabeallat}%
%%
\def\mtctitle{Sisdoallu}%
\def\mlftitle{Govvosat}%
\def\mlttitle{Tabeallat}%
%%
\def\stctitle{Sisdoallu}%
\def\slftitle{Govvosat}%
\def\slttitle{Tabeallat}%
⟨/samin⟩
```

8.100 "Scottish" language: scottish.mld

The titles for the "scottish" language are taken from the babel language [10]:

```
⟨*scottish⟩
\ProvidesFile{scottish.mld}[1999/03/16]
%%Scottish titles for minitoc.sty
\def\ptctitle{Cl\`ar-obrach}
\def\plftitle{LiostaDhealbh}
\def\plttitle{LiostaChl\`ar}
%%
\def\mtctitle{Cl\`ar-obrach}
\def\mlftitle{LiostaDhealbh}
\def\mlttitle{LiostaChl\`ar}
%%
\def\stctitle{Cl\`ar-obrach}
\def\slftitle{LiostaDhealbh}
\def\slttitle{LiostaChl\`ar}
⟨/scottish⟩
```

[30] Several Sami dialects/languages are spoken in Finland, Norway, Sweden, and on the Kola Peninsula (Russia). The alphabets differ, so there will eventually be a need for more .dtx files for e.g. Lule and South Sami. Hence the name samin.dtx (and not sami.dtx or the like) in the North Sami case. This note is copied from the samin.dtx file.

8.101 "Serbian" language: serbian.mld

The titles for the "serbian" language are taken from the babel package [10]. See also section 8.102.

```
⟨*serbian⟩
\ProvidesFile{serbian.mld}[1999/12/06]
%%Serbian titles for minitoc.sty
\def\ptctitle{Sadr\v{z}aj}%
\def\plftitle{Slike}%
\def\plttitle{Tabele}%
%%
\def\mtctitle{Sadr\v{z}aj}%
\def\mlftitle{Slike}%
\def\mlttitle{Tabele}%
%%
\def\stctitle{Sadr\v{z}aj}%
\def\slftitle{Slike}%
\def\slttitle{Tabele}%
⟨/serbian⟩
```

8.102 "Serbianc" language: serbianc.mld

The titles for the "serbianc" language [31] have been given by Marko ÈEHAJA and Frank KÜSTER. Cyrillic fonts are required. See also section 8.101.

```
⟨*serbianc⟩
\ProvidesFile{serbianc.mld}[2001/11/25]
%% Provides titles for minitoc.sty in Serbian Cyrillic
%%
%%Marko Èehaja Internut@Thetaworld.Org
%%Frank Küster, Biozentrum der Univ. Basel, frank@kuesterei.ch
%%Abt. Biophysikalische Chemie
\def\ptctitle{%
  {\cyr\CYRS\cyra\cyrd\cyrr\cyrzh\cyra\cyrje}}%
\def\plftitle{%
  {\cyr\CYRS\cyrl\cyri\cyrk\cyre}}%
\def\plttitle{\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyre}%
%%
\def\mtctitle{%
  {\cyr\CYRS\cyra\cyrd\cyrr\cyrzh\cyra\cyrje}}%
\def\mlftitle{%
  {\cyr\CYRS\cyrl\cyri\cyrk\cyre}}%
\def\mlttitle{\cyr\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyre}%
%%
\def\stctitle{%
  {\cyr\CYRS\cyra\cyrd\cyrr\cyrzh\cyra\cyrje}}%
```

[31] The "serbianc" language is written with cyrillic characters.

```
\def\slftitle{%
  {\cyr\CYRS\cyrl\cyri\cyrk\cyre}}%
\def\slttitle{\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyre}%
⟨/serbianc⟩
```

8.103 “Slovak” language: slovak.mld

The titles for the “slovak” language are taken from the babel package [10]:

```
⟨*slovak⟩
\ProvidesFile{slovak.mld}[1999/03/16]
%% Slovak titles for minitoc.sty
\def\ptctitle{Obsah}%
\def\plftitle{Zoznam obr\'azkov}%
\def\plttitle{Zoznam tabuliek}%
%%
\def\mtctitle{Obsah}%
\def\mlftitle{Zoznam obr\'azkov}%
\def\mlttitle{Zoznam tabuliek}%
%%
\def\stctitle{Obsah}%
\def\slftitle{Zoznam obr\'azkov}%
\def\slttitle{Zoznam tabuliek}%
⟨/slovak⟩
```

8.104 “Slovene” language: slovene.mld

The titles for the “slovene” language are taken from the babel package [10]:

```
⟨*slovene⟩
\ProvidesFile{slovene.mld}[1999/03/16]
%% Slovene titles for minitoc.sty
\def\ptctitle{Kazalo}%
\def\plftitle{Slike}%
\def\plttitle{Tabele}%
%%
\def\mtctitle{Kazalo}%
\def\mlftitle{Slike}%
\def\mlttitle{Tabele}%
%%
\def\stctitle{Kazalo}%
\def\slftitle{Slike}%
\def\slttitle{Tabele}%
⟨/slovene⟩
```

8.105 “Spanish” language: spanish.mld

The titles for the “spanish” language are taken from the babel package [10]. Note that the “spanish” is in fact “castillan” (see section 8.23 on page 236). Other languages are spoken in Spain: “basque” (section 8.12 on page 231), “catalan” (section 8.25 on page 237), and “galician” (section 8.47 on page 246). “Spanish2” is a version of “spanish” with shorter titles (see section 8.106). “Spanish3” (see section 8.107 on the next page) is a version for the ANTOMEGA [24] project; some titles are differents.

```
⟨*spanish⟩
\ProvidesFile{spanish.mld}[1999/03/16]
%% Spanish titles for minitoc.sty
\def\ptctitle{\'Indice General}%
\def\plftitle{\'Indice de Figuras}%
\def\plttitle{\'Indice de Tablas}%
%%
\def\mtctitle{\'Indice General}%
\def\mlftitle{\'Indice de Figuras}%
\def\mlttitle{\'Indice de Tablas}%
%%
\def\stctitle{\'Indice General}%
\def\slftitle{\'Indice de Figuras}%
\def\slttitle{\'Indice de Tablas}%
⟨/spanish⟩
```

8.106 “Spanish2” language: spanish2.mld

The titles for the “spanish2” language are taken from the babel package [10], but made shorter for chapter and section levels. See section 8.105.

```
⟨*spanish2⟩
\ProvidesFile{spanish2.mld}[2005/03/31]
%% Spanish titles for minitoc.sty
\def\ptctitle{\'Indice General}%
\def\plftitle{\'Indice de Figuras}%
\def\plttitle{\'Indice de Tablas}%
%%
\def\mtctitle{Contenido}%
\def\mlftitle{Figuras}%
\def\mlttitle{Tablas}%
%%
\def\stctitle{Contenido}%
\def\slftitle{Figuras}%
\def\slttitle{Tablas}%
⟨/spanish2⟩
```

8.107 "Spanish3" language: spanish3.mld

The titles for the "spanish3" language are taken from the omega-spanish.ldf file of the ANTOMEGA project [24]. See section 8.105 on the page before.

```
⟨*spanish3⟩
\ProvidesFile{spanish3.mld}[2005/09/06]
%% Spanish titles for minitoc.sty
%% from omega-spanish.ldf of the ANTOMEGA project.
\def\ptctitle{\localspanish{^^^^00cdndice general}}%
\def\plftitle{\localspanish{^^^^00cdndice de figuras}}%
\def\plttitle{\localspanish{^^^^00cdndice de cuadros}}%
%%
\def\mtctitle{\localspanish{^^^^00cdndice general}}%
\def\mlftitle{\localspanish{^^^^00cdndice de figuras}}%
\def\mlttitle{\localspanish{^^^^00cdndice de cuadros}}%
%%
\def\stctitle{\localspanish{^^^^00cdndice general}}%
\def\slftitle{\localspanish{^^^^00cdndice de figuras}}%
\def\slttitle{\localspanish{^^^^00cdndice de cuadros}}%
⟨/spanish3⟩
```

8.108 "Swedish" language: swedish.mld

The titles for the "swedish" language are taken from the babel package [10]:

```
⟨*swedish⟩
\ProvidesFile{swedish.mld}[1999/03/16]
%% Swedish titles for minitoc.sty
\def\ptctitle{Inneh\csname aa\endcsname ll}%
\def\plftitle{Figurer}%
\def\plttitle{Tabeller}%
%%
\def\mtctitle{Inneh\csname aa\endcsname ll}%
\def\mlftitle{Figurer}%
\def\mlttitle{Tabeller}%
%%
\def\stctitle{Inneh\csname aa\endcsname ll}%
\def\slftitle{Figurer}%
\def\slttitle{Tabeller}%
⟨/swedish⟩
```

8.109 "Thai" language: thai.mld and thai.mlo

The titles for the "thai" language are taken from the babel package [10], using fonts of the CJK system [29].

The titles for the "thai" language contain characters that cannot be easily generated, hence we load thai.mlo.

```
⟨*thai⟩
\ProvidesFile{thai.mld}[2005/01/28]
%% from thaicjk.ldf CJK 4.5.2 Thai support for the babel system
%% by Werner Lemberg <wl@gnu.org>
%%
\mtcloadmlo{thai}
⟨/thai⟩
```

8.110 "Turkish" language: turkish.mld

The titles for the "turkish" language are taken from the babel package [10]:

```
⟨*turkish⟩
\ProvidesFile{turkish.mld}[1999/03/16]
%% Turkish titles for minitoc.sty
\def\ptctitle{\.I\c cindekiler}%
\def\plftitle{\c Sekiller Listesi}%
\def\plttitle{Tablolar\in Listesi}%
%%
\def\mtctitle{\.I\c cindekiler}%
\def\mlftitle{\c Sekiller Listesi}%
\def\mlttitle{Tablolar\in Listesi}%
%%
\def\stctitle{\.I\c cindekiler}%
\def\slftitle{\c Sekiller Listesi}%
\def\slttitle{Tablolar\in Listesi}%
⟨/turkish⟩
```

8.111 "UKenglish" language: UKenglish.mld

The "UKenglish" language is just like "english", so we just load english.mld (see section 8.32 on page 240):

```
⟨*UKenglish⟩
\ProvidesFile{UKenglish.mld}[2005/07/11]
\mtcselectlanguage{english}%
⟨/UKenglish⟩
```

8.112 “Ukraineb” language: ukraineb.mld

The titles for the “ukraineb” language are taken from the babel package [10]. Cyrillic fonts are required.

```
⟨*ukraineb⟩
\ProvidesFile{ukraineb.mld}[1999/12/06]
%% Ukraine (B) titles for minitoc.sty
%% Needs cyrillic fonts
\def\mtctitle{{\cyr\CYRZ\cyrm\cyrii\cyrs\cyrt}}%
\def\mlftitle{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
      \ \cyrii\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyrii\cyrishrt}}%
\def\mlttitle{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
      \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyrsftsn}}%
%%
\def\ptctitle{{\cyr\CYRZ\cyrm\cyrii\cyrs\cyrt}}%
\def\plftitle{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
      \ \cyrii\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyrii\cyrishrt}}%
\def\plttitle{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
      \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyrsftsn}}%
%%
\def\stctitle{{\cyr\CYRZ\cyrm\cyrii\cyrs\cyrt}}%
\def\slftitle{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
      \ \cyrii\cyrl\cyryu\cyrs\cyrt\cyrr\cyra\cyrc\cyrii\cyrishrt}}%
\def\slttitle{{\cyr\CYRP\cyre\cyrr\cyre\cyrl\cyrii\cyrk
      \ \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyrsftsn}}%
⟨/ukraineb⟩
```

8.113 “USenglish” language: USenglish.mld

The “USenglish” language is just like “english”[32], so we just load english.mld (see section 8.32 on page 240):

```
⟨*USenglish⟩
\ProvidesFile{USenglish.mld}[2005/07/11]
\mtcselectlanguage{english}%
⟨/USenglish⟩
```

[32]It should be true for the mini-table titles; the languages themselves have some differences.

8.114 “Usorbian” language: lsorbian.mld

The titles for the “usorbian” language [33] are taken from the babel package [10]. See also section 8.77 on page 263.

```
⟨*usorbian⟩
\ProvidesFile{usorbian.mld}[1999/03/16]
%% Upper sorbian titles for minitoc.sty
%% Needs cyrillic fonts
\def\ptctitle{Wobsah}
\def\plftitle{Zapiswobrazow}
\def\plttitle{Zapistabulkow}
%%
\def\mtctitle{Wobsah}
\def\mlftitle{Zapiswobrazow}
\def\mlttitle{Zapistabulkow}
%%
\def\stctitle{Wobsah}
\def\slftitle{Zapiswobrazow}
\def\slttitle{Zapistabulkow}
⟨/usorbian⟩
```

8.115 “Vietnam” language: vietnam.mld

The titles for the “vietnam” language are taken from the vietnam.sty package. Vietnamese fonts are required. See also section 8.116 on the next page.

```
⟨*vietnam⟩
\ProvidesFile{vietnam.mld}[1999/03/16]
%% vietnamese titles for minitoc.sty
%%
\def\ptctitle{M\d{u}c l\d{u}c}
\def\plftitle{Danh s\'ach h\`inh v\~e}
\def\plttitle{Danh s\'ach b\h{a}ng}
%%
\def\mtctitle{M\d{u}c l\d{u}c}
\def\mlftitle{Danh s\'ach h\`inh v\~e}
\def\mlttitle{Danh s\'ach b\h{a}ng}
%%
\def\stctitle{M\d{u}c l\d{u}c}
\def\slftitle{Danh s\'ach h\`inh v\~e}
\def\slttitle{Danh s\'ach b\h{a}ng}
⟨/vietnam⟩
```

[33] Upper sorbian. Sorbian, or wendisch, is a member of the west slavic subgroup of indo-european languages spoken in Upper Lusatia in the german *länder* of Saxony and Brandenburg. The Sorbs are descendents of the Wends, the german name for the slavic tribes who occupied the area between the Elbe and Saale rivers in the west and the Odra (Oder) river in the east during the medieval period (VIth century).

8.116 "Vietnamese" language: vietnamese.mld

The "vietnamese" language is just a synonym for the "vietnam" language. So we just load `vietnam.mld`. Vietnamese fonts are required. See also section 8.115 on the preceding page.

```
⟨*vietnamese⟩
\ProvidesFile{vietnamese.mld}[2004/12/14]
\mtcselectlanguage{vietnam}%
⟨/vietnamese⟩
```

8.117 "Welsh" language: welsh.mld

The titles for the "welsh" language are taken from the `babel` package [10]:

```
⟨*welsh⟩
\ProvidesFile{welsh.mld}[1999/12/06]
%% Welsh titles for minitoc.sty
\def\ptctitle{Cynnwys}%
\def\plftitle{Rhestr Ddarluniau}%
\def\plttitle{Rhestr Dablau}%
%%
\def\mtctitle{Cynnwys}%
\def\mlftitle{Rhestr Ddarluniau}%
\def\mlttitle{Rhestr Dablau}%
%%
\def\stctitle{Cynnwys}%
\def\slftitle{Rhestr Ddarluniau}%
\def\slttitle{Rhestr Dablau}%
⟨/welsh⟩
```

Complements

Bibliography

[1] Alex AJ. Typesetting Malayalam Using LaTeX 2ε. http://www.tug.org/tex-archive/language/malayalam/doc/mm-usr.pdf, April 2003. alexaj@myrealbox.com.

[2] Donald Arseneau. The notoccite package. http://www.tug.org/tex-archive/macros/latex/contrib/misc/notoccite.sty, July 2000. asnd@triumf.ca.

[3] Donald Arseneau. The placeins package. http://www.tug.org/tex-archive/macros/latex/contrib/misc/placeins.sty and http://www.tug.org/tex-archive/macros/latex/contrib/misc/placeins.sty, June 2002. asnd@triumf.ca.

[4] Benjamin Bayart. *Joli manuel pour LaTeX 2ε*. Guide local pour l'ESIEE, December 1995. http://www.tug.org/tex-archive/info/JMPL.ps.gz.

[5] Emmanuel Beffara. Rubber Manual 1.0. http://www.pps.jussieu.fr/~beffara/soft/rubber/doc.html, July 2005.

[6] Berhanu Beyene, Manfred Kudlek, Olaf Kummer, and Jochen Metzinger. Ethiopian Language Support for the babel Package. http://www.tug.org/tex-archive/language/ethiopia/ethiop/doc/ethiodoc.ps, December 1998. Universität Hamburg.

[7] Javier Bezos. Typesetting Guaraní with TeX. http://www.tug.org/tex-archive/language/guarani/guarani.pdf, July 2004.

[8] Javier Bezos. The titlesec and titletoc packages. http://www.tug.org/tex-archive/macros/latex/contrib/titlesec/titlesec.pdf, January 2005. http://www.texytipografia.com/contact.php.

[9] Johannes Braams. Babel, a multilingual style-option system for use with LaTeX's standard document styles. *TUGboat*, 12(2):291–301, June 1991. http://www.tug.org/TUGboat/Articles/tb12-2/tb32braa.pdf.

[10] Johannes Braams. Babel, a multilingual package for use with LaTeX's standard document classes. http://www.tug.org/tex-archive/macros/latex/required/babel/babel.pdf

and `http://www.tug.org/tex-archive/macros/latex/required/babel/user.pdf`, June 2004. `babel@bramms.cistron.nl`.

[11] Johannes Braams, Victor Eijkhout, and Nico Poppelier. The development of national LaTeX styles. *TUGboat*, 10(3):401–406, November 1989.

[12] David Carlisle. The `xr` package. `http:/www.tug.org/tex-archive/macros/required/tools/xr.dtx`, May 1994. `carlisle@cs.man.ac.uk`.

[13] Steven Douglas Cochran. The `subfig` Package. `http://www.tug.org/tex-archive/macros/latex/contrib/subfig/subfig.pdf` , January 2004. `sds@cs.cmu.edu`, `cochran@ieee.org`.

[14] Oliver Corff. MonTeX – A Quick Guide *(draft)*. `http://www.tug.org/tex-archive/language/mongolian/montex/doc/mlsquick.pdf`, November 2001. `corff@zedat.fu-berlin.de`.

[15] Oliver Corff. MonTeX – mongolian for LaTeX 2_ε, Implementation Level System Documentation. `http://www.tug.org/tex-archive/language/mongolian/montex/doc/montex.pdf`, July 2002. `corff@zedat.fu-berlin.de`.

[16] Serguei Dachian, Arnak Dalalyan, and Vardan Vakopian. ArmTeX: a System for Writing in Armenian with TeX and LaTeX. `http://www.tug.org/tex-archive/language/armtex/v2.0/manual.ps`, written in armenian, June 1999. `Serguei.Dachian@univ-lemans.fr`.

[17] Bernard Desgraupes. *LaTeX, Apprentissage, guide et référence*. Vuibert, Paris, second edition, 2003.

[18] Jean-Pierre F. Drucbert. The `shorttoc` package. `http://www.tug.org/tex-archive/macros/latex/contrib/shorttoc/shorttoc.dtx`, August 2002. `jean-pierre.drucbert@onera.fr`.

[19] Jean-Pierre F. Drucbert. Le paquetage `minitoc`. `http://www.tug.org/tex-archive/macros/latex/contrib/minitoc/fminitoc.pdf`, July 2005. `jean-pierre.drucbert@onera.fr`.

[20] Jean-Pierre F. Drucbert. The `minitoc` package. `http://www.tug.org/tex-archive/macros/latex/contrib/minitoc/minitoc.pdf`, July 2005. `jean-pierre.drucbert@onera.fr`.

[21] Victor Eijkhout. The `comment` package. `http://www.tug.org/tex-archive/macros/latex/contrib/comment/`, October 1999. `victor@eijkhout.net`.

[22] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, Massachusetts, sixth edition, 1989. Revised to cover TeX3, 1991.

[23] Donald E. Knuth. *Le TeXbook, Composition informatique*. Vuibert, Paris, 2003. French translation of [22], by Jean-Côme Charpentier.

[24] Alexej M. Kryukov. Typesetting multilingual documents with ANTOMEGA. `http://www.tug.org/tex-archive/systems/omega/contrib/antomega/doc/omega/antomega/antomega.pdf`, September 2003.

[25] Klaus Lagally. ArabTEX—Typetting Arabic with vowels and ligatures. In *Proceedings of the 7th European TEX Conference*, pages 153–172, Prague, 1992. CsTUG. ISBN 80-210-0480-0, `lagallyk@acm.org`.

[26] Klaus Lagally. ArabTeX Version 3. `http://www.tug.org/tex-archive/language/arabtex/doc/html/arabtex.htm`, October 2003. `lagallyk@acm.org`.

[27] Leslie Michel Lamport. *LATEX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[28] LATEX3 Project Team. LATEX 2$_\varepsilon$ font selection. `http://www.tug.org/tex-archive/macros/latex/doc/fontguide.pdf`, February 2004.

[29] Werner Lemberg. The CJK package for LATEX 2$_\varepsilon$ — Multilingual support beyond babel. *TUGboat*, 18(3):214–224, 1997. `http://www.tug.org/TUGboat/Articles/tb18-3/cjkintro600.pdf`.

[30] Knut Lichert. LATEX 2$_\varepsilon$ for people in associations: `minutes.sty`. `http://www.tug.org/tex-archive/macros/latex/contrib/minutes/`, January 2001. `knut@lickert.net`.

[31] Nicolas Markey. Tame the BeaST (Tthe B to X of BIBTEX). `http://www.tug.org/tex-archive/info/bibtex/tamethebeast/ttb_en.pdf`, `markey@lsv.ens-cachan.fr`, June 2005.

[32] Rowland McDonnell. The `sectsty` package v2.0.2. `http://www.tug.org/tex-archive/macros/latex/contrib/sectsty/`, April 2002. `rowland.mcdonnell@physics.org`.

[33] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LATEX Companion 2*. Tools and Techniques for Computer Typesetting. Addison-Wesley Professional, Reading, Massachusetts, 2004.

[34] Tobias Oetiker, Hubert Partl, Helene Hyna, and Elisabeth Schlegl. Une courte (?) introduction à LATEX 2$_\varepsilon$. *Cahiers GUTenberg*, Hors-série numéro 2, February 2003. Translation in french of [35] by Matthieu Herrb, adaptation and complements by Daniel Flipo, see also: `http://www.tug.org/tex-archive/info/lshort/french/flshort-3.20.pdf`.

[35] Tobias Oetiker, Hubert Partl, Helene Hyna, and Elisabeth Schlegl. The Not So Short Introduction to LATEX 2$_\varepsilon$. `http://www.tug.org/tex-archive/info/lshort/english/lshort.pdf`, April 2004.

[36] Palash Baran Pal. Bangtex: a package for typesetting documents in Bangla using the TEX/LATEX systems. `http://tnp.saha.ernet.in/~pbpal/bangtex/bangtex.html`, January 2001.

[37] Oren Patashnik. BibTeXing. Documentation for general BibTeX users, `http://www.tug.org/tex-archive/biblio/bibtex/contrib/doc/btxdoc.pdf`, February 1988.

[38] Oren Patashnik. Designing BibTeX styles. The part of BibTeX's documentation that's not meant for general users, `http://www.tug.org/tex-archive/biblio/bibtex/contrib/doc/btxhak.pdf`, February 1988.

[39] Sebastian Rahtz and Heiko Oberdiek. Hypertext marks in LaTeX: a manual for `hyperref`. `http://www.tug.org/tex-archive/macros/latex/contrib/hyperref/doc/manual.pdf`, February 2004.

[40] Young Ryu. The TX Fonts. `http://www.tug.org/tex-archive/fonts/txfonts/doc/txfontsdoc.pdf`, December 2000.

[41] Stephan P. von Bechtolsheim. TeX in practice: Comments on a 4-volume, 1400-page series on TeX. *TUGboat*, 11(3):409–412, September 1990. `http://www.tug.org/TUGboat/Articles/tb11-3/tb29bechtolsheim.pdf`.

[42] Stephan P. von Bechtolsheim. *TeX in Practice*. Springer Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1992.

1. Basics.
2. Paragraphs, Maths and Fonts.
3. Tokens, Macros.
4. Output Routines.

(A progressive approach to the secrets of TeX).

[43] Peter R. Wilson. The `needspace` package. `http://www.tug.org/tex-archive/macros/latex/contrib/misc/needspace.sty`, January 2003. `pandgwilson@earthlink.net`.

[44] Peter R. Wilson. The `tocloft` package. `http://www.tug.org/tex-archive/macros/latex/contrib/tocloft/tocloft.pdf`, September 2003. `pandgwilson@earthlink.net`.

[45] Peter R. Wilson. The `abstract` package. `http://www.tug.org/tex-archive/macros/latex/contrib/abstract/abstract.pdf`, April 2004. `pandgwilson@earthlink.net`.

[46] Peter R. Wilson. The `appendix` package. `http://www.tug.org/tex-archive/macros/latex/contrib/appendix/appendix.pdf`, April 2004. `pandgwilson@earthlink.net`.

[47] Peter R. Wilson. The `tocbibend` package. `http://www.tug.org/tex-archive/macros/latex/contrib/tocbibind/tocbibind.pdf`, May 2004. `pandgwilson@earthlink.net`.

[48] Peter R. Wilson. The Memoir Class. `http://www.tug.org/tex-archive/macros/latex/contrib/memoir/memman.pdf` , February 2005. `pandgwilson@earthlink.net`.

[49] Peter R. Wilson and Donald Arseneau. The `iftmarg` package. `http://www.tug.org/tex-archive/macros/latex/contrib/misc/ifmtarg.sty`, March 2000. `pandgwilson@earthlink.net`, `asnd@triumf.ca`.

.

Change History

00 1990/10/01

Original version, by Nigel WARD.

00 1991/11/01

Revised to reuse `\chapter`, `\section`, `\subsection` commands transparently, generate toc-file-name automatically, assorted other cleanup (Dan JURAFSKY).

01 1993/06/01

Added `\chapterend` to terminate the scope of a minitoc. (*If you forgot putting* `\chapterend` at the end of *each* chapter, an entry for the next chapter will appear in each minitoc.) (Thanks to Yufan HU).

All the layout of the minitoc is in the `\minitableofcontents` command, so if someboby wants to redefine that layout, he has just to rewrite it (and only it).

At least three passes (3) of LaTeX are necessary to get correct minitocs (the first pass creates the `.mtc`⟨*X*⟩ files, the second uses them (but they may contain wrong page numbers) and recreates them, the third should be ok).

New design, to avoid allocating a newwrite, or file descriptor, for each chapter (a deadly bug) (Jean-Pierre F. DRUCBERT).

Problems: you *must* have `\chapterend` to terminate each chapter with a minitoc. How about avoiding this constraint?

Replaced the `minipage` environment by a `verse` environment, to allow a minitoc being split across pages.

Some mods added to work with `xr.sty` (external references). `xr.sty` version 5 is much more tolerant.

The depth of the minitoc is user-adjustable with the counter `minitocdepth` (similar to `tocdepth` for the table of contents).

Works with `\chapter[xxx]{yyy}` and floating bodies.Works with two columns (but the minitoc is composed in one column; how to make it to spread over the two columns?).

You can inhibit the minitoc for the next chapter by preceding it with `\minitocno`. (`\minitocyes` is useless for the user, because it is implicit *after* the `\chapter*` pseudo-chapters).

02 1993/07/05

Added compatibility with `hangcaption.sty` (the package `hangcaption` (if present) must be loaded *before* the `minitoc` package). *Beware* to options modifying `\@caption`.

03 1993/07/09

Version 3 not released (buggy).

04 1993/07/09

Added `\if@realch` to avoid contentslines from pseudo-chapters to go into the toc. The package file `mtcoff.sty` allows you to use a LaTeX document with minitoc commands and to make them transparent: just replace the minitoc option by mtcoff.

05 1993/07/13

Added a selection mechanism to not write spurious things in the minitocs.

06 1993/07/15

Fixed problems about chapters in the toc, removed obsolete `\caption` stuff (filters are better) added compatibility with `toch.sty` (`toch.sty` makes a table of chapters; if used, must be loaded *before* `minitoc.sty`).

07 1993/07/22

(*major differences*) Completely rewritten, using tricks from `xr.sty` (the version 5, by David CARLISLE). The info for minitocs is directly stolen from the `.toc` file. `\chapterend` and `\minitocno` are suppressed, `\minitoc`, `\dominitoc` and `\faketableofcontents` added.

08 1993/07/29

Spacing adjustements.

09 1993/08/04

Added mods for MS-DOS (search for MS-DOS, uncomment; search for UNIX, comment out). MS-DOS allows only 3 characters for extensions in file names (what a pity).

10 1993/08/05

Works now with appendices. Detects now the obsolete versions of `latex.tex` (`\@inputcheck` or `\reset@font` not defined).

11 1993/08/18

Added `\mtcSfont`, font for section entries, `\mtcSSfont` for subsection entries, `\mtcSSSfont` for subsubsection entries, `\mtcPfont` for paragraph entries, `\mtcSPfont` for subparagraph entries.

12 1993/12/16

Use `\kern` in place of `\vspace*`, and added penalties (`\nopagebreak`) to avoid a page break just before last `\mtc@rule`. Also added a `samepage` environnement. Removed old commented out lines from previous versions.

13 1993/12/17

Added `minilof` and `minilot` stuff. For MS-DOS, uncomment the definition of `\SHORTEXT`.

14 1994/01/03

Corrected some spacing problems (avoiding ~'s). `\mtifont` is changed from `\normalsize\bf` to `\large\bf`.

Corrected space under `minitoc/lof/lot` and added a `\raggedright` setting to avoid "underfull" warnings.

Removed the setting of `\clubpenalty` and `\widowpenalty` to `10000` (done by `\samepage`), and `\noindent`.

Simplified processing of optional argument in `\minitoc`, `\minilof` and `\minilot`.

Some modifications suggested by Donald Arseneau (thanks): `\@newread` becomes `\newread`, not outer version of `\newread`; `\empty` replaced by `\relax` in the spare definition of `\reset@font`.

15 1994/01/27

Added `\parttoc`, `\partlof` and `\partlot` for books, `\secttoc`, `\sectlof` and `\sectlot` for articles, with some commands and parameters parallel to those for mini-tables.

16 1994/02/02

Bug fixes (typos).

17 1994/06/23

'`n`' (null) synonym of '`e`' (empty) in the optional argument of `\minitoc`, `\dominitoc`, and siblings.

Compatibility with "LaTeX 2_ε". Thanks to Denis Roegel (who found the problem) and Frank Mittelbach (who gave the hints to solve).

18 1994/06/26

Introduce the language files as options. Many thanks to Michel Goossens (via Frank Mittelbach) who was inspired by the code of the `babel` package (by Johannes Braams).

Make `minitoc` really compatible with LaTeX 2_ε.

19 1994/08/16

`\mtcrule`, `\nomtcrule` etc. commands added.

Added stuff for numbering of chapters (parts, sections) not starting at 1. `\firstchapteris` etc. commands added.

Corrected `mtcswedish.sty` (Jan Michel Rynning.)

Corrected a bug in `\c@mti`.

Corrected appendix in articles.

20 1994/08/25

Added the `\mtcpagenumbers` and `\nomtcpagenumbers` commands (and siblings) to make minitocs with/without page numbers. Default: with page numbers.

Corrected (difficult bug) appendix in articles.

Corrected a problem with chapters numbered with uppercase roman numbers.
Corrected spacing before and after minitocs and siblings.
Corrected vertical spacing.

21 1994/09/07
Corrected typos in `minitoc.sty` and `minitoc.tex`.

22 1994/10/10
Corrected typos in `minitoc.sty`.

23 1994/11/08
Added a missing line in `\sectlof`.
Removed appendix stuff.
Removed stuff for `\firstchapteris` and co. These commands are obsolete.
Works with document classes resetting chapter (or section) number at each part (thanks to Denis ROEGEL).

24 1994/12/21
The `\protect` commands have been removed from the `.toc`, `.lot` and `.lot` files, so some internal macros have been corrected to be compatible with the LaTeX 2_ε release of December 1994. Thanks to Denis ROEGEL who did the work.

25 1996/09/13
Updated `mtcnorsk.sty` and added `mtcnynorsk.sty` on a suggestion from Dag LANGMYHR.

26 1996/11/14
Added breton, estonian, germanb, greek, irish, russianb, scottish, lower and upper sorbian; renamed "esperanto" by "esperant" like in the `babel` package.
Language specific files are now named ⟨*language*⟩`.mld` (replacing `mtc`⟨*language*⟩`.sty`) because they are not packages and it makes shorter names.

27 1996/12/20
`english.mld` loaded as default language.
Added `vietnam.mld` and `arab.mld`.
Corrections for starred sectionning commands.
Renamed `minitocoff.sty` into `mtcoff.sty` to keep the name short.

28 1997/10/29
Added *coffee* stuff.
Added `\addstarred` stuff (for starred chapter stuff).
Added autoconfiguration of extensions.
Added the `shortext` package option.
Added the afrikaan(s), brazil, and ethiopia(n) languages.
Fixed bug in parttocs.

28 1998/06/15
A typo corrected by Donald ARSENEAU:
`{\let@dottedtocline\@undottedtocline}{}`
should probably be
`{\let\@dottedtocline\@undottedtocline}{}`
(a backslash was missing after `\let`). Thanks to him.
Added the bahasa language.

28 1998/12/03
Added the `tight` and `loose` package options.

29 1999/03/16
Added the bicig, buryat, mongol and russianc languages.

29 1999/06/28
Added the armenian language (from ArmTeX).

29 1999/07/23
Added the `dotted/undotted` package options (default: `dotted`).

29 1999/07/29
Added the lithuanian language.

30 1999/12/06

Added the basque, ngermanb, serbian, ukraineb, and welsh languages.

Corrected a bug in `\sltname` definition (`mlt` should be `slt`).

31 2000/04/04

Added compatibility with the `hyperref` package, thanks to Heiko OBERDIEK, who has also simplified some code and fixed the infamous `\chapter*` bug.

32 2000/08/08

`\nomtcrule` corrected.

Added very (too) numerous new commands for the mini-table features: `\beforeparttoc`, `\beforepartlof`, `\beforepartlot`, `\afterparttoc`, `\afterpartlof`, `\afterpartlot`, `\thispageparttocstyle`, `\thispagepartlofstyle`, and `\thispagepartlotstyle`.

Documentation improved by Stefan ULRICH.

33 2000/12/07

Added new adjustment commands: `\mtcaddchapter`, `\mtcaddsection`, and `\mtcaddpart`. These commands add stuff in the `.toc`, `.lof` and `.lot` files for the `\chapter*` (`\section*` and `\part*`) problem. From a suggestion by Karl F. EVERITT.

33 2000/12/08

Corrected a feature in `\mtcaddchapter` and co. with a blank optional argument.

34 2000/12/13

Added `.mld` files for alternate names of languages: so, `american.mld` just loads `english.mld`, which contains the real definitions.

Added in the documentation a section for use with the `tocbibend` package.

35 2001/01/09

Added macros to test if a file is “empty” (*i.e.* empty, blank or inexistent) or “non empty” (*i.e.* useful). I used some code from Stephan P. von BECHTOLSHEIM.

Added the `checkfiles/nocheckfiles` package options.

Replaced `\The@chapter` by `\The@mtc`.

35 2001/02/26

Added `bulgarian.mld`, `hebrew.mld`, `icelandic.mld`, `latin.mld`, and `samin.mld`.

35 2001/03/09

Added `\mtcselectlanguage`.

35 2001/06/01

Fixed the `estionian` package option (missing).

35 2001/07/04

Added the interlingua language.

36 2002/02/11

Corrected an interaction with `\tableofcontents` which creates a `\chapter*` or a `\section*`, perturbing `mtc/stc` counters (problem signalled by Frank MITTELBACH).

36 2002/02/18

Corrected a spacing problem with empty titles (problem signalled by Frank MITTELBACH).

Workaround for the `\parttoc`-`\chapter*` problem.

36 2002/02/19

Added `\mtcskip` and `\mtcskipamount`.

36 2002/02/27

Fixed test for empty files.

36 2002/03/13

Added the bangla language.

36 2002/03/15

Reduced depth of `\mtc@strutbox`.

37 2003/01/24

Version #37 dropped.

38 2003/01/24

`\hrule` and `\vrule` replaced by `\rule` (LaTeX).

pt becomes \@pt and 0pt becomes \z@.
Added mtc@zrule for zero-dims rules.
Added the frenchb language (synonym of french).

38 2003/01/30
Added the flsection and flsectionb package options.
Changed the test for empty titles.

38 2003/01/31
The tight and loose package options are applied to \parttoc (Thomas Leonhardt).

38 2003/02/07
Package options flsection and flsectionb removed and replaced by the insection package option (like flsectionb).

38 2003/02/11
Corrected numbering of SLF, SLT.

38 2003/02/20
Added frenchle and frenchpro languages (synonyms of french).
Corrected secttocs, at least.

38 2003/03/18
Corrected some vertical spacings and struts (I added some mods by Frank Mittelbach, many thanks to him.). A lot of cleaning remains to do, but the release seems to be needed now.

39 2003/04/09
\nomtcpagenumbers and memoir class.
Modern font commands for compatibility with the memoir class.

39 2003/06/08
Added \@fileswfalse and \mtc@hook@beforeinputfile for the notoccite package (requested by Donald Arseneau); added the notoccite package option (loads the notoccite package).

39 2004/09/08
Added comments in .mld files using special fonts.
Added language options and .mld files for dialects: canadian (english), acadian, acadien, canadien (french), naustrian, ngerman (ngermanb).
Documentation: added a paragraph about making a TOC for appendices, eventually not listed in the main TOC.

39 2004/09/17
Corrections in the documentation; corrections about rules.

40 2004/12/09
Added a figure in minitoc.tex about the need of three compilations.
Added a paragraph about a problem with the appendix package.
Added some infos in minitoc.bug.
Added the japanese and castillan languages.
Removed the test on the presence of the multicol package in minitoc.tex, because multicol is a required package.

40 2004/12/13
Updated fminitoc.bib and minitoc.bib.

40 2004/12/14
Added the hints package option. This option is still experimental; you advice is welcome.

40 2004/12/20
Added fminitoc.pdf (french documentation in PDF format).

41 2005/01/05
Corrections in documentation.
Message added if some sectionning commands are not available.
Replaced \typeout commands in minitoc.sty by the \PackageInfo or \PackageWarning commands; with the line number when useful (\@gobble if no line number). Hence,

the package is less verbose (\PackageInfo writes only in the .log file, not on the terminal).

41 2005/01/06
Added the \mtcsetfont (Benjamin BAYART) and \mtcsettitlefont commands, with a much simpler syntax.

41 2005/01/10
Added bibliography.

41 2005/01/11
AMS classes: amsart and amsproc are incompatible with minitoc, amsbook needs precautions.

41 2005/01/12
Added \mtcsetformat.

41 2005/01/18
Added \mtcsettitle.
Added a hint for recommending the insection package option.

41 2005/01/19
Added a hint about coherence \dominitoc/\minitoc and co.
Added a hint about the presence of \dominitoc and co.
Improved documentation about hints.

41 2005/01/20
Added a hint about using short extensions with more that 99 parts or 99 chapters or 99 sections.

41 2005/01/25
\ptifont: \Huge\bfseries becomes \LARGE\bfseries.

41 2005/01/26
Added \mtcsetpagenumbers.

41 2005/01/28
Added many new language files: serbianc.mld, chinese1.mld, chinese2.mld, hangul1.mld, hangul2.mld, hangul3.mld, hangul4.mld, hanja1.mld, hanja2.mld, japanese2.mld, japanese3.mld, japanese4.mld, japanese5.mld, thai.mld.

41 2005/02/02
Added \mtcsetrules.

41 2005/02/03
Added \plfrule, \noplfrule, \mlfrule, \nomlfrule, \slfrule, \noslfrule, \pltrule, \nopltrule, \mltrule, \nomltrule, \sltrule, \nosltrule.

41 2005/02/04
Added the mtchideinmaintoc environment.

41 2005/02/08
Added latvian.mld, letton.mld, greek-mono.mld, greek-polydemo.mld, greek-polykatha.mld, polish2.mld, russian2m.mld, and russian2o.mld as new language files.

41 2005/02/09
Added the mtchideinmainlof and mtchideinmainlot environments.

41 2005/02/10
Added tests on the mtchideinmain* environments.

41 2005/02/14
Added \mtcfixindex.

42 2005/02/14
Replaced “language” by “langue” in the french documentation.

42 2005/02/15
Fixed a minor typo.

42 2005/02/16
Upgraded \mtcfixindex.

42 2005/02/21
Added \mtcsettitle, forgotten to be inserted in v41.

43 2005/02/21
Version 43: consolidation of v40, v41 and v42.
43 2005/02/24
Fixed a big bug in \mtcsetformat.
Fixed a bug in mtcoff.sty about \mtcfixindex.
43 2005/03/02
Added the INSTALL file and a chapter about installation.
Fixed the \mtcset... macros.
Moved history to the end of package code.
43 2005/03/07
Completed the hint about coherence \dominitoc/\minitoc and co.
Fixed a typo (Benjamin Bayart).
43 2005/03/08
Added a hint about coherence \minitoc and \tableofcontents.
43 2005/03/09
Added comments about fonts.
43 2005/03/10
Corrections in documentation.
43 2005/03/11
Added \mtcsetfeature.
43 2005/03/14
Added bulgarianb.mld (upper bulgarian).
43 2005/03/15
Added *[-\baselineskip] after the \\ after the top rule of each part level mini-table.
43 2005/03/16
Corrections in the arguments of \mtcsetfeature.
43 2005/03/18
Removed \markboth for minitocs (...) and secttocs (...).
43 2005/03/21
Added spanish2.mld.
43 2005/03/22
Added a hint for the abstract package.
43 2005/04/06
Added finnish2.mld, latin2.mld, and magyar2.mld.
Corrected the stc@verse environment.
43 2005/04/08
Renamed portuges.mld as portugues.mld.
43 2005/04/12
Correction in \mtcskip.
First version in .dtx format.
43 2005/04/14
Removed \ypart, \ychapter, \ysection, and stuff; unused.
43 2005/05/11
Added \mtcfixglossary.
Added minitoc.ist to format the index correctly.
Corrected a typo in \@dosectlot.
Print the documentation with "oneside" to have all marginal notes on left. Added the (extended) code of morefloats.sty (Don Hosek) to allow more marginpars and floats.
43 2005/05/26
Fixed rules in parttocs, partlofs and partlots.
43 2005/05/30
Added a hint about the sectsty package (must be loaded *before* minitoc).
Fixed chapter-level entries in parttocs, when page numbers must be removed.

43 2005/06/01

Added a hint about attempts to insert empty mini-tables.

Added a hint about the use of obsolete commands.

The mini-lists of figures or tables should not be printed empty even if `tocdepth` < 1.

43 2005/06/02

Added `\mtcsetdepth`.

Added the notion of depth for mini-tables of figures/tables.

The `hints` option is the default and no more considered as experimental.

43 2005/06/03

Added an error message in `\mtcsetdepth` if the counter is not available.

43 2005/06/06

Added `portuges.mld`, which loads `portugues.mld`.

43 2005/06/07

Added three variants for the malayalam language: `malayalam-keli.mld`, `malayalam-rachana.mld`, and `malayalam-rachana2.mld`.

43 2005/06/14

Added method for bilingual documentation

43 2005/06/15

Added `fminitoc.ist` to format correctly the index in french.

43 2005/06/16

Changed "Liste des Tables" by "Liste des Tableaux" in `french.mld`, and in the french documentation, to stick to the choices of the `babel` package.

43 2005/06/17

The file `fminitoc.dtx` is now generated by `minitoc.ins`.

43 2005/06/21

Added "OUI", "NON", "oui", "non", "O", and "o" as true/false keywords.

Compacted the code about detection of short/long extensions.

43 2005/06/22

Added "VRAI", "FAUX", "vrai", "faux", "V", and "v" as true/false keywords.

43 2005/06/23

Correctly set the `\ifFTR` flag to have the months in the right language in the bibliography.

43 2005/06/29

Set the flag `\mtcoffwarn@true` in `mtcoff.sty` if a command `\mtcadd...` is found.

43 2005/07/01

Added `castillian.mld`.

Renamed `portugues.mld` as `portuguese.mld`.

43 2005/07/11

Added `brazilian.mld`, `british.mld`, `UKenglish.mld`, and `USenglish.mld`.

43 2005/07/12

Suppressed "General:" in the changes history.

43 2005/07/13

Replaced some `\PackageWarning` commands by `\PackageInfo`.

43 2005/07/18

Restoring the correspondence of each language option with a `.mld` file.

43 2005/07/20

Improving the `mtchideinmainlof` and `mtchideinmainlot` environments.

43 2005/07/21

Added the `\decrementptc`, `\decrementmtc`, and `\decrementstc` commands.

Removing unused some flags `\if@mtc@setpagenumbers@act@` and `\if@mtc@setrules@act@`.

43 2005/07/22

Added a test on the version of the `placeins` package.

Corrected a bug in `mtcoff.sty`.

Improved some messages in `mtcoff.sty`.

43 2005/08/23
Added a note about `\FloatBarrier`.
43 2005/08/24
Added a note about an alignment problem in the minitocs. Updated `minitoc.bug`.
Made two versions of the `mtchideinmainlof` and `mtchideinmainlot` environment, depending of the presence of the corresponding depth counter.
The `memoir` class is incompatible if too recent.
43 2005/08/25
Added a comment about the position of the `\do...` preparation commands.
Corrections in the `mtchideinmainlof` and `mtchideinmainlot` environments.
43 2005/08/26
Added `guarani.mld`.
43 2005/08/29
Added `\incrementptc`, `\incrementmtc`, and `\incrementstc`. Added an optional argument to `\adjustptc`, `\adjustmtc`, and `\adjuststc`.
Added the `k-tight` and `k-loose` package options.
43 2005/09/02
Added a patch for the recent version of the `memoir` class.
43 2005/09/06
Added `spanish3.mld`.
43 2005/09/08
Use `\mtcselectlanguage` in language options and in “secondary” `.mld` files.
43 2005/09/09
Added `\mtcloadmlo` to be used in some `.mld` files to load a `.mlo` file.
43 2005/09/12
Added a test to forbid direct calls of `\mtcloadmlo` by the user.
43 2005/09/13
Added `farsi1.mld`, `farsi1.mlo`, `farsi2.mld`, and `farsi2.mlo`.
Added a note about the `rubber` script.
43 2005/09/15
Added `mtcglo.ist` to format the glossary.
43 2005/09/16
Removed the page numbers in the glossary. Done in the *mk scripts.

Index

Symbols

A

B

C

R

S

T

U

V

W

X

Y