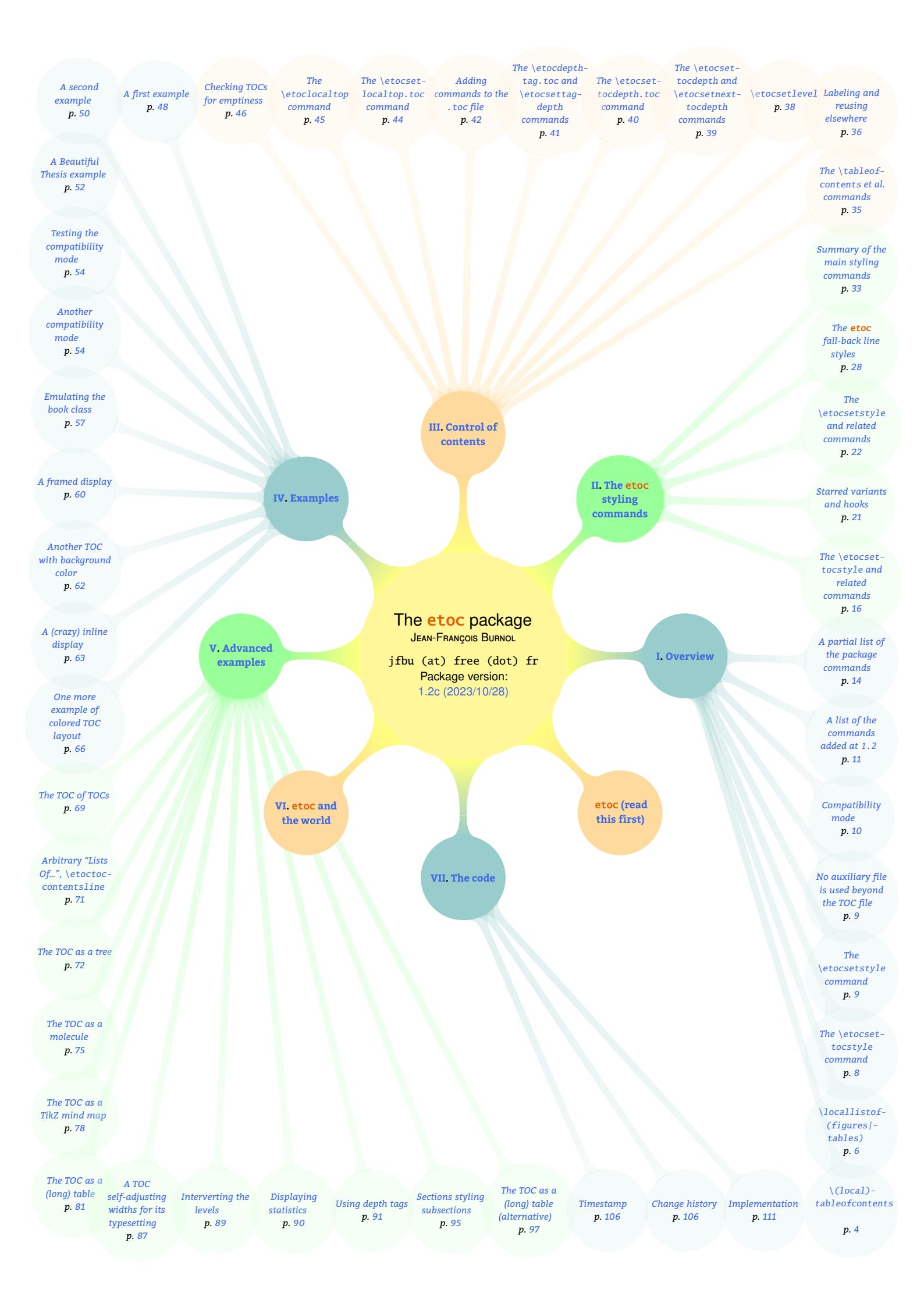

The etoc package
Jean-François Burnol
jfbu (at) free (dot) fr
Package version:
1.2c (2023/10/28)
I. Overview
II. The etoc styling commands
III. Control of contents
IV. Examples
V. Advanced examples
VI. etoc and the world
VII. The code
etoc (read this first)
Compatibility mode p. 10
A list of the commands added at 1.2 p. 11
A partial list of the package commands p. 14
The \etocsetstyle and related commands p. 16
Starred variants and hooks p. 21
The etoc fall-back line styles p. 28
Summary of the main styling commands p. 33
The \tableof-contents et al. commands p. 35
Labeling and reusing elsewhere p. 36
\etocsetlevel p. 38
The \etocset-tocdepth and \etocsetnext-tocdepth commands p. 39
The \etocset-tocdepth.toc command p. 40
The \etocdepth-tag.toc and \etocsettag-depth commands p. 41
Adding commands to the .toc file p. 42
The \etocset-localtop.toc command p. 44
The \etoclocaltop command p. 45
Checking TOCs for emptiness p. 46
A first example p. 48
A second example p. 50
A Beautiful Thesis example p. 52
Testing the compatibility mode p. 54
Another compatibility mode p. 54
Emulating the book class p. 57
A framed display p. 60
Another TOC with background color p. 62
A (crazy) inline display p. 63
One more example of colored TOC layout p. 66
The TOC of TOCs p. 69
Arbitrary “Lists Of...”, \etoctoc-contentsline p. 71
The TOC as a tree p. 72
The TOC as a molecule p. 75
The TOC as a TikZ mind map p. 78
The TOC as a (long) table p. 81
A TOC self-adjusting widths for its typesetting p. 87
Interverting the levels p. 89
Displaying statistics p. 90
Using depth tags p. 91
Sections styling subsections p. 95
The TOC as a (long) table (alternative) p. 97
Timestamp p. 106
Change history p. 106
Implementation p. 111
\(local)-tableofcontents p. 4
\locallistof-(figures|-tables) p. 6
The \etocset-tocstyle command p. 8
The \etocsetstyle command p. 9
No auxiliary file is used beyond the TOC file p. 9

# etoc

The command names throughout the user documentation which are displayed with this colour are doubly hyperlinked: the first two-thirds of the command name hyperlinks to the user documentation, and the remaining third part hyperlinks to the source code. You can try it out now: \localtableofcontents. But read this first: if you get lost inside the source code, clicking on control sequences displayed with this colour brings you back to the part of the user manual discussing that specific command.

## Abstract


With **etoc** loaded, \tableofcontents can be used multiple times and an added command \localtableofcontents allows to typeset "local" tables of contents, i.e. having their scope limited to the last sectioning command encountered.

No auxiliary file is used additionally to the standard .toc file. Release 1.2 provides EXPERIMENTAL additions \locallistoffigures and \locallistoftables which also use only the .toc file.

Such local TOCs or "Lists Of" typically need to adopt a "display style" (i.e. the way the title is rendered, whether it should add itself an entry in the .toc file, ...) somewhat distinct from the global TOC. The release 1.2 default adapts automatically the titles of local TOCs to their depths in the sectioning hierarchy. Should the need arise to customize such "display style", full control is allowed by package commands.

Regarding how the individual "contents lines" are handled, here again complete control is given to the user to define from the ground-up how to use the *name*, *number*, and *page number* for each entry, according to their "levels" (i.e. part, chapter, section, subsection, ...). As this requires some LaTeX fluency, many examples which can serve as starting points are attached to the PDF documentation as extractible files.

Loading **etoc** per itself modifies nothing to "contents lines" rendering from the class default or changes from other packages. But full usage of the package allows spectacular effects such as displaying TOCs as trees or mind maps.


## License

```
%          http://www.latex-project.org/lppl/lppl-1-3c.txt
%     and 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 2005/12/01 or later.
%
% The Author of this Work is:
% Jean-Francois Burnol <jfbu at free dot fr>
%
% This Work consists of the main source file etoc.dtx and the derived
% files etoc.sty, etoc.tex, etoc.pdf, etoc.dvi, README.md.
%
% Running etex (or latex or pdflatex) on etoc.dtx extracts etoc.sty,
% etoc.tex and README.md.  See README.md for further instructions.
```

# Part I.

# Overview

*Here are some statistics for this part: it contains 8 sections and no subsection. The name of the first section is "*`\(local)tableofcontents`*" and the corresponding number is "*1*". The name of the last section is "*A partial list of the package commands*" and its number is "*8*".*



## 1. The `\tableofcontents`, `\localtableofcontents` and other main package commands

`\tableofcontents` can be used arbitrarily many times in the document,

`\localtableofcontents` produces tables of contents which are limited in scope by the nearest preceding sectioning command,[1]

`\etocsettocstyle{`⟨***before_toc***⟩`}{`⟨***after_toc***⟩`}` defines (from the ground up, completely) the TOC titles, or more generally everything either before or after the actual TOC contents,

`\etocsetstyle{`⟨***levelname***⟩`}{`⟨***start***⟩`}{`⟨***prefix***⟩`}{`⟨***contents***⟩`}{`⟨***finish***⟩`}` defines (completely, from the ground up) arbitrarily the way the `\contentsline` entries in the `.toc` file are rendered, depending on their first argument, which is a 'level name' such as `part`, `chapter`, `section`, `subsection`, ...

It is possible to create new levels or re-assign the numerical level of a named one via `\etocsetlevel{`⟨*levelname*⟩`}{`⟨*number*⟩`}`, thus opening the way for sophisticated (ab)-uses of the data stored in the `.toc` file.

[1] After adding a `\localtableofcontents` in-between existing ones, the first compilation will typeset at its location a pre-existing next one; only at second compilation will the contents match the location. Hence often a third compilation is needed for document to stabilize. And if the "to toc" mechanism is active (see the discussion of `\etocsetup` option **`localtoctotoc`** and similar options), only on second compilation is a new entry made in the `.toc` file so three compilations is always a minimum. If one starts compilations from a given source, and no auxiliary files, the first compilation prepares the `.toc` file, the second typesets the main and local TOCs, possibly changing page numbers due to added contents, and only at the third compilation will things perhaps stabilize, and this third step is surely needed in case of **`localtoctotoc`**. latexmk is highly recommended.

**\etocname**, **\etocnumber**, **\etocpage** are free to use arbitrarily in the \etocsetstyle {⟨*prefix*⟩} and {⟨*contents*⟩} arguments for a given contents line level name, they stand for what will be extracted by **etoc** from the actual data which is stored in the .toc file and is a bit entangled there (for the first two).

Throughout this documentation "layout", "display", "TOC style" always are synonyms and refer roughly to how the TOC title is typeset. And "line styles" refer to how each individual entry in the TOC will be rendered according to its "level". Whenever "TOC style" is mentioned it is for the title related things, else the documentation will say "line styles".

\etocsetstyle is for line styles and it requires LaTeX fluency. It is however absolutely not required to use it: by default **etoc** does not intervene at all into the rendering of the contents lines. For example, one can use packages such as tocloft to customize these contents lines as wished.

\etocsettocstyle is for styling the titles (and all things related to material before and after the actual TOC contents). It was originally conceived mainly for being used after the main document TOC (assuming it comes first). By default **etoc** renders the main TOC as specified by the class it knows about. But for example in a book-class document, one does not want to use a chapter-like heading for a local TOC in a section. Hence the need for \etocsettocstyle. But if used in the preamble it will apply to the main TOC too (if it comes first in the document body), hence its usage has to be delayed or stored into some command, else one has to insert branches in the original definition to query the kind of TOC it is applied to.

With 1.2 some utilities facilitate using \etocsettocstyle only once in the preamble and branch according to whether it is handling a main document TOC or a local one. But the macro was really initially conceived to be used for local TOCs once the main TOC was typeset (commands are provided to restore the default configuration in time for a main TOC at end of document). There is no command to customize the titles only for local TOCs from the preamble.

With 1.2 an effort has been made to make using \etocsettocstyle for local TOCs purely an option: by default the package adopts a style for local headings which takes into account the local top level. This corresponds to the command \etocetoclocaltocstyle, which is emitted by default by the package.

Furthermore 1.2 detects tocbibind[2] and executes a compatilibity layer so that this package can be used with hopefully the same result as in the main document classes. Package options **maintoctotoc**, **localtoctotoc**, **localloftotoc** and **localtottotoc** trigger "to toc" mechanisms even without tocbibind if desired, directly at **etoc** level. The starred variants of the table of contents and "list of" commands will ignore the status of these options, as in the memoir class.

[2] Support for tocloft customization of contents lines was already in place for many years; the compatibility with tocbibind was added only at 1.2 and required to update the tocloft related code.

## 2. The 1.2 \locallistoffigures and \locallistoftables commands

The major novelty with 1.2 is the addition of \locallistoffigures and \locallistoftables.

The following major problem is known with the current implementation: \locallistoffigures and \locallistoftables actually list the figures, respectively the tables, say in a given \section which are **TYPESET THEREIN**, which is not necessarily the same figures, tables, which in the LaTeX source are located **"WITHIN" THAT SECTION**.

Recall that LaTeX does not have a structured representation of the document (despite what many people like to think), so there is no pre-existing official way from inside a table to know in which section or subsection it is defined. And figures and tables are *floating*.

If a figure, for example, floats to next page, and the section were it was *defined* is followed on the initial page by another one, then a \locallistoffigures in the first section will NOT see the figure which has drifted to next page! That floating figure will be listed by a \locallistoffigures in the NEXT section!

Now, if all sections are equipped with \locallistoffigures this current "feature" of etoc means that all figures will be accounted for and one will be able to see in which section they physically reside after typesetting!

Which is nice but perhaps counter intuitive.

The source of this problem is well-understood by package author but addressing the issue is delayed probably for some months, by lack of time and ressources; or may remain a permanent "feature" as the author does not intend to hack in any way into code for floats, or even captions.

Here is a non-etoc file you can play with and which illustrates the mechanism:

```
\documentclass{article}
\begin{document}
\tableofcontents % the table of contents will show the "table" entry
                 % as belonging to **second** section
\section{first}
\begin{table}[p]
  some tabular
  \makeatletter
  % add some entry in the toc file, like a subsection would do
  \write\@auxout{\string \@writefile {toc}
                         {\protect \contentsline {subsec-
tion}{table}{\thepage }{}}}
  \caption{table}
\end{table}

\section{second}
\end{document}
```

- This is experimental code and the user interface as well as the output may change.

- It is mandatory to load **etoc** with option **lof** for \locallistoffigures and **lot** for \locallistoftables.
- **etoc** does not interfere whatsoever with \listoffigures and \listoftables, and can not customize them in any way.
- **etoc** still uses no additional auxiliary file, it uses only the .toc file.
- This is experimental code and the user interface as well as the output may change.
- The new command \etocsetup allows to configure at any location in the preamble or in the document body the boolean options **maintoctotoc**, **localtoctotoc**, **localloftotoc**, **locallottotoc**, and **ouroboros** which can also be issued as package options. The first four default to false, except with the memoir class, in which case they are set to true, or when tocbibind is detected, in which case they are set to true depending on the setting of the tocbibind options.
- The **ouroboros** option defaults to true. When set to false, local TOCs which have been added to the .toc file due to **localtoctotoc** will *not* list themselves (they will still list entries from local TOCs at deeper levels in their scope, and the "lists of" at their same level, and are listed in the main TOC, the only thing is that they don't show themselves in themselves).[3] The default is that they *do* list themselves, hence the name **ouroboros**. It has no impact on the main TOCs, use their starred variant to inhibit the self-display, but then why have issued **maintoctotoc** in the first place?
- An effort has been made to facilitate as much as possible customization by the user, i.e. via commands without @ letter. However this documentation will only list their names with very brief comments at the appropriate place and the reader is expected to check in the source code to understand better what they are for.
- This is experimental code and the user interface as well as the output may change.
- For example, the actual figure entries or table entries use by default the code from the main LOF or LOT; but one may prefer to render them in a way taking into account the local "list of" scope: for example in a section, perhaps use the same style as subsections. This is possible and is explained briefly later.
- Although the default "**etoc**" choices for local titles should be hopefully satisfactory in most cases, it is also possible to tell **etoc** to try to use emulation from the document class via the command \etocclasstocstyle. An effort has been made to obtain something which works at one level deeper than the ones under \part. For example with KOMA-script, we use the option leveldown. However this will not really work for deeper local "lists of".[4] But for some document using local "lists of" only for chapters or for sections in classes

[3] It is complex to try to anticipate all scenarii, but in general, it is expected this behavior may cause the .toc file to need more time to stabilize when a new local TOC is added in the midst of a document having already a bunch of them.

[4] And with KOMA-script at version 3.30 or later the fact that unnumbered sections reset the subsection counter will cause bad problems in a scrbook document if a TOC, local to a \subsection uses a title which is un unnumbered section!

without chapters, this \etocclasstocstyle may be appropriate as it may allow the user to use the class interface for advanced control of the marks or other details.

- The etoc document class agnostic default, from \etocetoclocaltocstyle which is automatically issued at package loading time, is completely customizable.
- This is experimental code and the user interface as well as the output may change.
- If using for example all three of \localtableofcontents, \locallistoffigures and \locallistoftables, in the same location after a division heading, and if the "to toc" related options either from etoc or from tocbibind are used, then \localtableofcontents must be the first one in order to be able to list the other two (and itself). It can not see the "lists of" coming before itself at the same division level.
- Regarding the TOC title style, etoc knows the standard classes, the KOMA-script main classes and the memoir class. In an unknown class it will use the code from the article class emulation. For the local TOCs and local "lists of" it uses as with the known classes its own class-independent code from \etocetoclocaltocstyle.
- One can tell etoc to not replace the original \tableofcontents via \etockeeporiginaltableofcontents (it will still be possible to use \etoctableofcontents), but the non-etoc \tableofcontents is usable only if document does not use \locallistoffigures or \locallistoftables as the latter two insert entries for figures, resp. tables, in the main .toc file and only etoc's \tableofcontents knows to ignore them.
- This is experimental code and the user interface as well as the output may change.

## 3. The \etocsettocstyle command

This is a command with two mandatory arguments:

\etocsettocstyle{⟨*before_toc*⟩}{⟨*after_toc*⟩}

The {⟨*before_toc*⟩} part is responsible for typesetting the heading, for example it can be something like \section*{\contentsname}. Here is an example of input:

```
\etocsettocstyle
   {\section*{Local table of contents}}%
   {}% don't do anything special after the toc contents
```

Once issued it will have an impact on all next usages of \localtableofcontents (or \tableofcontents) until it is used again (or some related command).

Generally speaking the {⟨*before_toc*⟩} part should leave TEX in "vertical mode": the line styles (either from the standard classes or the package default ones) all expect to get started in 'vertical mode'.

For more documentation of this and related commands see first section 5 then section 9 for the more detailed pre-1.2 documentation.

Only daring people will continue reading documentation as it now starts telling how to truly activate etoc power.

## 4. The \etocsetstyle command

A distinction must be made between the *line styles*, *i.e.* the way the name, number and page numbers (aka **etoc**-provided \etocname, \etocnumber, and \etocpage) are used at each level, and the *toc display style* (for lack of a better name) which tells how the title should be set, whether an entry in the .toc file should be made, whether the contents should be typeset with multiple columns, etc... the latter is governed by the command \etocsettocstyle (and related commands) which has already been mentioned, and the former by the command \etocsetstyle with is the core of **etoc** functionality.

It has five mandatory arguments.

\etocsetstyle{⟨*levelname*⟩}{⟨*start*⟩}{⟨*prefix*⟩}{⟨*contents*⟩}{⟨*finish*⟩}

The first one is the name of the sectional unit: a priori known names are book, part, chapter, section, subsection, subsubsection, paragraph, and subparagraph. Any other name can be declared and assigned to a (numeric) level via the \etocsetlevel command.[5]

The four other arguments of \etocsetstyle specify: 1) *what to do when this level is first encountered, down from a more general one,* then 2) & 3) (two arguments, a 'prefix' and a 'contents') *how to use for this level the* \etocname*,* \etocnumber *and* \etocpage *parsed data,* and 4) *the last argument is the code to execute when a division unit of higher importance than the defined level style is encountered.*

Notice that this means that virtually **etoc** manages a kind of tree-like substratum which is abstracted from the 'flat' structure of the .toc data.

You should now read section 11 for the detailed documentation.

## 5. No auxiliary file is used beyond the TOC file

An important characteristic of **etoc** is that it is allows many different TOCs in the same document, *using only one* .toc *file*!

The present documentation contains 38 visible tables of contents (and a few invisible ones) and uses only one .toc file![6]

However, each \localtableofcontents or \tableofcontents command will trigger the execution of the *full contents* from the .toc file. The effect of \localtableofcontents as well as the enforcement of the line styles as defined via \etocsetstyle are achieved via suitable redefinition of the \contentsline LaTeX macro. But everything else present in the .toc file will be executed, as it is not possible for **etoc** to control in any way what is present in the .toc file beyond \contentsline entries.

So one should think twice before adding manually extra commands to the .toc file. See section 20 for further discussion.

[5] **etoc** issues automatically \etocsetlevel{appendix}{0} (or wit h 1 if the document has no \chapter command), since 1.2. Formerly this was done only with class memoir.

[6] and the counting itself has been achieved by a table of contents which was inserted in this paragraph! See section 34.

# 6. Compatibility mode

**etoc** starts in a "compatibility mode", which means that it does not at all interfere with how the commands from the `.toc` file get executed as long as it has not been told explicitly to do so.[7]

This "compatibility mode" stops for matters of the "toc display style" as soon as `\etocsettocstyle` is made use of, and for matters of the "toc line styles" as soon as `\etocsetstyle` is used for *any* level (part, chapter, section, ...). Levels not receiving explicit configurations will use some pre-defined defaults which are in **etoc** source code.

**`\etocclasstocstyle`** sets the 'main toc layout' to be as without `\usepackage{etoc}`. Local TOCs will also obey the document class style but with an attempt to use a heading one level down. This tries to adapt to being in the top level below `\part` (i.e. in a `\chapter` or `\section`) but is not adapted to deeper local TOCs.

At `1.2`, previously existing `\etocstandarddisplaystyle` was made into a deprecated synonym to this (the version from earlier releases made no attempt to adapt the style of the local TOCs to the level where it is located).

**`\etocetoclocaltocstyle`** (package default) Activates a heading style for local tables of contents (and "lists of") which tries to adapt automatically to the surrounding level. It thus by-passes the configuration done by `\etocsettocstyle` which will then apply only to the main TOC. Using again `\etocsettocstyle` de-activates this behavior.

**`\etocusertocstyle`** means to obey the `\etocsettocstyle` configuration as previously in place, not only for the main TOC but also for local ones. If `\etocsettocstyle` has never been used, this means that we return to the situation from `\etocclasstocstyle`.

**`\etocstandardlines`** (package default) Sets the 'content lines' to be as without `\usepackage{etoc}`.

**`\etoctoclines`** sets the 'content lines' to match the last encountered `\etocsetstyle` specs.

**`\etocdefaultlines`** sets the 'content lines' to use **etoc** pre-defined ones. The denomination is a bit confusing as 'default' here means that these line styles are the ones defined by default, but not used by default... perhaps `\etocfallbacklines` would have been a better name.

Please be aware that by design the `paragraph` and `subparagraph` linestyles are configured by `\etocdefaultlines` to display nothing at all.

Naturally `\etocsettocstyle` and `\etocsetstyle` can be used arbitrarily many times in the document body (or already in the preamble but there one can not typeset a TOC; however one can prepare commands which will be activated later from inside the document body). And they obey the scope-limiting effect of LaTeX environments.

[7] for the "toc display style", by this we mean the aspects independant from the contents of the `.toc` file, **etoc** checks if it knows the class, and then uses emulation code which was added manually to its source, and if not it defaults to the `article` class layout. No automated way to recover the global toc display for arbitrary document classes is implemented. But **etoc** will detect if tocloft has customized the TOC title.

For an illustration of using etoc in compatibility mode see section 27. And see section 28 for some ways to let the line styles use the code from the document class but with some custom changes.

During expansion of \locallistoffigures or \locallistoftables, the macros \etoclocallistoffigureshook resp. \etoclocallistoftableshook are executed right before typesetting the entries. Their default definitions (refer to source code for more info):

```
\def\etoclocallistoffigureshook{\etocstandardlines}%
\def\etoclocallistoftableshook {\etocstandardlines}%
```

means compatibility mode, i.e. the macros \l@figure and \l@table will be the ones involved, as if we were in the global \listoffigures or \listoftables.

If you redefine these "hook"-macros to do nothing, the figure/table entries will use the style as appropriate for entries at numerical level one deeper than the "local top": i.e. in a section they will be typeset as if being subsections. If you have both \locallistoffigures and a \localtableofcontents, this may align better vertically than the LaTeX line style from the global "Lists Of".

Note that "hook" is used here in a very naive meaning of some macro which is pre-located somewhere and that one can redefine to obtains various effects.

## 7. A list of the commands added at 1.2

One pre-existing command has been modified: \etocstandarddisplaystyle which is now deprecated and replaced by \etocetoclocaltocstyle. The former behavior was to set also local TOCs to apply the style from the main TOC, which basically never works really well. As explained already one can use \etocclasstocstyle to try to get for top local TOCs a suitable behavior, for example in the case of KOMA-script classes, a behavior using their leveldown option. But this will not be satisfactory for deeper local TOCs or "lists of".

First there is \etocsetup:

\etocsetup{⟨*key[=true|false],...*⟩} has already been mentioned earlier. It is usable with **maintoctotoc**, **localtoctotoc**, **localloftotoc**, **locallottotoc**, and **ouroboros**, everywhere in the document after the package loading.

Then some utilities relative to the \etocsetlevel command:

- \etoclevel{⟨*levelname*⟩} expands to a numeric quantity giving the level of a given string argument. For example \etoclevel{section} will usually produce a TeX number denotation of value one. It is provided for matters of programming, as its output format is suitable for usage in \ifnum test. For typesetting, it should be prefixed by the TeX primitive \number.
- \etocthelevel{⟨*levelname*⟩} expands to a explicit digits (possibly prefixed by a minus sign), i.e. it is equivalent to \number\etoclevel{⟨*levelname*⟩} and is provided as a convenience. (added at 1.2a)

- \etocthemaxlevel expands to either 12 or 6 (default) depending on whether the **deeplevels** option was used or not. See the documentation of \etocsetlevel for more. (added at 1.2a)
- \etocifunknownlevelTF{⟨*levelname*⟩}{⟨*True*⟩}{⟨*False*⟩} is a conditional to test if a level name has been declared to **etoc**.

Then two utilities to help store and restore toc line styles:

**\etocstorelinestylesinto{⟨*control_sequence*⟩}** This is a command with one mandatory argument which must be a control sequence such as \foo. The macro \foo is then overwritten with no check if it exists already. Its effect is to store inside \foo the current toc line styles configuration, for all levels at once. Then inserting \foo in the document will restore all line styles to the state they were in at time of usage of \etocstorelinestylesinto.

**\etocstorethislinestyleinto{⟨*name or number*⟩}{⟨*control_sequence*⟩}** Same principle but now \foo will store the line style for the specified level only. This argumant can be numerical or a name. And thus executing \foo will then have the same effect as re-doing the \etocsetstyle for that level.

The more numerous additions are relative to usage of \etocsettocstyle, i.e. customizing the "display" style. This mainly means matters relative to the title of the local TOCs or Lists Of, inclusive of whether they should add an entry into the .toc file matching their titles. Indeed, please be aware that as soon as you use \etocsettocstyle you will have to handle yourself the support for such behavior. Of course, as you also control the options, you don't have to use the provided command layer which tests the status of these options, but it is provided anyway as public interface. Indeed all of this is done via commands with no @ letter in their names, and the package itself uses this public interface in its implementation of the default behavior, adjusted to the document class. The documentation here is brief, check for more the source code to which the listed commands are hyperlinked (from the last third of their names).

Some of these commands are defined via batch processing done in loops in the code, and the hyperlinks only point to some initial dummy definitions, which were added only to provide such link targets, and one has to read the code further to discover the actual definitions.

- \etocifislocal, \etocifislocaltoc, \etocifislocallof, \etocifislocallot can be used from inside the \etocsettocstyle first or second argument to select either one of the {⟨*True*⟩} and {⟨*False*⟩} branches.
- \etocifmaintoctotoc, \etociflocaltoctotoc, \etociflocalloftotoc, \etociflocallottotoc, are conditionals matching the options and selecting one of the {⟨*True*⟩} or {⟨*False*⟩} branches.
- \etocifisstarred{⟨*True*⟩}{⟨*False*⟩} says from inside the \etocsettocstyle arguments if the toc or 'list of' command was in starred form.

  The \etocetoclocaltocstyle and \etocclasstocstyle have been configured so that when a * follows a \tableofcontents or \localtableofcontents or local "lists of", its

"to toc" behavior (if active from tocbibind or etoc own options) is canceled. They use \etocifisstarred to this effect.

- \etoclocalheadtotoc{⟨*levelname*⟩}{⟨*text*⟩} is a synonym for
  \addcontentsline{toc}{⟨*@levelname*⟩}{⟨*text*⟩}
  Pay attention to automatically added @ character.[8]

  Use this for example in the second argument of \etoclocalmulticol.[9]

- \etocglobalheadtotoc{⟨*levelname*⟩}{⟨*text*⟩} is a synonym for
  \addcontentsline{toc}{⟨*levelname*⟩}{⟨*text*⟩}
  We have so many commands we can define another useless one… for aesthetic reasons of coherent names…

- \etocdivisionnameatlevel{⟨*number*⟩} is an expandable construct which starts from a numerical level from -2 to 5 and produces one of book, part, ..., up to subparagraph as per the built-in LaTeX (and standard classes) default assignments (and memoir for book).

  Under option **deeplevels** the numerical level can go up to 11 but as there is no standard default for documents with extra sectioning commands (e.g. is it using \subsubsubparagraph? or \subsubsubsection? or some \subivsection?), the user has probably to copy and redo with changes the package definition of \etocdivisionnameatlevel in that case.

- \etocetoclocaltocmaketitle is the command used by \etocetoclocaltocstyle which typesets a local "list of" title using an un-numbered sectioning appropriate to its scope. See the source code for how it looks.

- \etocetoclistoffiguresmaketitle and \etocetoclistoftablesmaketitle are similar, see the source code.

- \localcontentsname, \locallistfigurename, \locallisttablename are self-explanatory.

- \etocclasslocaltocmaketitle, \etocclasslocallofmaketitle, \etocclasslocallotmaketitle, \etocclassmaintocaddtotoc, \etocclasslocaltocaddtotoc, \etocclasslocallofaddtotoc, \etocclasslocallotaddtotoc, well again see source code.

Finally there is a command to help store/restore a given display style configuration.

**\etocstoretocstyleinto{⟨*control_sequence*⟩}** This is a command with one mandatory argument which must be a control sequence such as \foo. The macro \foo is then overwritten with no check if it exists already. Its effect is to store inside \foo the data configured by the last \etocsettocstyle. Then inserting \foo in the document will restore the saved toc style.[10]

[8] Without this syntax, it would not be possible to have all three of \localtableofcontents, \locallistoffigures and \locallistoftables one after the other following a given document heading, due to deep internals of the local TOC etoc mechanism. Notice that these internal details are susceptible to change with no advance notice and the actual implementation of \etoclocalheadtotoc may change.

[9] There is currently no analog of \etoclocalmulticol and similar commands which would be related to \locallistoffigures or \locallistoftables as \etoclocalmulticol is to \localtableofcontents.

[10] It is impossible to store the style used by the \etocetoclocaltocstyle for local TOCs. Only the one for main TOCs, which was configured by last usage of \etocsettocstyle is saved in \foo. After \foo is executed, one needs to again issue \etocetoclocaltocstyle if one wants the latter to be active.

# 8. A partial list of the package commands

\etocbeforetitlehook
\etocaftertitlehook
\etocaftercontentshook
\etocaftertochook
\etocsettocstyle
\etocclasstocstyle
\etocetoclocaltocstyle
\etocmulticol
\etocframed
\etocruled
\etocsettocdepth
\etocsettocdepth.toc
\etocsetnexttocdepth
\etocdepthtag.toc
\etocsettagdepth
\etocobeydepthtags
\etocdefaultlines
\etocstandardlines
\etoctoclines
\etocsetlevel
\etocsetstyle
\etocskipfirstprefix
\etocifnumbered
\etociffirst
\etoclink
\etocthelink
\etocname
\etocnumber
\etocpage
\etocthename
\etocthenumber
\etocthepage
\etocthelinkedname
\etocthelinkednumber
\etocthelinkedpage
\localtableofcontents
\localtableofcontentswithrelativedepth
\locallistoffigures
\locallistoftables
\tableofcontents

The above is not an exhaustive list of all the package user commands. For example, it does not include most of those added at 1.2 and which were listed in the previous section.

# Part II.

# The etoc styling commands

*Here are some statistics for this part: it contains 5 sections and 22 subsections. The name of the first section is "*The \etocsettocstyle and related commands*" and the corresponding number is "*9*". The name of the last section is "*Summary of the main styling commands*" and its number is "*13*". The name of the first subsection is "*The \etocsettocstyle command*" and the corresponding number is "*9.1*". The name of the last subsection is "*Labels and references*" and its number is "*13.4*".*

**Contents of Part II**

The two main commands \etocsettocstyle and \etocsetstyle can be used anywhere in the document. Typically one will render the main global TOC in one style and local tables of contents in another. So the commands to style the local tables of contents will be executed after the main \tableofcontents (if it is first in document) either by direct injection in the document source, or encapsulated in user commands defined in the preamble.

All commands obey the scope limiting effect induced by LaTeX environments or the core TeX\begingroup/\endgroup (or braces {...}) constructs.

# 9. The \etocsettocstyle and related commands



## 9.1. The \etocsettocstyle command

The basics are explained in section 3. Recall that the syntax is

\etocsettocstyle{⟨*before_toc*⟩}{⟨*after_toc*⟩}

and here is a typical example of use:

```
\etocsettocstyle
   {\section*{Local table of contents}}%
   {}% don't do anything special after the toc contents
```

The first argument to \etocsettocstyle can also contain instructions to mark the page headings. Or it could check to see if two-column mode is on, and switch to one-column style, and the ⟨*after_toc*⟩ part would then reenact the two-column mode.

The commands to be described next \etocmulticolstyle, \etocruledstyle, and \etocframedstyle all call \etocsettocstyle as a lower-level routine, to initiate a multicols environment in {⟨*before_toc*⟩} and close it in {⟨*after_toc*⟩}.

### 9.1.1. The \etocarticlestyle, \etocbookstyle, and others commands

These are the commands used internally by **etoc** in compatibility mode depending on the document class. For example \etocarticlestyle instructs **etoc** to use \section*{\contentsname}

(with marks on the page) and \etocbookstyle says to use \chapter*{\contentsname}. It can prove useful to issue \etocarticlestyle for a \localtableofcontents inside a chapter, in book class and compatibility mode for the global TOC display style.

Here is the current list:

- \etocarticlestyle
- \etocarticlestylenomarks
- \etocreportstyle
- \etocreportstylenomarks
- \etocbookstyle
- \etocbookstylenomarks
- \etocmemoirstyle
- \etocscrartclstyle
- \etocscrreprtstyle
- \etocscrbookstyle

The command \etocclasstocstyle will adapt to the document class. One can not use the KOMA-script related commands or the memoir one in standard classes.

### 9.1.2. The \etocinline and \etocdisplay commands

With \etocinline, or its synonym \etocnopar, the \localtableofcontents command and its variants do *not* first issue a \par to close the previous paragraph. Hence, the table of contents can be printed in an inline style; or, if used only for preparing some token list or macro, it will leave nothing in the token stream on execution.

Issue \etocdisplay to return to the default situation that \localtableofcontents and variants issue a \par to switch to vertical mode before typesetting the TOC title and contents.

Here is an example of an *inline* table of contents, which has only subsections and uses the itemize* environment from enumitem for them. The code used is

```
And the output is:
\begingroup
    \etocglobaldefs
    \etocsetstyle {subsection}
          {\begin{itemize*}[itemjoin={{; }}, itemjoin*={{, and }}]}
          {}
          {\item [{\bfseries\etocnumber.}] \etocname\ (\emph{p. \etocpage })}
          {\end{itemize*}.}%
    \etocsetnexttocdepth {subsection}%
    \etocsettocstyle {a clone of a local table of contents, originally defined in
                      \autoref{sec:tocstyle}, but here rendered completely
                      differently via an inline \ctanpkg{enumitem} list: }{}%
    \etocinline\tableofcontents \ref{toc:tocstyle}
\endgroup
```

(observe that on executing the above there is no extra space after the colon beyond what is intended, and the current paragraph is simply continued) And the output is: a clone of a local table of contents, originally defined in section 9, but here rendered completely differently via

an inline enumitem list: **9.1.** The \etocsettocstyle command (*p. 16*); **9.2.** The \etocmulticolstyle, \etocmulticol, and \etoclocalmulticol commands (*p. 18*); **9.3.** The \etoctocstyle command (*p. 18*); **9.4.** The \etocruledstyle, \etocruled and \etoclocalruled commands (*p. 19*); **9.5.** The \etocframedstyle, \etocframed, and \etoclocalframed commands (*p. 19*); **9.6.** Customizing the pre-defined toc display styles (*p. 20*), and **9.7.** Headings, titles, \etocoldpar, \etocinnertopsep (*p. 21*).

It was needed to use \etocglobaldefs because the \item command, as modified by enumitem closes a group, hence the meaning of \etocname, \etocnumber and \etocpage would have been lost after it.

A more impressive example of an inline table of contents (containing the full contents of this document, which subsections rendered as page footnotes...) is to be found in section 32.

## 9.2. The \etocmulticolstyle, \etocmulticol, and \etoclocalmulticol commands

This is a command with one optional and one mandatory argument:

\etocmulticolstyle[⟨*number_of_columns*⟩]{⟨*heading*⟩}

The ⟨*number_of_columns*⟩ can go from 1 to 10 (it defaults to 2; if its value is 1, naturally no multicols environment is then created). The ⟨*heading*⟩ will typically be some 'vertical' material like: ⟨*heading*⟩ = \section*{⟨*title*⟩} but one may also have horizontal material like \fbox{Hello World} (**etoc** adds automatically a \par at the end of this "heading" argument to \etocmulticolstyle). Here is an example:

```
\etocmulticolstyle{\noindent\bfseries\Large
                   \leaders\hrule height1pt\hfill
                   \MakeUppercase{Table of Contents}}
\localtableofcontents
```

After \etocmulticolstyle all future \tableofcontents will use the specified style until modified by renewed usage of \etocsettocstyle or variants.

A shortcut combining the style specification and the table of contents and not impacting the styles of later TOCs is:

\etoclocalmulticol[⟨*number_of_columns*⟩]{⟨*heading*⟩}

So the above example can be also obtained using:

```
\etoclocalmulticol{\noindent\bfseries\Large
                   \leaders\hrule height1pt\hfill
                   \MakeUppercase{Table of Contents}}
```

It has the advantage that the TOC styling as specified applies only to this sole local TOC.

And there is also \etocmulticol[⟨*number_of_columns*⟩]{⟨*heading*⟩} for global TOC.

## 9.3. The \etoctocstyle command

\etoctocstyle[⟨*kind*⟩]{⟨*number_of_columns*⟩}{⟨*title*⟩}

is the exact equivalent of doing

\etocmulticolstyle[*number_of_columns*]{\kind*{*title*}}

where kind is one of chapter, section, . . . and defaults to chapter or section depending on the document class.

As explained above one still has to issue \localtableofcontents to typeset the TOC. And the styling, if not enclosed in a scope-limiting group or environment, applies to subsequent local TOCs.

#### 9.3.1. The \etoctocstylewithmarks command

\etoctocstylewithmarks[⟨*kind*⟩]{⟨*number_of_columns*⟩}{⟨*title*⟩}{⟨*mark*⟩}

is the exact equivalent of doing

\etocmulticolstyle[*number_of_columns*]{\kind*{*title* \markboth{\MakeUppercase{*mark*}}}}

where kind is one of chapter, section, ... The actual display of the marks depends on the settings of the page style. There is variant \etoctocstylewithmarksnouc which does not uppercase.

##### 9.3.1.1. Do we really want paragraph entries in the TOC?

##### 9.3.1.2. really?

### 9.4. The \etocruledstyle, \etocruled and \etoclocalruled commands

The general format of \etocruledstyle is:

\etocruledstyle[⟨*number of columns*⟩]{⟨*title of the toc*⟩}

The title is horizontal material (the LR mode of *LaTeX, a document preparation system*): if it does not fit on one line it should be put in a \parbox of a given width. The green frame for the heading of the table of contents at the the start of this part was obtained with:

```
\etocruledstyle[1]{\etocfontminusone\color{green}%
     \fboxrule1pt\fboxsep1ex
     \framebox[\linewidth]
              {\normalcolor\hss Contents of this part\hss}}
\localtableofcontents
```

As a shortcut to set the style with \etocruledstyle and then issue a \localtableofcontents, all inside a group so that future table of contents will not be affected, there is:

\etoclocalruled[⟨*number_of_columns*⟩]{⟨*title*⟩}

And there is also \etocruled for the global TOC.

### 9.5. The \etocframedstyle, \etocframed, and \etoclocalframed commands

Same mechanism:

\etocframedstyle[⟨*number_of_columns*⟩]{⟨*title*⟩}

and the accompanying shortcut:

\etoclocalframed[⟨*number_of_columns*⟩]{⟨*title*⟩}

The shortcut is used if one does not want to modify the style of the next TOCs (the other way is

to put the whole thing inside braces or a \begingroup...\endgroup; there is also \etocframed for a global table of contents).

The entire table of contents is framed. The title itself is not framed: if one wants a frame one should set it up inside the ⟨*title*⟩ argument to \etocframedstyle or \etocframed. The colors for the background and for the components (top, left, right, bottom) of the border are specified via suitable \renewcommand's (see subsection 9.6).

A minipage is used, hence the produced table of contents isn't compatible with a page break. For allowing page breaks, use of the commands of mdframed or tcolorbox in the arguments of \etocsettocstyle is recommended.

Examples in this document are on pages 37, 64, 61, and 67.

## 9.6. Customizing the pre-defined toc display styles

We list the relevant macros, what they do should be legible from their names. Note that dimensions are stored in macros so are modifed using \renewcommand's and not \setlength's. And color related commands are not color definitions, they execute \color, and their effect gets canceled by re-defining them to do \relax or \empty.

```
\newcommand*\etocabovetocskip{3.5ex plus 1ex minus .2ex}
\newcommand*\etocbelowtocskip{3.5ex plus 1ex minus .2ex}

\newcommand*\etoccolumnsep{2em}
\newcommand*\etocmulticolsep{0ex}
\newcommand*\etocmulticolpretolerance{-1}
\newcommand*\etocmulticoltolerance{200}
\newcommand*\etocdefaultnbcol{2}
\newcommand*\etocinnertopsep{2ex}
\newcommand*\etoctoprule{\hrule}
\newcommand*\etoctoprulecolorcmd{\relax}

% for the framed style only:
\newcommand*\etocinnerleftsep{2em}
\newcommand*\etocinnerrightsep{2em}
\newcommand*\etocinnerbottomsep{3.5ex}

\newcommand*\etocleftrule{\vrule}
\newcommand*\etocrightrule{\vrule}
\newcommand*\etocbottomrule{\hrule}
\newcommand*\etocleftrulecolorcmd{\relax}
\newcommand*\etocrightrulecolorcmd{\relax}
\newcommand*\etocbottomrulecolorcmd{\relax}

\newcommand*\etocbkgcolorcmd{\relax}

% hooks
\newcommand\etocframedmphook{\relax}
```

The \etocframedmphook is positioned immediately after the beginning of a minipage environment where the contents of the framed TOC are typeset.

The \...colorcmd commands are initially set to expand to \relax (hence do not require package color or xcolor to be loaded). If one has modified a command such as \etocbkgcolorcmd to expand to a color command and wants to reset it to do nothing, one *must* use \renewcommand{\etocbkgcolorcmd}{\relax} and not \let\etocbkgcolorcmd\relax.

Regarding the dimensions of the top rule they can be specified in ex's or em's as in this example:

```
\renewcommand{\etoctoprule}{\hrule height 1ex}
```

The package code is done in such a manner that it is the font size in instance at the end of typesetting the title argument to \etocruled or \etocframed which will be used for the meaning of the '1ex'. Of course also the other rule commands can have their dimensions in font relative units, but their values are decided on the basis of the font in effect just before the table of contents.

The top and bottom rules do not have to be rules and can be horizontal *leaders* (of a specified height) in the general TEX sense. However the left and right rules are not used as (horizontal) leaders but as objects of a given specified width. Note that *only* the Plain TEX syntax for rules is accepted here.

### 9.7. Headings, titles, \etocoldpar, \etocinnertopsep

For \etocmulticolstyle the mandatory ⟨*heading*⟩ argument can be either vertical mode material like \section*{\emph{Table of Contents}} or horizontal mode material like in the simple \etocmulticolstyle{Hello World}.

No explicit \par or empty line can be inserted in the mandatory argument of \etocmulticolstyle, but **etoc** provides \etocoldpar as a substitute: it does \let\etocoldpar\par before the multicols environment and inserts this \etocoldpar[11] at the end of the heading, then does a vertical skip of value \etocinnertopsep. The command \etocoldpar can also be used explicitely if needed in the mandatory argument to \etocmulticolstyle (it is not allowed to insert an empty line in this argument).

On the other hand the commands \etocruledstyle and \etocframedstyle expect an argument "in LR mode" (to use the terminology from *LaTeX, a document preparation system*). This means that multiline titles are only possible if enclosing them inside something like a \parbox.

An important dimension used by all three of \etocmulticolstyle, \etocruledstyle and \etocframedstyle is \etocinnertopsep. It gives the amount of separation between the heading and the start of the contents. Its default value is 2ex and it is changed with \renewcommand*{\etocinnertopsep}{⟨*new_value*⟩}, not with \setlength.

## 10. Starred variants and hooks

The \tableofcontents, \localtableofcontents, \etocmulticol, and all their cousins have starred variants (the star must be before the other arguments). The non-starred variants execute

[11] this command \etocoldpar (= working \par in the argument to \etocmulticolstyle) is not related to the switch \etocinline whose purpose is to tell **etoc** not to do a \par before the table of contents.

the \etocaftertitlehook, whose default definition is to do nothing. The starred variants do not execute this hook.

For example, imagine you are using book class and want \localtableofcontents to use a section-like title, but unnumbered. Assuming the main \tableofcontents comes first in the document, you can insert this after it:

```
\etocarticlestyle
\renewcommand{\etocaftertitlehook}{\addcontentsline{toc}{section}{\contentsname}}
```

This configures the way \localtableofcontents will behave (or \tableofcontents) from now on in the document.

The first line tells essentially to use \section*{\contentsname}, and the second line says to insert the title in the .toc file itself (thus to be displayed by the main table of contents). Notice that hyperref package will then automatically create suitable anchor and one should *not* use explicitly \phantomsection here (it would let the anchor be located below not above the title).

With this set-up issuing \localtableofcontents* will ignore the \etocaftertitlehook hence not send the local toc title to the .toc file. This mimics the memoir class behavior, and can also be used with it. For more on memoir class with **etoc**, see subsection 47.2.

There are further hook macros: \etocaftercontentshook, \etocbeforetitlehook and \etocaftertochook which are initially defined to do nothing and can be used for some special effects. They are executed whether or not the table of contents command was starred.

For example, the present document executed

```
\renewcommand{\etocbeforetitlehook}{\setstretch{1}}
```

as it is globally using the setspace command \onehalfspacing. Not using \singlespacing in the hook as it does a systematic vertical skip of one baseline, which is unwanted in our usage.

> In recent years, the LaTeX kernel has added a general "hook" mechanism with a user interface of the type \AddToHook{...}{...}.
>
> The **etoc** macros with 'hook' ending their names are much simpler things which are supposed to be manipulated only via \renewcommand or \def direct overwrites.
>
> In future, and to the extent the author has time for that addition, with its costly documentation updates collaterals, and thoughts about backward compatibility, **etoc** should arguably tap into the general tools provided by recent LaTeX kernels.

## 11. The \etocsetstyle and related commands

Let us explain how **etoc** was used to produce the table of contents displayed at the beginning of this Part II. This is a local table of contents, and we used the command \localtableofcontents.

The line styles were (essentially) obtained in the following manner:[12]

```
\etocsetstyle{section}
{\begin{enumerate}}
{\normalsize\bfseries\rmfamily\item}
{\etocname{} (page \etocpage)}
{\end{enumerate}}

\etocsetstyle{subsection}
{\begin{enumerate}}
{\normalfont\item}
{\etocname{} (p.~\etocpage)}
{\end{enumerate}}

\etocsetstyle{subsubsection}
{\par\nobreak\begingroup\normalfont
        \footnotesize\itshape\etocskipfirstprefix}
{\allowbreak\,-\,}
{\etocname}
{.\hfil\par\endgroup\pagebreak[3]}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-01.tex

These provisory style definitions rely on the automatic numbering generated by the enumerate environments but it is much better to use the further command \etocnumber inside the item label, which gives the real thing. The improved definitions will thus be explained later.

With this style, one would have to be imaginative to design something then for paragraph and subparagraph entries! perhaps as superscripts? Well, usually one does not need paragraphs and subparagraphs numbered and listed in the TOC, so our putative user here chose a design where no provision is made for them and added the definitive:

```
\etocsetstyle{paragraph}{}{}{}{}
\etocsetstyle{subparagraph}{}{}{}{}
```

This is also the situation with the default package line styles!

Each \etocsetstyle command has five mandatory arguments:

\etocsetstyle{⟨*levelname*⟩}{⟨*start*⟩}{⟨*prefix*⟩}{⟨*contents*⟩}{⟨*finish*⟩}

The initially recognized ⟨*levelname*⟩'s are the sectioning levels of the standard document classes: from *part* (or *book* which is used by the memoir class) down to *subparagraph*.

The ⟨*start*⟩ code is executed when a toc entry of that level is encountered and the previous one was at a higher level. The ⟨*finish*⟩ code is executed when one again encounters a higher level

[12] the present document has \renewcommand{\familydefault}{\sfdefault} in its preamble, hence \normalfont switches to the sans typeface; so in the section line-style, I wrote \rmfamily instead.

toc entry. In the meantime all entries for that level are typeset by executing first the ⟨*prefix*⟩ code and then the ⟨*contents*⟩ code.

The (robust) commands \etocname, \etocnumber and \etocpage are provided for use inside the ⟨*prefix*⟩ and ⟨*contents*⟩ parts of the \etocsetstyle specification. They represent of course, the name, number, and page number of the corresponding toc entry. If package hyperref is active in the document and has added hyperlinks to the TOC data, then these links are kept in the commands \etocname, \etocnumber and \etocpage (this last one will have a link only if hyperref was passed either option *linktoc=all* or option *linktoc=page*.)[13] In accordance with the hyperref native behavior, no link gets incorporated into \etocpage if the page number is empty.

## 11.1. The \etocskipfirstprefix and \etociffirst commands

The chosen subsubsection style made use of the command \etocskipfirstprefix, which instructs **etoc** to *not* use for the first item the specified ⟨*prefix*⟩ code.

The command \etociffirst{⟨*YES CODE*⟩}{⟨*NO CODE*⟩} is a more flexible way to customize the ⟨*prefix*⟩ (and ⟨*contents*⟩) specifications. It executes the ⟨*YES CODE*⟩ branch if this is the first unit at that level (inside a lower level) and the ⟨*NO CODE*⟩ if not. This is a robust command which survives to expansion (for example in an enumitem label).

The variant \etocxiffirst does the same, but is expandable.

## 11.2. The \etocnumber command

So far, our specifications would use the numbering generated by the enumerate environments, but of course we generally want the actual numbers as found in the .toc file. This is available via the \etocnumber command. To get the labels in the enumerate list to use it we can proceed with the syntax label=␣ from the package enumitem:

```
\etocsetstyle{section}
{\begin{enumerate}[label=\etocnumber]}
{\normalsize\bfseries\rmfamily\item}
{\etocname{} (page \etocpage)}
{\end{enumerate}}
```

Rather than just \etocnumber we then used something like \fbox{\etocnumber}. Note that \etocnumber is a robust command which explains why it can be used inside the label specification without needing an added \protect.

## 11.3. The \etocifnumbered switch

The \fbox would give an unaesthetic result in the case of an unnumbered section (which ended up in the table of contents via an \addcontentsline command).[14]

[13] As expected, in case of *linktoc=page*, only \etocpage is an hyperlink, not \etocname nor \etocnumber. See \etoclink on how to create hyperlinks with the entry target.

[14] as seen we use \fcolorbox rather than \fbox. Due to some redefinition made by package xcolor, had we used \fbox (and not used hyperref) we would have needed \protect\fbox.

The \etocifnumbered{⟨A⟩}{⟨B⟩} command executes ⟨A⟩ if the number exists, and ⟨B⟩ if not. So we use it in the code which was finally chosen for the section level:

```
\etocsetstyle{section}
{\begin{enumerate}[leftmargin=.75cm, label=\etocifnumbered
      {{\fboxrule1pt\fcolorbox{green}{white}{\etocnumber}}}{}]}
{\normalsize\bfseries\rmfamily\item}
{\etocname{} (page \etocpage)}
{\end{enumerate}}

\etocsetstyle{subsection}
{\begin{enumerate}[leftmargin=0cm, label=\etocnumber]}
{\normalfont \item}
{\etocname{} (p.~\etocpage)}
{\end{enumerate}}
```

If we had changed only the section level, and not the subsection level, an error on compilation would have occurred because the package style for subsections expects to start 'in vertical mode'. An additional \par token in the ⟨*contents*⟩ part of the section level would have fixed this: {...(page \etocpage)\par}.

The command \etocifnumbered is robust; \etocxifnumbered has the same effect but is expandable.

## 11.4. The \etocthename, \etocthenumber, and \etocthepage commands

It is sometimes desirable to have access to the name, number and page number without the hyperref link data: something similar to the starred variant of the \ref command, when package hyperref is used. For example one may wish to use the unit or page number in some kind of numeric context, or change its formatting. This is provided by \etocthename, \etocthenumber, and \etocthepage.

These commands are not "robust", in fact it is expected they will be often submitted to one expansion step so that their contents can easily be recovered and stored perhaps for delayed usage.

## 11.5. The \etoclink command

The command \etoclink{⟨*text*⟩} can be used in the line style specifications in a manner analogous to \etocname, \etocnumber and \etocpage. It creates a link (if hyperref is present[15]) whose target is the corresponding document unit and whose name is the given ⟨*text*⟩ mandatory argument.

Hence \etoclink{\etocthename} is under default conditions of hyperref like the original \etocname, because the latter is already hyperlinked. Under *linktoc=page* context \etoclink {\etocthename} adds the hyperlink which is missing from \etocname. Similarly under the default hyperref condition (i.e. *linktoc=section*) \etocpage is not an hyperlink, but one can use \etoclink{\etocthepage}.

The command \etoclink is robust.

[15] Prior to 1.1a, no such link was added if the .toc file entry was encountered with hyperref's option linktoc set to none.

## 11.6. The \etocthelinkedname, \etocthelinkednumber, \etocthelinkedpage and \etocthelink commands

The meanings of these commands can be stored for delayed usage. For example this is done in the examples with trees.

There has been a **breaking change** at 1.1a. Here is the behavior *prior* to this release:

- \etocthelinkedname and \etocthelinkednumber were hyperlinks only if hyperref was configured via linktoc=all or linktoc=section (the default),
- \etocthelinkedpage was an hyperlink only if hyperref was configured via linktoc=all or linktoc=page and the page number was not empty.

This behavior was coherent with the commands \etocname, \etocnumber, and \etocpage being the robust variants of \etocthelinkedname, \etocthelinkednumber, and \etocthelinkedpage.

At 1.1a it was decided that the commands should match their denominations.[16] So they are now *always* hyperlinks independently of linktoc hyperref option (\etocthelinkedpage has no hyperlink if the page number is empty, to match hyperref behavior):

- \etocthelinkedname and \etocthelinkednumber and \etocthelinkedpage are always (in presence of hyperref) hyperlinks (for \etocthelinkedpage the page number must not be empty).

A further command is provided: \etocthelink, which wraps[17] an hyperlink around its argument: \etocthelink{⟨*foo*⟩} hyperlinks an arbitrary text ⟨*foo*⟩ to the target sectioning unit in the document. The command \etoclink is its robust variant.

## 11.7. The \etocsetlevel command

One can inform **etoc** of a level to associate to a given sectioning command with \etocsetlevel. For example:

```
\etocsetlevel{cell}{0}
\etocsetlevel{molecule}{1}
\etocsetlevel{atom}{2}
\etocsetlevel{nucleus}{3}
```

In compatibility mode, it will be assumed that the commands \l@cell, \l@molecule, ..., have been defined somewhere either by the user or a class: doing only \etocsetlevel is not enough for the corresponding level to work out-of-the-box in compatibility mode.

However, if table of contents are never using compatibility mode, then all that matters is that the various line styles have been set. If, for example section is at level 1, then there is no need to do some \etocsetstyle{molecule}{..}{..}{..}{..} after \etocsetlevel{molecule}{1} if \etocsetstyle{section}{..}{..}{..}{..} has already been done (and it has been done by the package itself in its definition of its own line styles).

[16] To tell the whole truth, the author in refactoring the code completely at 1.1a was tricked by the names and forgot to read the old documentation so the new behavior was implemented and it was decided to keep the change.

[17] Prior to 1.1a, there was a link added only if hyperref option linktoc was not none.

The accepted levels (but see the frame below) run from -1 to 6 inclusive (also -2 with class memoir). Anything else is mapped to 6, which is a dummy level, never displayed. The package does:

```
\etocsetlevel{book}{-2}% (only with class memoir)
\etocsetlevel{part}{-1}
\etocsetlevel{chapter}{0}
\etocsetlevel{appendix}{0}% or 1 if document has no \chapter
\etocsetlevel{section}{1}
\etocsetlevel{subsection}{2}
\etocsetlevel{subsubsection}{3}
\etocsetlevel{paragraph}{4}
\etocsetlevel{subparagraph}{5}
```

**etoc** own custom styles are activated by \etocdefaultlines.

Boolean option **deeplevels** added at 1.2a has the effect of replacing 6 by 12 as the maximal numerical level (which, as has been said above, is never displayed). The value 6 (default) or 12 (if **deeplevels** is set to true) is held by \etocthemaxlevel. With **deeplevels** option one can for example do:

```
\etocsetlevel{subsubsubsection}{4}
\etocsetlevel{subsubsubsubsection}{5}
\etocsetlevel{subsubsubsubsubsection}{6}
\etocsetlevel{paragraph}{7}
\etocsetlevel{subparagraph}{8}
```

but it is up to user to actually define LaTeX commands such as \subsubsubsection, and also to provide, if **etoc** is left in "compatibility mode", the suitable \l@subsubsubsection et al. needed for TOC rendering. If you use \etocsetstyle (even only for one level name) though, which quits "compatiblity mode", it is not \l@subsubsubsection which needs definition, but \etocsetstyle{subsubsubsection}{..}{..}{..}{..} which has to have been used.

The numerical level assignments can be modified at anytime. See Part V for various applications of this technique.

## 11.8. Using enumerate or itemize environments for line styles

The code for the line styles of the \localtableofcontents of this part was already reproduced in section 11. It is very simple and uses enumerate environments for sections and subsections, then an "inline" paragraph style for subsubsection titles.

Actually, the very first version of **etoc** from 2012 was originally motivated by the aim to do exactly this kind of things, which necessitates to be aware of when for example after a series of subsections, a section line appears in the .toc file.[18] Indeed, this should trigger the emission of an \end{enumerate}.

[18] At the source code level, the legacy method dating back to the origins in 2012 was replaced by a completely new one at release 1.2.

With the {⟨*start*⟩} and {⟨*finish*⟩} arguments of \etocsetstyle, etoc provides an easy systematic interface to accomplish this kind of task.

But there are some limitations to the use of list environments for typesetting TOCs. One of them is intrinsic to the scope limitations created by the groups associated to the environments: the .toc file may contain, besides the information to be typeset in the TOCs, some other commands, such as language changing commands from babel, and some such commands do not expect to see their scope limited in this way by the presence of an environment (which will not be visible in the .toc file itself but enacted dynamically by the user-specified line styles).

The built-in "default line styles" provided with etoc (see section 12) do not make use of environments. Actually, in this user manual, only the table of contents at the start of this Part II, the subsection 41.1 (which is a TOC!) and examples from subsection 11.6 have their line styles expressed in terms of enumerate or itemize environments.

### 11.9. The\etocglobaldefs and \etoclocaldefs commands

In LaTeX the meaning of a command defined via \newcommand\foo{...} inside an environment (or group) vanishes from TeX's memory on exit from this environment (or group). At times however it is needed to make definitions with global scope, for this TeX has the primitive prefix \global.

By default etoc's definitions of \etocname etc... are local. This causes problems in certain contexts such as TOC as tables (section 39, section 45) and also with enumitem *inline* variants of its standard environments, because the command \item then closes a group (see subsection 11.6).

After \etocglobaldefs has been issued, the \etocname, \etocnumber and \etocpage will be defined during execution of \tableofcontents and \localtableofcontents with global scope. For normal use this is not necessary. It does not hurt either to activate it systematically.

To return to the default, which lets etoc only define them locally to the context in place where the \contentsline is encountered, execute \etoclocaldefs. Both \etocglobaldefs and \etoclocaldefs have an effect only locally to the environment or group where they are used.

## 12. The etoc fall-back line styles

This is a table of contents for the (few) subsections of this section. It carries the label toc:c



### 12.1. A demo of a TOC using \etocdefaultlines

These line styles were written at an early stage in the development of the package; although the next section explains how to customize the font choicess or vertical spaces, etc..., used by these line styles, most other changes would require copying them from the sources and modify them

directly. Admittedly they have been written at a rather scary low-TEX level, and will not serve as a very friendly starting point.

Activating their use is done via \etocdefaultlines, or \etoctoclines if the line styles have not been modified with \etocsetstyle. Sections and sub-sections are printed in essentially the same manner, except that the leading for sub-sections is a bit smaller (with document classes lacking a \chapter command, the sections are printed in bold typeface; this is the case in the present document). Sub-sub-sections are printed inline, in one paragraph, with no numbers or page numbers. This style was designed and tested with documents having lots of sub-sub-sections, and should be used on a two-column layout: it provides (only in that situation with many sub-sub-sections) a more compact presentation than what is achieved by the LATEX default.[19] On the other hand, used with a one-column layout, and with few sub-sub-sections, the style is a bit more spread out vertically than the LATEX default, sub-sections are not visually much different from sections (especially for document classes with a \chapter command), so the result is less hierarchical in appearance than in the LATEX default.

Let us typeset the global table of contents of the present document as if it had been done with a class having the \chapter command: we will print sections as chapters, and subsections as sections. We use \etocsetlevel for that, and also we need to change the font style of "sections" (which in truth are our subsections) to use not the bold but the medium series; we modify the \etocfontone command for that. Also we use dot leaders which are less spread out than in the package default.

```
\etocruledstyle[2]{\normalfont\normalsize\rmfamily\itshape
  \fbox{\parbox{.6\linewidth}{
      \leftskip 0pt plus .5fil
      \rightskip 0pt plus -.5fil
      \parfillskip 0pt plus 1fil This is the global table of
      contents on two columns, using \etoc default line styles, but with
      sections as chapters, and subsections as sections.
      }}}
\etocdefaultlines
\etocsetnexttocdepth{1}
\begingroup
\etocsetlevel{section}{0}
\etocsetlevel{subsection}{1}
\renewcommand*{\etocfontone}{\normalfont \normalsize}
\renewcommand*{\etoctoclineleaders}
        {\hbox{\normalfont\normalsize\hbox to 1ex {\hss.\hss}}}
\sloppy
\tableofcontents
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available.

etocsnippet-02.tex

[19] and there will never be a Part or Chapter entry alone at the bottom of a column or page (except if it has no sub-unit).

*This is the global table of contents on two columns, using etoc default line styles, but with sections as chapters, and subsections as sections.*

## 12.2. Customizing the etoc pre-defined line styles

We will simply list the relevant commands as defined in the package. Customizing them goes through suitable \renewcommands:

```
\newcommand*\etocfontminustwo{\normalfont \LARGE \bfseries}
\newcommand*\etocfontminusone{\normalfont \large \bfseries}
\newcommand*\etocfontzero{\normalfont \large \bfseries}
\newcommand*\etocfontone{\normalfont \normalsize \bfseries}
% (in classes with chapter, \etocfontone does not do \bfseries)
\newcommand*\etocfonttwo{\normalfont \normalsize}
\newcommand*\etocfontthree{\normalfont \footnotesize}

\newcommand*\etocsepminustwo{4ex plus .5ex minus .5ex}
\newcommand*\etocsepminusone{4ex plus .5ex minus .5ex}
\newcommand*\etocsepzero{2.5ex plus .4ex minus .4ex}
\newcommand*\etocsepone{1.5ex plus .3ex minus .3ex}
\newcommand*\etocseptwo{.5ex plus .1ex minus .1ex}
\newcommand*\etocsepthree{.25ex plus .05ex minus .05ex}

\newcommand*\etocminustwoleftmargin{1.5em plus 0.5fil}
\newcommand*\etocminustworightmargin{1.5em plus -0.5fil}
\newcommand*\etocminusoneleftmargin{1em}
\newcommand*\etocminusonerightmargin{1em}

\newcommand*\etocbaselinespreadminustwo{1}
\newcommand*\etocbaselinespreadminusone{1}
\newcommand*\etocbaselinespreadzero{1}
\newcommand*\etocbaselinespreadone{1}
\newcommand*\etocbaselinespreadtwo{1}
```

```
\newcommand*\etocbaselinespreadthree{.9}
\newcommand*\etoctoclineleaders
    {\hbox{\normalfont\normalsize\hbox to 2ex {\hss.\hss}}}
\newcommand*\etocabbrevpagename{p.~}  % initial of "page"
\newcommand*\etocpartname{Part}  % prior to 1.08b, was \partname
% but this didn't make sense e.g. with babel+frenchb whose \frenchpartname
% takes into account the value of the part counter.
\newcommand*\etocbookname{Book} % to be modified according to language
```

No customizing of the standard line styles is possible from within **etoc**. As already explained, when \etocstandardlines has been issued, the package just makes itself very discrete and acts only at the global level, and the TOC entries are (hopefully) formatted as would have happened in the absence of **etoc**.[20]

The \etocstandardlines compatibility mode will work also with sectioning commands made known to **etoc** via \etocsetlevel, under the condition of course that these sectioning commands are accompanied with all the relevant definitions for typesetting toc entries in the LaTeX default manner (existence of the macros \l@something . . .).

Using the command \etocsetstyle, be it in the preamble or in the body of the document, has the secondary effect of switching off the compatibility mode.

# 13. Summary of the main styling commands

## 13.1. Setting up local styles

\etocsetstyle{⟨*levelname*⟩}{⟨*start*⟩}{⟨*prefix*⟩}{⟨*contents*⟩}{⟨*finish*⟩}
\etocname, \etocnumber, \etocpage, \etocifnumbered{⟨*A*⟩}{⟨*B*⟩}
\etocthename, \etocthenumber, \etocthepage, \etoclink{⟨*linkname*⟩}

## 13.2. Setting up toc display styles

\etocmulticolstyle[⟨*number_of_columns*⟩]{⟨*heading*⟩}
\etoctocstyle[⟨*kind*⟩]{⟨*number_of_columns*⟩}{⟨*title*⟩}
\etoctocstylewithmarks[⟨*kind*⟩]{⟨*number_of_columns*⟩}{⟨*title*⟩}{⟨*mark*⟩}
\etoctocstylewithmarksnouc[⟨*kind*⟩]{⟨*number_of_columns*⟩}{⟨*title*⟩}{⟨*mark*⟩}
\etocruledstyle[⟨*number_of_columns*⟩]{⟨*title*⟩}
\etocframedstyle[⟨*number_of_columns*⟩]{⟨*title*⟩}
\etocsettocstyle{⟨*before_toc*⟩}{⟨*after_toc*⟩}

[20] with the KOMA-script classes, we noticed that \etocclasstocstyle was apparently needed for the KOMA options toc=left to be active at the level of the line entries.

## 13.3. Displaying tables of contents

```
\tableofcontents
\localtableofcontents
\etocmulticol[⟨number_of_columns⟩]{⟨heading⟩}
\etoclocalmulticol[⟨number_of_columns⟩]{⟨heading⟩}
\etocruled[⟨number_of_columns⟩]{⟨title⟩}
\etoclocalruled[⟨number_of_columns⟩]{⟨title⟩}
\etocframed[⟨number_of_columns⟩]{⟨title⟩}
\etoclocalframed[⟨number_of_columns⟩]{⟨title⟩}
    and their starred variants
```

## 13.4. Labels and references

The commands (starred or not) to actually display the table of contents can be followed with optional labels or references:

```
\tableofcontents \label{toc:here}
\tableofcontents \ref{toc:far}
\tableofcontents \label{toc:here} \ref{toc:far}
\localtableofcontents \label{toc:here}
\localtableofcontents \ref{toc:far}
\localtableofcontents \label{toc:here} \ref{toc:far}
    similarly with \etocmulticol etc . . .
```

`\localtableofcontents \ref{toc:far}` acts the same as `\tableofcontents \ref{toc:far}`. When re-displaying another toc, only its contents are transferred: both the line styles and the toc display style are the ones currently defined, not the ones from the cloned toc.

# Part III.

# Control of contents

*Here are some statistics for this part: it contains 10 sections and 8 subsections. The name of the first section is "*The `\tableofcontents` et al. commands*" and the corresponding number is "*14*". The name of the last section is "*Checking TOCs for emptiness*" and its number is "*23*". The name of the first subsection is "*The hyperref option *bookmarksdepth*" and the corresponding number is "*17.1*". The name of the last subsection is "*The `\etocifwasempty` command*" and its number is "*23.3*".*



## 14. The `\tableofcontents`, `\localtableofcontents` and `\localtableofcontentswithrelativedepths` commands

`\tableofcontents` can be used arbitrarily many times in the document. Styling either globally the TOC or its individual entries is customizable at any time in the document.

`\etoctableofcontents` is a synonym to **etoc**'s `\tableofcontents`. The `\tableofcontents` command reverts to its non-**etoc** definition if `\etockeeporiginaltableofcontents` is issued after loading the package. [21]

[21] This was added to fix a compatiblity issue with listings's `\lstlistoflistings`, as it needs the `\tableofcontents` macro to keep its original meaning.

**\localtableofcontents** will print local tables of contents: *i.e.* all sections and sub-units inside a given chapter, or all subsubsections and lower inside a given subsection, etc... (see also \etocsetnexttocdepth).[22,23]

**\localtableofcontentswithrelativedepth{⟨*number*⟩}** [24] can be used to override the document or current tocdepth setting (see section 16 for a discussion of tocdepth) to become relative to where the local TOC originates. For example, assuming the default numeric level assignments to standard sectioning units

```
\section{This is a section}
\localtableofcontentswithrelativedepth{+2}
```

will create a local table of contents taking into account the subsections and subsubsections inside this section, independently of what is the value of the tocdepth counter at this position in the document. If the numeric argument had been 3, the local TOC would have displayed also paragraphs. If the section had been a chapter, and again for a relative tocdepth of 2, the taken into account levels would have been sections and subsections.

## 15. Labeling and reusing elsewhere

**etoc** allows to typeset at some location a local table of contents which is defined elsewhere. For this, two simple steps:

1. insert \localtableofcontents at the distant place, and follow it by some \label{foo}.
2. insert \tableofcontents \ref{foo} (or \localtableofcontents \ref{foo}, it does the same) at the place where you want this distant table of contents to appear.
3. in step **1**, if you use \invisiblelocaltableofcontents in place of \localtableofcontents, there will be no typesetting at its place of definition.

At the place of use of \tableofcontents \ref{foo}, the layout and looks is entirely configurable locally. It may be completely different from how the same contents are rendered elsewhere by another \tableofcontents \ref{foo} or by the original label-decorated \localtableofcontents \label{foo}. The current value of the tocdepth counter is obeyed.

As an example the table of contents corresponding to Part II has been cloned here in a float which appears on the facing page. We used this:

```
\begin{figure}[ht!]
  \centering
  \begingroup
  % this is a KOMA-script specific customization
  \DeclareTOCStyleEntry[numwidth=2em,indent=0pt]{tocline}{section}
  \DeclareTOCStyleEntry[numwidth=3.2em,indent=2em]{tocline}{subsection}
```

[22] As is explained in section 15 the syntax allows to create somewhere a local table of contents and to display it at some other location either before or after its origin.
[23] As is explained in section 16 **etoc** allows at anytime to locally redefine the numeric levels associated to named ones, which brings great flexibility to achieve special effects, all done using only a single auxiliary file, the standard .toc file.
[24] Thanks to Tony Roberts for feature request.

**I am from far away**



```
  \etocstandardlines % <-- use the defaults from the document class
  \renewcommand{\etocbkgcolorcmd}{\color{green!5}}
  \renewcommand{\etocbelowtocskip}{0pt\relax}
  \fboxsep1ex
  \etocframedstyle [1]{\fbox{\makebox[.5\linewidth]{\etocfontminusone
        I am from \hyperref[toc:part:styling]{far away}}}}
  \etocsetnexttocdepth{subsection}
  \tableofcontents \label{toc:d} \ref{toc:part:styling}
  \endgroup
\end{figure}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-03.tex

In the above example, not only did we use \ref{toc:part:styling} to print here the distant

(local) table of contents which has been labeled toc:part:styling but we added a (possibly confusing) \label{toc:d}. This is done for the down-to-earth reason of being able to use, as we did in the previous paragraph, \vpageref{toc:d}. But if one wants to clone again the original local table of contents, one must reference its original label: \tableofcontents \ref {toc:part:styling}.[25] This original local table of contents is to be found on page 15.

## 16. A powerful functionality of etoc: the re-assignment of levels with \etocsetlevel

The intrinsic levels manipulated by etoc are numeric: from -2 (which corresponds to book in the memoir class) down (from the big to the small) to 5 (subparagraph). But the assignment of a numeric level to a given name can be modified at any time with the command \etocsetlevel{⟨*level_name*⟩}{⟨*number*⟩}. In conjunction with the use of the LaTeX tocdepth counter, this has powerful applications: ⟨*level_name*⟩ does not have to coincide with an actual document sectioning command, and etoc can be used to print arbitrary "lists of things", using no other auxiliary file than the .toc file. This is explained further in Part V.

It is often said that in the standard classes, the sectioning level of \part is 0 in the classes not having a \chapter command, and -1 in classes having a \chapter command. This is *correct* for what regards the *automatic numbering*, as is governed by the value of the secnumdepth counter; but it is *wrong* for what regards the effect of the tocdepth counter: setting the tocdepth to -1 in the article class just before \tableofcontents does *not* prevent Parts from appearing in the Table of Contents. One has to set it to -2 for that, whether in the article or in the book class.

The canonical levels, a priori known to etoc, are those of relevance to the **tocdepth** counter in the standard classes and are recapitulated in this table:

| | |
|---|---|
| (memoir class) book | -2 |
| part | -1 |
| chapter | 0 |
| section | 1 |
| subsection | 2 |
| subsubsection | 3 |
| paragraph | 4 |
| subparagraph | 5 |

With etoc, the user can easily print a local table of contents inside a given subsection, where subsubsections will be printed in the style of sections, paragraphs in the style of subsections, and subparagraphs in the style of subsubsections, if so desired. One can also decide to set everything to be at the level 6 (never displayed by etoc), except for example paragraphs, promoted to be at level 1, and then one obtains a nice table of contents of all

[25] Why does this author always give complicated examples rather than down-to-earth ones?

the paragraphs from the document! (tocdepth at least 1)[26]

## 17. The \etocsettocdepth and \etocsetnexttocdepth commands

The tocdepth counter has no bearing on what gets written to the .toc file; its action is only on the actual typesetting of the table of contents.[27] In the standard classes there is only one \tableofcontents possible, whereas with **etoc**, arbitrarily many are allowed, so one may change tocdepth to the appropriate value (which decides the finest sectioning level displayed) again and again each time a table of contents needs to be typeset.

**etoc** provides \etocsettocdepth{⟨*level*⟩} whose mandatory argument is either numeric (from -3 to 5) or a division name such as subsection or subsubsection or any name previously declared to **etoc** with \etocsetlevel (the keywords all and none are recognized, although not corresponding to a document division). This does the appropriate \setcounter{tocdepth}{numeric_level}.

As is explained in the next subsection, tocdepth is used by hyperref, and one must take steps to prevent its changes from influencing the bookmarks, too. So, **etoc** has \etocsetnexttocdepth{⟨*level*⟩} whose influence ceases immediately after the next table of contents. The package defines \invisibletableofcontents essentially as

\etocsetnexttocdepth{none}\tableofcontents

The simplest organization is probably to have after \begin{document} and before the first \tableofcontents a single instance of the \etocsettocdepth command, with argument the deepest level (or most commonly used deepest level) among the tables of contents of the document, and to use locally, where needed, \etocsetnexttocdepth before \tableofcontents or \localtableofcontents.

It is possible to use \etocsettocdepth inside the first argument of \etocsettocstyle (possibly in conjunction with checking the \etoclocaltop value, *which however will be up-to-date there only if \etoccheckssemptiness was executed*). There is no worry then about possible impact on hyperref bookmarks later on, because **etoc** always resets the tocdepth counter after typesetting a TOC to the value it had before it.

The macro \etocsetnexttocdepth works also if located in first argument of \etocsettocstyle, but there is no reason to use it there as \etocsettocdepth has no durable effet on the tocdepth counter if executed there.

Check \localtableofcontentswithrelativedepth for a simpler way to control the depth of local tables of contents. This has the advantage of working reliably whether or not

[26] and one should naturally not print this TOC of paragraphs in compatibility mode, which would insist on inserting a gigantic left margin.

[27] In the standard classes (at least), it also influences the \listoftables and \listoffigures, via \@dottedtocline.

the \etoccheckseemptiness is used.

### 17.1. The hyperref option *bookmarksdepth*

When modifying the counter tocdepth for the purposes of multiple uses of \tableofcontents or \localtableofcontents, one should be aware that package hyperref by default takes into account the *current* value of the tocdepth counter to decide whether the pdf file will contain a bookmark corresponding to sectioning commands encountered in the source file. Thus, one typically needs to reset tocdepth to its previous value after having temporarily modified it for a given table of contents.

Or, there is the *bookmarksdepth=n* option of package hyperref, with *n* the desired document bookmarks maximal depth, which can be numeric or the name of a level known to hyperref. This documentation previously passed bookmarksdepth=3 as option to hyperref, so even if tocdepth was left to 1 by inadvertance after printing a certain table of contents this did not modify the bookmark tree of the pdf file. Now that \etocsetnexttocdepth has been added to the package, we have used it systematically and there was no need for bookmarksdepth=3 anymore.

## 18. The \etocsettocdepth.toc and \etocimmediatesettocdepth.toc and commands

This command \etocsettocdepth.toc implements some functionality of tocvsec2[28], a package which however was incompatible with **etoc** (it can still be used for its secnumdepth-related commands, but its toc-related activities will get canceled by **etoc**) and more-or-less designed for a single table of contents.

The action of \etocsettocdepth.toc is totally different than the one of \etocsettocdepth. Rather than modifying the tocdepth counter immediately, it adds a line to the .toc file which, when executed inside a table of contents will enact this change.

The command \etocsettocdepth.toc, like \etocsettocdepth, accepts both numeric and named arguments. In the case of a named argument, the actual numeric value to be used is not yet decided at the time the .toc file is created; it will be the value currently specified for the named level at the time each table of contents (not having done \etocignoretoctocdepth) is typeset.

The tocdepth counter will never be set to a value finer than its initial value at the start of the table of contents: so adding commands \etocsettocdepth.toc in the document is a way to *restrict* locally the depth of the table of contents. For example to prevent inclusion in the tables of contents of the sub-sub-sections of a given chapter.

This gets executed in ALL tables of contents.

Also \etocimmediatesettocdepth.toc is provided. For explanations, refer to the discussion of \etocimmediatedepthtag.toc in the next section.

[28] I thank Denis Bitouzé for drawing my attention to the incompatibility of this package with **etoc**.

### 18.1. The \etocobeytoctocdepth and \etocignoretoctocdepth commands

So \etocignoretoctocdepth is provided to cancel the \etocsettocdepth.toc mechanism when needed; and \etocobeytoctocdepth will re-activate it. The package does initially \etocobeytoctocdepth.

## 19. The \etocdepthtag.toc, \etocimmediatedepthtag.toc and \etocsettagdepth commands

The command \etocdepthtag.toc allows to control dynamically the which contents end up included in the displayed TOCs (this documentation also decribed formerly a way using \etocsettocdepth.toc with some dummy level name, which got then set via \etocsetlevel according to what was locally needed, but it was too hacky and I am not sure if it was understandable).

It is used as \etocdepthtag.toc{⟨*tag_name*⟩}, where the ⟨*tag_name*⟩ is anything, and this will put the tag in the .toc file. When typesetting a TOC, one issues a series of commands \etocsettagdepth{⟨*tag_name*⟩}{⟨*level*⟩} where the ⟨*level*⟩ may be either numeric (from -3 to 5) or the name of a division unit known to **etoc**, or none or all. The effect of the tag inside the .toc file will then be to set the tocdepth counter to the desired value, in real time (this can not get finer than the initial value of tocdepth at the start of the TOC).

The added flexibility is thus that \etocsetlevel has not been used in a kind of hacky way, that one may use named level depths, and the keywords none and all.

As usual, once the tag depths have been set, they remain in effect until getting redefined or seeing their scope expire via the closing of a group or of a surrounding environment. For an example, see section 43.

When using \etocdepthtag.toc in combination with LaTeX's \include, data may not end up in the .toc file in the correct order. For example in this situation:

```
\clearpage % or anything ending up causing its presence here right before
           % the \etocdepthtag.toc
\etocdepthtag.toc{sometag}
\include{some file containing sections}
```

The tag will end up in the .toc file *after* all section headings from the included file. The cause is that LaTeX inserts immediately in the main auxiliary file a command to input the auxiliary file of the included file (which in turn, contains instructions to add data to the .toc file). But \etocdepthtag.toc does not internally use such immediateness, as it uses the same interface as \section and alike commands when they want to write extra data to the .toc file.

So[29] there is \etocimmediatedepthtag.toc which will force the tag to be written immediately to the .toc file (well, rather immediately to the .aux file, so before the inclusion of the auxiliary file of the included file).

One should not use this variant systematically. For example if your document looks like:

```
\clearpage
```

[29] Thanks to Norman Ramsey who reported this problem, together with a fix, in July... 2016. Sorry for long delay before updating **etoc** six years later...

```
\section{bbbb}

Some text

\etocdepthtag.toc{sometag}
\etocimmediatedepthtag.toc{someimmediatetag}
\section{cccc}

Some text

\end{document}
```

then the someimmediatetag will end up being inserted in .toc file *before* the bbbb section. This is because LaTeX's \section uses a *delayed* write, not an *immediate* one. And \etocdepthtag.toc wisely uses a *delayed* write.

As it seems very hard programmatically to identify automatically if the *immediate* variant of \etocdepthtag.toc should be used, the package provides two separate commands and it is up to user to make the correct choice.

### 19.1. The \etocobeydepthtags and \etocignoredepthtags commands

After \etocignoredepthtags, the .toc depth tags are ignored (but \etocdepthtag.toc still works). The package does initially \etocobeydepthtags which makes **etoc** react to the found tags in the .toc file.

## 20. Adding commands to the .toc file

We described above \etocsettocdepth.toc and \etocdepthtag.toc which both insert commands inside the .toc file. An even more general mechanism of adding "action tags" to the .toc file could be envisioned, but this would just be a wrapper for direct use of \addtocontents{toc}{\something}.

One should be cautious when adding in this way things to the .toc file. For example, inserting \addtocontents{toc}{\string\clearpage} just before a \part to fix the problem when some part entry (in the table of contents) is isolated at the bottom of one page, will cause problems with multiple TOCs: this \clearpage will be executed by **etoc** each time a \tableofcontents or \localtableofcontents command is encountered! The more prudent thing is to do rather: \addtocontents{toc}{\string\myclearpage}, to have a \let\myclearpage\relax at the top level of the document and to use where needed something like:

```
\let\myclearpage\clearpage
\tableofcontents
\let\myclearpage\relax
```

The memoir class has the command \settocdepth which writes a \changetocdepth command inside the .toc file. This will impact the typesetting by **etoc** of *all* tables of contents, with (possibly) unexpected results: imagine the document has \settocdepth{chapter} at some point

to avoid having the sections from subsequent chapters be listed in the main table of contents. Then a local table of contents in one of these chapters will print a title but will be without any entry.

As the memoir class by itself allows multiple`\tableofcontents` these issues already arise there, independently of **etoc**, see page 170 of the memoir manual.

For this specific issue, the commands `\etocsettocdepth.toc`, `\etocignoretoctocdepth` and `\etocobeytoctocdepth` are the way to go; or their variants `\etocdepthtag.toc` and `\etocsettagdepth`.

As an aside, any `\setcounter{tocdepth}{n}` command added directly to the `.toc` file will see its effect cease when the end of a table of contents is reached, as **etoc** executes a `\setcounter{tocdepth}{previous_value}` there, to reste the `tocdepth` counter to the value it had on entering the table of contents.

## 20.1. The hyperref option *hidelinks*

The colored links (and also the rectangle links) are a bit annoying when used in tables of contents, especially when the document uses **etoc** and has plenty of them! One may wish for having colored links, *except* for those within table of contents! Indeed, why would things in TOCs need to be either framed in rectangles or colored, when the user *already expects them to be links*?

I use the following trick: either in the preamble using `\AtBeginDocument`, or right after `\begin{document}`, I have the command

```
\addtocontents{toc}{\protect\hypersetup{hidelinks}}
```

> All TOCs typeset by **etoc** have their contents done within a group (as if enclosed in an environment). So the command `\hypersetup{hidelinks}` will be executed by *each* TOC, but its effect will be limited to that TOC.

I found out experimentally that the option `hidelinks` could indeed be set many times with `\hypersetup` (this is not the case of all hyperref options).

## 20.2. Disabling protrusion in all TOCs

If using microtype it looks like a generally advisable counsel to disable protrusion in particular for all TOCs (see however subsection 48.3 for further information if you don't want to do that). To achieve this simply add

```
\addtocontents{toc}{\protect\microtypesetup{protrusion=false}}
```

immediately after `\begin{document}` (or use `\AtBeginDocument`). We ended up doing this for the present document after checking for a few of our TOCs that it improved their looks. As **etoc** always encloses the typesetting of tables of contents inside scope limiting groups, the effect will be limited to tables of contents only. Notice that adding the above command to an existing document will have an effect only on second compilation.

# 21. The \etocsetlocaltop.toc and \etocimmediatesetlocaltop.toc commands

It is important to understand that \localtableofcontents works entirely from data *in the .toc file*. If the document, say with article class, contains starred sectioning commands, which are not accompanied by suitable \addcontentsline, then these units are completely transparent to \localtableofcontents:

- If \localtableofcontents is issued before \section*{Foo}, say locally to a \section, then the local TOC will include not only the subsections between the \section and the \section*{Foo} but also those following, and it will stop only at encountering a later \section or \part from the document's body.
- If the command is issued right after \section*{Foo} and the later was itself subsequent to a (numbered) \subsection, then **etoc** will think it must display a TOC local to the *subsection*.

There is the command \etocsetlocaltop.toc to insert into the .toc file a kind of "ghost" of a given sectioning unit. Here is an example:

```
\part*{Extra unnumbered part}
\etocsetlocaltop.toc{part}
\localtableofcontents
```

So with no \part heading inserted into the table of contents via an \addcontentsline, still \localtableofcontents will know it is local to a part. In this example the local contents will be delimited by the next numbered \part, or \part* with \addcontentsline, or also by a later, second, \etocsetlocaltop.toc{part}.

As a (counter)-example consider this document:

```
\documentclass{article}
\usepackage{etoc}
\begin{document}
\tableofcontents

\part*{A}
\etocsetlocaltop.toc{part}
\localtableofcontents

\section{I}

\section{II}

\part*{B}

\section{III}

\part*{C}

\section{IV}
\end{document}
```

It uses only \part*. Thanks to the \etocsetlocaltop.toc the \localtableofcontents knows it should report only sections. But the other \part* are invisible to it as nothing is recorded in the .toc file. So the local table of contents in this example will list *all* sections not only I and II. To fix thix one may e.g. insert another \etocsetlocaltop.toc{part}, this time after \part* {B} (or make this a numbered part, or use \addcontentsline for it).

The above document amended with added \etocsetlocaltop.toc{part} after each unnumbered part will thus have its main TOC without any Part heading, but each \part can show a correct \localtableofcontents. The simpler approach would be to use \addcontentsline with each unnumbered \part so that it ends up in the .toc file, but **etoc** is keen on allowing the most diverse point of views.

It should be stressed that the various \etocsetlocaltop.toc{⟨*sect. unit*⟩} do impact the global \tableofcontents: they really act like actual sectioning units, except for not inducing any typesetting. In usual document classes, this would appear to mean that they are completely transparent to the global \tableofcontents. Not the case with **etoc**, which adds a virtual assembly of levels: the .toc data originating in \etocsetlocaltop.toc{⟨*sect. unit*⟩} will trigger the execution of the {⟨*finish*⟩} parts of the line styles of finer sectioning units encountered before (either in the global \tableofcontents or in an active \localtableofcontents); and it triggers the {⟨*start*⟩} parts of the line styles of finer units encountered after it (again in the global \tableofcontents, but also in any \localtableofcontents which is already activated at a coarser lever).

Depending on how the toc line styles are configured this may translate into some visual effect; for example with the **etoc** own line styles the {⟨*start*⟩} and {⟨*finish*⟩} mostly insert penalties or vertical spaces.

It is a matter of debate if this is good design; a variant serving purely to influence boundaries of local table of contents with no collateral effects could be provided. And the name of the macro was perhaps not so well chosen as it suggests it acts as would such an hypothetical variant. In absence of feature requests we leave the matter standing for now.

Usage of \etocsetlocaltop.toc interacts with \etoccheckseptiness in the expected way: it modifies (as explained above) the selection made by \localtableofcontents, hence the decision whether this local TOC will end up empty or not.

There is also \etocimmediatesetlocaltop.toc. This may be useful in some very special circumstances involving \include. For related discussion see the documentation of \etocimmediatedepthtag.toc.

## 22. The \etoclocaltop command

Within either the TOC style (\etocsettocstyle) or the local title styles (\etocsetstyle), the control sequence \etoclocaltop is made equivalent for the duration of \localtableofcontents to a numeric (self-delimiting) denotation of the current top level.

Thus: it will in numeric contexts (\ifnum, \ifcase, ...) represent zero for a local TOC corresponding to chapter, or one if in a section, or two if in a subsection, etc..., assuming of course here that the default levels are obeyed (see section 16).

\etoclocaltop from inside the TOC heading (first argument of \etocsettocstyle) has the correct value *only under* \etoccheckssemptiness *regime*. Special circumstances correspond to some special values:

**-3** (-\thr@@)
: signals that **etoc** considers the local TOC to be "unknown"; this happens at the last local TOC, for the first LaTeX run after adding a new \localtableofcontents to the document. In doubt, **etoc** assumes the TOC will prove non empty, hence it prints (independently of whether the check for emptiness was activated or not) the heading as specified by \etocsettocstyle. Thus, check if \etoclocaltop gives -3 as a a ⟨*number*⟩ to detect that situation from within the first argument of \etocsettocstyle, if desired.

**-1000** (-\@m)
: is in case of a \localtableofcontents being considered "known" (although it may still refer to the data in the .toc file from the previous run) but without the check for emptiness having been executed.

**-10000** (-\@M)
: is the value when accessed from the title of a global TOCs.

When executed from within a local table of contents **line styles** (\etocsetstyle), \etoclocaltop always will hold the correct value, whether or not the emptiness check was executed.

For a global table of contents however, it will always keep the value -3.

Attention! \etoclocaltop is only to be queried not set.

# 23. Checking TOCs for emptiness

## 23.1. The \etoccheckssemptiness command

The user needs to issue \etoccheckssemptiness to tell **etoc** to check whether local tables of contents are empty and in case of emptiness to print nothing at all.[30] This can be useful to authors of LaTeX classes who for example wish to have a \chapter command doing systematically a \localtableofcontents, or for people producing files via automatic conversions and some of those might have sectioning commands and others not.

«Emptiness» means that no \contentsline command would get executed within the scope of the local table of contents — empty line styles by themselves do not make the TOC empty. **etoc** always executes the \etocaftertochook command; and the test for emptiness itself executes everything else found in the .toc file. See section 20 in this context.

1. the \etocifwasempty command discussed below can be used from inside \etocaftertochook, and even from inside \etocbeforetitlehook.

2. there is also \etocdoesnotcheckemptiness.

The suppression of the heading (more precisely of the toc display style elements) may be effective only for the final LaTeX runs. For example in the situation of a \tableofcontents \ref{foo} where the label foo is not yet recognized, the heading (but not the contents) is printed and

[30] Thanks to Paul Gaborit who asked for such a feature.

the TOC is declared non-empty. Or, if one adds a \localtableofcontents to a document, on the next run, the test for emptiness will in fact apply to the next one, and the last local TOC of the document will have its contents temporarily unknown to **etoc**, hence will be declared non empty, and the heading will be printed.

For a finalized document compiled with initially no auxiliary files, the first LaTeX run will declare all local TOCs non empty and print for each of them a heading (and no contents naturally). The second LaTeX run will then correctly decide which local TOC is empty or not.

## 23.2. The \etocnotocifnotoc command

The user can then extend the emptiness-checking to the global TOCs with \etocnotocifnotoc. May I respectfully give the advice then to rather do none of \usepackage{etoc} nor \tableofcontents? ;-). Well, there is always the case of batch conversions of documents having or not sectioning units.

## 23.3. The \etocifwasempty command

The command \etocifwasempty{⟨*YES*⟩}{⟨*NO*⟩} executes ⟨*YES*⟩ if the previous TOC was found to be empty and ⟨*NO*⟩ if its was not so. This may serve to act appropriately after a truly empty TOC. If \etoccheckssemptiness has not been issued, this conditional always executes the ⟨*NO*⟩ branch.

This command is robust, and \etocxifwasempty is its expandable version.

Do not forget the second argument: at least an empty pair of braces must be present.

This conditional may wrongly say that the local TOC is empty or not empty until LaTeX compilations stabilize. But if it says that a local TOC is empty, this does mean that **etoc** considered the just encountered local table of contents to be empty (for that run) and thus printed nothing (not even a \par).

# Part IV.

# Examples

*Here are some statistics for this part: it contains 10 sections and no subsection. The name of the first section is "*A first example*" and the corresponding number is "*24*". The name of the last section is "*One more example of colored TOC layout*" and its number is "*33*".*



To understand all code snippets in detail, one will need to have first browsed through section 11 and section 9.

## 24. A first example

This section is called "a first example" due to legacy reasons of the various defects of this documentation...

Let us present a "first example" (sort of) of specification for line styles:

```
\begingroup\parindent 0pt \parfillskip 0pt \leftskip 0cm \rightskip 1cm
\etocsetstyle {section}
              {}
              {\leavevmode\leftskip 0cm\relax}
              {\bfseries\normalsize\makebox[.5cm][l]{\etocnumber.}%
               \etocname\nobreak\hfill\nobreak
               \rlap{\makebox[1cm][r]{\mdseries\etocpage}}\par}
              {}
\etocruledstyle[1]{\bfseries \Large My first \etoc: TOC of
  \autoref{part:overview} (\nameref{part:overview})}
\tableofcontents \ref{toc:overview}
\endgroup
```

In the above verbatim there is a mysterious

```
\ref{toc:overview}
```

which will be commented upon later. But let use first see what this code produces (of course its

output depends on the contents of the present document, as applies to all other examples in this documentation).

**My first etoc: TOC of Part I (Overview)**



This author always complicates things, so the above example had one additional twist I now have to explain. The mysterious

```
\tableofcontents \ref{toc:overview}
```

means that the contents which are displayed are those of a local table of contents located somewhere else and labeled there via a

```
\label{toc:overview}
```

Turns out that this was done via a \localtableofcontents located at the start of Part I, here is how the code looked like overthere:

```
\part{Overview}
\etocdefaultlines
\etocsettocstyle{}{}
\localtableofcontents \label{toc:overview}
```

For more explanations refer to section 15. Notice in particular that there is no relation whatsoever between the line styles used for the original \localtableofcontents and those applying here to the cloned one. Actually the original one could even have been made invisible via usage of \invisiblelocaltableofcontents!

Regarding the line styles, we could have used those defined by etoc, which are activated via \etocdefaultlines, or the default document class styles which are activated by \etocstandardlines, but we were a bit more ambitious here and wanted to design our own. The technique is a simple one: each heading is in its own paragraph, which may extend on multiple lines; it is responsible for setting its own \leftskip.

Here is again the code used (now displayed more fully). Notice that we defined styles for sections, subsections, and subsubsections, but actually that Part I only has sections!

```
\begingroup\parindent 0pt \parfillskip 0pt \leftskip 0cm \rightskip 1cm
\etocsetstyle {section}
              {}
              {\leavevmode\leftskip 0cm\relax}
              {\bfseries\normalsize\makebox[.5cm][l]{\etocnumber.}%
               \etocname\nobreak\hfill\nobreak
               \rlap{\makebox[1cm][r]{\mdseries\etocpage}}\par}
              {}
\etocsetstyle {subsection}
```

```
                {}
                {\leavevmode\leftskip .5cm\relax }
                {\mdseries\normalsize\makebox[1cm][l]{\etocnumber}%
                 \etocname\nobreak\hfill\nobreak
                 \rlap{\makebox[1cm][r]{\etocpage}}\par}
                {}
\etocsetstyle {subsubsection}
                {}
                {\leavevmode\leftskip 1.5cm\relax }
                {\mdseries\normalsize\makebox[1cm][l]{\etocnumber}%
                 \etocname\nobreak\hfill\nobreak
                 \rlap{\makebox[1cm][r]{\etocpage}}\par}
                {}
\etocruledstyle[1]{\bfseries \Large My first \etoc: TOC of
  \autoref{part:overview} (\nameref{part:overview})}
\tableofcontents \ref{toc:overview}
\endgroup
```

etocsnippet-04.tex Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available.

The two commands used are `\etocsetstyle` for specifying the line styles, and `\etocruledstyle` for the TOC global style.

The `\rightskip` is shared by all, and creates space where the page numbers get printed. For an elaboration of this technique see the next section 25 as well as section 43 which provides a TOC with parts and paragraphs. Both allow multi-line headings and employ a technique for putting page numbers in the right margin which was inspired from what LaTeX2e's `\@dottedtocline` macro does.

## 25. A second example

This second example:

1. illustrates displaying subsections of a given section "horizontally" in one single paragraph,
2. does a selection of contents via the technique of *depth tags*, described in section 19.

Again the qualities of the author innovative pedagogy skills are well illustrated by the simplicity of the example.

# Contents

### Part II – The etoc styling commands

The code looks like this. For more explanations relative to depth tags, and especially how they were incorporated into the present document, see also section 43.

```
\begingroup
\newcommand*{\DotsAndPage}
{\nobreak\leaders\hbox{\bfseries\normalsize\hbox to .75ex {\hss.\hss}}%
         \hfill\nobreak
         \makebox[\rightskip][r]{\bfseries\normalsize\etocpage}\par}

\etocsetstyle {part}
{\parindent 0pt
 \nobreak
 \etocskipfirstprefix}
{\pagebreak[3]\bigskip}
{\large\rmfamily\bfseries\centering %\scshape
 \etocifnumbered{Part \etocnumber{} -- }{}\etocname\par}
{}

\etocsetstyle {section}
{\leftskip 0pt \rightskip .75cm \parfillskip-\rightskip
 \nobreak\medskip
 \etocskipfirstprefix}
{\leftskip 0pt \rightskip .75cm \parfillskip-\rightskip
 \pagebreak[1]\smallskip}
{\normalsize\rmfamily\bfseries %\scshape
```

```
 \etocnumber. \etocname\DotsAndPage }
{\parfillskip 0pt plus 1fil\relax }

\etocsetstyle {subsection}
{\leftskip1cm\rightskip .75cm \parfillskip 0pt plus 1fil\relax
 \nobreak\smallskip}
{}
{\footnotesize\sffamily\mdseries\itshape
 \etocname{} (\etocnumber, p. \etocpage). }
{\par\medskip}

\etocsettagdepth {preamble}   {none}
\etocsettagdepth {overview}   {none}
\etocsettagdepth {styling}    {subsection}
\etocsettagdepth {control}    {none}
\etocsettagdepth {examples}   {none}
\etocsettagdepth {advanced}   {none}
\etocsettagdepth {etocandworld}{subsection}
\etocsettagdepth {code}       {none}

\etocsettocstyle {\centering\LARGE\textsc{\contentsname}\par\nobreak\medskip}{}
\etocsetnexttocdepth {all} %  but depth tags will control the actual contents
\etocobeydepthtags % this is default anyhow, but may have been turned off
\tableofcontents
\endgroup
```

etocsnippet-05.tex Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

One last remark: the code above uses `\etocsetstyle` only for parts, sections and subsections. Non-styled levels would be displayed using fall-back defaults which are incorporated into the package code. Those fall-back defaults for `paragraph` and `subparagraph` display nothing at all (deliberately). So even if the `tocdepth` counter setting allowed it, and even if we had used

```
\etocsettagdepth {code}{paragraph}
```

(as that Part VII does contain `\paragraph`’s), no paragraph entry would have been displayed here.

## 26. A Beautiful Thesis example

Here is a relatively simple example of use of the package functionalities. Let us set up some line styles. We choose a style for sections and sub-sections which would be suitable for, respectively, sections and sub-sections in an average length memoir. The line style specifications have some redundancy for clarity, and do not care about what to do at possible page breaks. Also, they do not worry about potential multi-column use.

*My Beautiful Thesis*

**Chapter 1 \(local)tableofcontents 4**

```
\begingroup % we start a group to keep the style changes local
\newlength{\tocleftmargin}    \setlength{\tocleftmargin}{4cm}
\newlength{\tocrightmargin}   \setlength{\tocrightmargin}{1cm}

\etocsetstyle{section}              % will pretend to be a Chapter
{\addvspace{1ex}\parfillskip0pt
 \leftskip\tocleftmargin            % (already done in title)
 \rightskip\the\tocrightmargin plus 1fil
 \parindent0pt\color{cyan}}         % (already done)
{\bfseries\LARGE\upshape\addvspace{1ex}\leavevmode}
{\llap{Chapter\hspace{.5em}{\etocnumber}\hspace{.75cm}}\etocname
 \nobreak\hfill\kern1em\makebox[-\tocrightmargin][l]{\makebox[0pt]{\etocpage}}\par}
{}

\etocsetstyle{subsection}           % will pretend to be a Section
{}
{\mdseries\large\addvspace{.5ex}\leavevmode}
{\llap{\etocnumber\hspace{.75cm}}\textit{\etocname}%
 \hfill\makebox[-\tocrightmargin][l]{\makebox[0pt]{\etocpage}}\par}
{}

\def\tmptitle{My Beautiful Thesis}
\etocsettocstyle{\color{cyan}\parindent0pt \leftskip\tocleftmargin
  \leavevmode\leaders\hrule height 1pt\hfill\
  \huge\textit{\tmptitle}\par}{\bigskip}

\tableofcontents \ref{toc:overview}
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-06.tex

## 27. Testing the compatibility mode

As a further example we now print the local table of contents of section 9. First we will test the compatibility mode.[31] The original is the local table of contents of section 9, to which we allocated the label `toc:tocstyle`.

```
\begingroup % to keep in particular toc=left with local effect
\KOMAoptions{toc=left}
\etocclasstocstyle % necessary for the display to obey toc=left
\etocstandardlines % use the document class built-in TOC line styles
\tableofcontents \ref{toc:tocstyle}
\endgroup
```

### Contents



## 28. Another compatibility mode

As explained in section 6, the commands \etocstandardlines and \etocetoclocaltocstyle tell **etoc** to, essentially, act as an observer. And it starts in this state initially. The document class layout for the table of contents is then perfectly obeyed (well, hopefully). There is no way *if remaining in this compatibility mode* to customize this standard layout (change fonts, margins, vertical spacings, etc...) from within the package.

For customizing *while remaining in the compatibility mode*, use some package dedicated to this task; because **etoc** either is (temporarily perhaps) in compatibility mode with no customization on its part possible, or the user has specified the layout in \etocsetstyle commands (and \etocsettocstyle) and is supposedly in complete control.

Well, there is actually an alternative. It is possible to use the \etocsetstyle commands to recreate an artificial compatibility mode, in order to achieve effects like the following, all things being otherwise equal to the document class defaults:

1. get the hyperref link to encapsulate only the names, but not the numbers of each entry of the table of contents,
2. use the document class style for chapters and sections, but modify it only for subsections,
3. do either of the above only for some portions of the table of contents.

[31] the present document uses the scrartcl class, and we check here that the **etoc** compatibility mode does respect the customizing done via the class commands.

One only needs to use within the arguments of \etocsetstyle the LaTeX standard \l@chapter, \l@section, etc... re-constituting their arguments using \etocname, \etocnumber, \etocpage as one wishes. Here is an example. Include in the preamble:

```
\makeatletter
\newcommand{\MyLocalTOC}[1][section]{%
  \begingroup
  \etocsetstyle{section}{}{}
    {\l@section{\numberline{\etocnumber}\etocname}{\etocpage}}{}%
  \etocsetstyle{subsection}{}{}
    {\l@subsection{\numberline{\etocnumber}\etocname}{\etocpage}}{}%
  \etocsetstyle{subsubsection}{}{}
    {\l@subsubsection{\numberline{\etocnumber}\etocname}{\etocpage}}{}%
  % etc... if further sectioning units are needed
  %        (i.e. not excluded by tocdepth and actually there in document)
  % Here #1 defaults to section, meaning this is appropriate
  % for local TOC in a chapter
  \etocsettocstyle{\@nameuse{#1}*{Local contents}}
                  {}
  %
  \localtableofcontents
  \endgroup}
\makeatother
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-07.tex

Then use \MyLocalTOC in the document body. It is prepared for being local to a \chapter's as it typesets the heading of the TOC by default as un unnumbered section.[32]

One can add to the above arbitrary text formatting commands, for example one can replace \etocpage in the code above by \textcolor{blue}{\etocpage}.

Only pay attention to using \makeatletter/\makeatother as we are handling LaTeX macros with the dangerous sign @ in their names.

To give another example, one sees in article.cls the following definition:

```
\newcommand*\l@subsection{\@dottedtocline{2}{1.5em}{2.3em}}
```

The first argument is the level, the second the indent, and the third the numwidth (see the tocloft documentation). So if we issue in a document using the article class:

```
\makeatletter
\etocsetstyle{subsection}
    {]
    {}
    {\@dottedtocline{2}{1.5em}{2.3em}{\numberline{etocnumber}\etocname}{\etocpage}}
    {}
\makeatother
```

we then basically reconstitute the default rendering. Here is more careful code:

[32] Parts are handled somewhat differently according to whether one uses the standard or other classes; please check the source of these classes for what is to emulate here.

```
\makeatletter
\etocsetstyle{subsection}
    {]
    {}
    {\@dottedtocline{2}{1.5em}{2.3em}{\etocifnumbered{\numberline{etocnumber}}{}%
                                      \etocname}{\etocpage}}
    {}
\makeatother
```

Hence one can very easily without any (additional...) package modify the hard-coded indent `1.5em` and numwidth `2.3em`. But in general one has to do this in a synchronized way also for subsubsections, and for sections. The definition of `\l@section` in the `article` class source is a bit more complex.

Nevertheless this technique is probably the fastest (but see the example at start of section 11) to get going with **etoc** even if one is primarily interested only in its `\localtableofcontents`, as typesetting local tables of contents exactly as global tables of contents is not ideal. For example for a local TOC in a section, it looks appropriate to modify the above into

```
\makeatletter
\etocsetstyle{subsection}
    {]
    {}
    {\@dottedtocline{2}{0}{2.3em}{\etocifnumbered{\numberline{etocnumber}}{}%
                                      \etocname}{\etocpage}}
    {}
\makeatother
```

to cancel the indentation. One will have to keep the indents and numwidths in sync with similar changes to other line styles, if `\contentsline`'s of various levels are to be executed.

The number and the name of each entry are each separately an hyperref links, as is always the case with **etoc**, when not in compatibility mode. Replacing `\etocnumber` with `\etocthenumber` will give a TOC where the numbers are not links anymore, but the names still are. Or one may decide to use `\etocthename` and keep an hyperlinked number with `\etocnumber`.

For a more sophisticated example see section 44.

*Attention Please! The LaTeX kernel is moving towards adding* tagging *to the PDF, in a way mostly automated and transparent to user.* **etoc** *will in due time accompany that evolution but this may mean that it will use for its own the hooks that LaTeX will place for example in* `\@dottedtocline`*. So, if the user explicitly also requires usage of* `\@dottedtocline`*, this* may *mean that some tagging code would be executed twice, possibly causing some havoc.*

*My remark is* purely hypothetical*, as a.t.t.o.w. (2023/02/22) I have only started looking in the matter, and the LaTeX and hyperref TOC related changes have started being visible to developers only a few days ago.*

*It may be however, that activating tagging could mean that the simple-minded recycling techniques described in this section will not work. I guess* **etoc** *will always have the possibility to let the user specify that* **etoc** *should not take care itself of the tagging (which it has to do in general, because the LaTeX hooks are located in places such as* `\@dottedtocline` *which* **etoc** *does not execute, except if asked to do so as in the example above), so perhaps the techniques here will still work but require some* `\etocnotagging` *or some* noetoctagging *option to the* `\localtableofcontents` *or* `\tableofcontents` *commands.*

# 29. Emulating the book class

As explained in section 6: without explicit use of an `\etocsetstyle` command the package will leave to the document class the hand regarding the "toc line styles". It is sometimes asked by users (for example those using **etoc** for its `\localtableofcontents`) how to stay close to but not completely identical with the design implemented by the standard classes, such as book. I can recommend package tocloft for this, as it is compatible with **etoc** (see subsection 48.7) and thus **etoc** will obey the tocloft customizations (as long as no use has been made of `\etocsetstyle`). It is also possible to modify only the style for, say, sections and leave the parts, chapters, subsections as in the document class, via the technique from section 28.

But for complete control, here is a translation of the book class code into **etoc** lingua. It is then easy to modify the relevant lengths or adjust the used fonts. I thank Denis Bitouzé for prompting me to include this in the **etoc** manual, as it resulted from some conversation we had about this. The code is not 100% faithful to the book class, and particularly its rendering of (multi-line) non-numbered units differs (... I think, as I copied pasted as is the code from where I had stored it and did not do much thinking about it again). Some proficiency in low-level TEX and LATEX macros is needed to understand what the code says, but for modifying fonts or some lengths such in-depth understanding is not needed.

With some extra code one can *automatically adjust the widths* assigned to typesetting sectioning numbers in order to prevent overflows, even with for example XXXVIII; but this is a more advanced feature which I have moved to section 40.

First we set up some lengths. I use macro registers, not real LATEX lengths. When using `em`'s however, this means that one must pay attention to when the actual dimension assignment is made, as this will then depend upon the current font settings. In the code below, at the location where the `\TOCnumwidthB` and `\TOCnumwidthC` will be used, the `1em` from their specification will be matched to the normal medium series font, not the bold font; this is deliberate so that one can compare more readily with the other dimensions; besides, with the `\TOCcomputenumwidths` from section 40 these macros will actually hold a dimension using `pt` as dimensional unit.

```
% it will be easy to globally shift the TOC horizontally if needed
\def\TOCleftmargin     {0pt}
\def\TOCrightmargin    {2.55em}% like LaTeX's \@tocrmarg

% this is for dotted leaders
\newbox\TOCleaderbox
\def\TOCleaderboxwidth {0.7777em}% about like what standard classes do

% vertical spacing
\def\TOCverysmallvskip {0pt plus .2pt}
\def\TOCmedvskip       {1em plus 1pt}
\def\TOCbigvskip       {2.25em plus 1pt}

% the "numwidths" for typesetting the numbering of division units.
% I don't recall exactly how (and for which fonts) these figures were chosen.
% They quickly prove too small if using Roman numerals (as do too the book
% class defaults even though they are a bit larger).
\def\TOCnumwidthB {1.5em} % chapter
```

```
\def\TOCnumwidthC {2.278em}% section, I think default is 2.3em
\def\TOCnumwidthD {3.056em}% analog in standard class is 3.2em
\def\TOCnumwidthE {3.833em}% analog in standard class is 4.1em
\def\TOCnumwidthF {4.611em}% analog in standard class is 5em
\def\TOCnumwidthG {5.389em}% analog in standard class is 6em

% The code for the “global toc style”.

\newcommand*\TOCglobalstyle {%
\etocsettocstyle
   {\if@twocolumn \@restonecoltrue \onecolumn \else \@restonecolfalse \fi
    \parindent\z@ \leftskip\z@skip \rightskip \z@skip
    \setbox\TOCleaderbox\hbox to \TOCleaderboxwidth{\hss.\hss}%
    \chapter *{\noindent\kern\TOCleftmargin\relax % uses "pt"...
        \contentsname
        \@mkboth {\MakeUppercase \contentsname}{\MakeUppercase \contentsname}}%
    \rightskip    \TOCrightmargin\relax
    \parfillskip -\rightskip % or a smaller value if desired
    \leftskip     \TOCleftmargin \relax }
   {\if@restonecol \twocolumn \fi\cleardoublepage}%
%
\etocsetstyle{part}
 {}
 {\addpenalty {-\@highpenalty}%
  \addvspace \TOCbigvskip
  \leavevmode
   {\large \bfseries % use a group to limit font change
      \interlinepenalty\@M
      \etocifnumbered{\etocnumber\hspace{1em}}{}%
      \etocname
      \nobreak\hfil\makebox[-\parfillskip][r]{\etocpage}}\par
  \nobreak
  }
 {}
 {}%
%
\etocsetstyle{chapter}
 {\advance\leftskip\TOCnumwidthB\relax}
 {\addpenalty {-\@highpenalty }%
  \vskip \TOCmedvskip\relax
  \leavevmode
  {\interlinepenalty\@M
   \etocifnumbered
      {\llap{\makebox[\TOCnumwidthB][l]{\bfseries\etocnumber}}}
      {\advance\leftskip-\TOCnumwidthB\relax}%
   \bfseries\etocname
      \nobreak\hfil\makebox[-\parfillskip][r]{\etocpage}\par }%
  \penalty \@highpenalty
  }
 {}
 {\advance\leftskip-\TOCnumwidthB\relax}%
%
```

```
 \TOCsetlinestyle {section}      {\TOCnumwidthC}%
 \TOCsetlinestyle {subsection}   {\TOCnumwidthD}%
 \TOCsetlinestyle {subsubsection}{\TOCnumwidthE}%
 \TOCsetlinestyle {paragraph}    {\TOCnumwidthF}%
 \TOCsetlinestyle {subparagraph} {\TOCnumwidthG}%
}% end of \TOCglobalstyle

%The common code for line styles is abstracted into a macro:

\newcommand\TOCsetlinestyle [2]{% #1= unit, #2= numwidth as macro
\etocsetstyle{#1}
 {\advance\leftskip#2\relax}
 {\vskip \TOCverysmallvskip\relax
  \leavevmode
  {\interlinepenalty\@M
   \etocifnumbered
     {\llap{\makebox[#2][l]{\etocnumber}}}{\advance\leftskip-#2\relax}%
   \etocname
   \nobreak\leaders \copy\TOCleaderbox
   \hfil\makebox[-\parfillskip][r]{\etocpage}%
   \par }%
 }
 {}
 {\advance\leftskip-#2\relax}%
}
\makeatother
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-08.tex

Nota Bene: the code deliberately handles the non-numbered sectioning units differently from what the standard document classes `article`, `report`, book do, particularly regarding the alignment of multi-line headings.

The whole thing was encapsulated in `\TOCglobalstyle`, because we also want a `\TOClocalstyle` for local tables of contents which typically will want to use `\section*` rather than `\chapter*` and not insert page marks in the headers. The `\TOClocalstyle` is to be issued once, after the main document TOC, or rather before using `\localtableofcontents`. If one wants a full TOC at end of document one will naturally have to issue again `\TOCglobalstyle` there.

```
\makeatletter
\newcommand*\TOClocalstyle {%
\etocsettocstyle
    {\if@twocolumn \@restonecoltrue \onecolumn \else \@restonecolfalse \fi
     \setbox\TOCleaderbox\hbox to \TOCleaderboxwidth{\hss.\hss}%
     \parindent\z@
     \dimen@ 2.25em % for left indenting
     \section *{\kern\dimen@ % use of \dimen@ works here by sheer luck
       \contentsname
       % un-comment this if marks are wanted:
       %\@mkboth {\MakeUppercase \contentsname}{\MakeUppercase \contentsname}%
       }% end of \section
     \parskip \z@skip
```

```
     \vspace{-1.25\baselineskip}% somewhat ad hoc
     \leftskip  2.25em
     \rightskip 4.5em
     \advance\rightskip-\TOCrightmargin\relax
     \leavevmode\leaders\hrule\@height\p@\hfill\kern\z@\par
     \rightskip 4.5em
     \parfillskip -\TOCrightmargin\relax }
    {\nobreak\vskip-.5\baselineskip
     \leavevmode\leaders\hrule\@height\p@\hfill\kern\z@\par
     \bigskip
     \if@restonecol \twocolumn \fi }%
%
\etocsetstyle{section}
 {\advance\leftskip\TOCnumwidthC\relax}
 {\addpenalty \@secpenalty
  \etociffirst{}{\addvspace{\TOCmedvskip}}%
  \leavevmode
  {\interlinepenalty\@M
   \bfseries\etocifnumbered
      {\llap{\makebox[\TOCnumwidthC][l]{\etocnumber}}}
      {\advance\leftskip-\TOCnumwidthC}%
   \etocname\nobreak\hfil\makebox[-\parfillskip][r]{\etocpage}\par }%
  \penalty \@highpenalty }
 {}
 {\advance\leftskip-\TOCnumwidthC\relax}%
% the rest is identical with code for global tocs:
 \TOCsetlinestyle {subsection}   {\TOCnumwidthD}%
 \TOCsetlinestyle {subsubsection}{\TOCnumwidthE}%
 \TOCsetlinestyle {paragraph}    {\TOCnumwidthF}%
 \TOCsetlinestyle {subparagraph} {\TOCnumwidthG}%
}% end of \TOClocalstyle
\makeatother
```

etocsnippet-09.tex

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

As mentioned previously, this handles non-numbered (multi-line) sectioning units somewhat differently from what happens in the standard document classes.

For some reason this code has some hard-coded `2.25em` and `4.5em` which were not abstracted into macros or lengths. The code inserts horizontal rules above and below the TOC contents in a non-separable by pagebreak way.

See section 40 for more.

## 30. A framed display

We now opt for a “framed” style, using the package default line styles and some colors added (it has been put in a float which appears on the facing page).

```
\etocdefaultlines
\begingroup
\renewcommand{\etoccolumnsep}{2em}
```

```
\renewcommand{\etocinnerleftsep}{1.5em}
\renewcommand{\etocinnerrightsep}{1.5em}
% specify a background color for the toc contents
\renewcommand{\etocbkgcolorcmd}{\color{yellow!10}}
% set up the top and bottom rules
\renewcommand{\etoctoprule}{\hrule height 1pt}
\renewcommand{\etoctoprulecolorcmd}{\color{red!25}}
\renewcommand{\etocbottomrule}{\hrule height 1pt}
\renewcommand{\etocbottomrulecolorcmd}{\color{red!25}}
% set up the left and right rules
\renewcommand{\etocleftrule}{\vrule width 5pt}
\renewcommand{\etocrightrule}{\vrule width 5pt}
\renewcommand{\etocleftrulecolorcmd}{\color{red!25}}
\renewcommand{\etocrightrulecolorcmd}{\color{red!25}}
% use \fcolorbox to set up a colored frame for the title
\fboxrule1pt
\renewcommand{\etocbelowtocskip}{0pt\relax}
\etocframedstyle {\normalsize\rmfamily\itshape
  \fcolorbox{red}{white}{\parbox{.8\linewidth}{\centering
      This is a table of contents \`a la \etoc, but for
      the subsections and subsubsections of \autoref{sec:tocstyle}.
      As it is put in a frame, it has to be small enough to fit on
      one page. It has the label |toc:b|.}}}
\begin{figure}[ht!]
  \centering
\tableofcontents \label{toc:b} \ref{toc:tocstyle}
\end{figure}
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

etocsnippet-10.tex

*This is a table of contents à la etoc, but for the subsections and subsubsections of section 9. As it is put in a frame, it has to be small enough to fit on one page. It has the label toc:b.*

# 31. Another TOC with background color

Let us now try out some more sophisticated line styles. The display will use the \etocframedstyle package command, which requires that the produced table of contents fits on a single page. We wrap it up in a figure environment showing up on page 64.

This design uses the **etoc** 'framed' style with a background color. The frame borders have been set to have the same color as the one serving as background for the entire thing. It would be advantageous to use rather inside \etocsettocstyle commands from a package like tcolorbox as this allows sophisticated breakable boxes (with TikZ/pgf for decoration.)

The details of the line styles used here are a bit involved, they were written by the author at some early stage of this documentation and have only been slightly revised to use more LaTeX-commands and less TeX-primitives. Similar code is used also for this other toc.

```
\begin{figure}[htbp!]\centering
\colorlet{subsecnum}{black}
\colorlet{secbackground}{green!30}
\colorlet{tocbackground}{red!20!green!20}

\renewcommand{\etocbkgcolorcmd}{\color{tocbackground}}
\renewcommand{\etocleftrulecolorcmd}{\color{tocbackground}}
\renewcommand{\etocrightrulecolorcmd}{\color{tocbackground}}
\renewcommand{\etocbottomrulecolorcmd}{\color{tocbackground}}
\renewcommand{\etoctoprulecolorcmd}{\color{tocbackground}}

\renewcommand{\etocleftrule}{\vrule width 3cm}
\renewcommand{\etocrightrule}{\vrule width 1cm}
\renewcommand{\etocbottomrule}{\hrule height 12pt}
\renewcommand{\etoctoprule}{\hrule height 12pt}

\renewcommand{\etocinnertopsep}{0pt}
\renewcommand{\etocinnerbottomsep}{0pt}
\renewcommand{\etocinnerleftsep}{0pt}
\renewcommand{\etocinnerrightsep}{0pt}

\newcommand\shiftedwhiterule[2]{%
    \hbox to \linewidth{\color{white}%
    \hskip#1\leaders\vrule height1pt\hfil}\nointerlineskip
    \vskip#2}

\etocsetstyle{subsubsection}
{\etocskipfirstprefix}
{\shiftedwhiterule{\leftskip}{6pt}}
{\sffamily\footnotesize
  \leftskip2.3cm\hangindent1cm\rightskip.5cm\relax
  \makebox[1cm][l]{\color{subsecnum}\etocnumber}%
  \color{black}\etocname
  \nobreak\leaders\hbox to.2cm{\hss.}\hfill
  \rlap{\makebox[.5cm][r]{\etocpage\hspace{.1cm}}}\par
  \nointerlineskip\vskip3pt}
{}
```

```
\etocsetstyle{subsection}
{\etocskipfirstprefix}
{\shiftedwhiterule{1.5cm}{6pt}}
{\sffamily\small
  \leftskip1.5cm\hangindent.8cm\rightskip.5cm\relax
  \makebox[.75cm][l]{\color{subsecnum}\etocnumber}%
  \color{black}\etocname
  \nobreak\leaders\hbox to.2cm{\hss.}\hfill
  \rlap{\makebox[.5cm][r]{\etocpage\hspace{.1cm}}}\par
  \nointerlineskip\vskip3pt}
{}

\newcommand{\coloredstuff}[2]{%
            \leftskip0pt\rightskip0pt\parskip0pt
            \fboxsep0pt % \colorbox uses \fboxsep also when no frame!
       \noindent\colorbox{secbackground}
               {\parbox{\linewidth}{%
                    \vskip5pt
                    {\noindent\color{#1}#2\par}\nointerlineskip
                    \vskip3pt}}%
       \par\nointerlineskip}

\etocsetstyle{section}
{\coloredstuff{blue}{\hfil \bfseries\large Contents of Part One\hfil}}
{\vskip3pt\sffamily\small}
{\coloredstuff{blue}
    {\leftskip1.5cm\rightskip.5cm\parfillskip-\rightskip
     \makebox[0pt][r]{\makebox[.5cm][l]{\etocnumber}}%
     \etocname\nobreak\hfill\makebox[.5cm][r]{\etocpage\hspace{.1cm}}}%
 \vskip6pt}
{}

\etocframedstyle[1]{}
\tableofcontents \label{toc:floating} \ref{toc:part:styling}
\vspace{-\baselineskip}
\centeredline{|\tableofcontents \ref{toc:part:styling}|
(\emph{cf.} \hyperref[toc:clone]{this other toc})}
\end{figure}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-11.tex

The table of contents produced by this code appears on on the following page.

## 32. A (crazy) inline display

Let us construct some crazy inline display of the table of contents of this entire document. We will typeset the subsections as footnotes... This kind of style is suitable for a hyperlinked document, probably not for print! (although I like it, but my personal tastes in many matters do not seem to be widely shared).

## Contents of Part One



`\tableofcontents \ref{toc:part:styling}` (*cf.* this other toc)

Here is the inline table of contents. . **etoc (read this first):** *License.* **Overview:** *\(local)tableofcontents, \locallistof(figures|tables), The \etocsettocstyle command, The \etocsetstyle command, No auxiliary file is used beyond the TOC file, Compatibility mode, A list of the commands added at 1.2, A partial list of the package commands.* **The etoc styling commands:** *The \etocsettocstyle and related commands*[33]*, Starred variants and hooks, The \etocsetstyle and related commands*[34]*, The etoc fall-back line styles*[35]*, Summary of the main styling commands*[36]*.* **Control of contents:** *The \tableofcontents et al. commands, Labeling and reusing elsewhere, \etocsetlevel, The \etocsettocdepth and \etocsetnexttocdepth commands*[37]*, The \etocsettocdepth.toc command*[38]*, The \etocdepthtag.toc and \etocsettagdepth commands*[39]*, Adding commands to the .toc file*[40]*, The \etocsetlocaltop.toc command, The \etoclocaltop command, Checking TOCs for emptiness*[41]*.* **Examples:** *A first example, A second example, A Beautiful Thesis example, Testing the compatibility mode, Another compatibility mode, Emulating the book class, A framed display, Another TOC with background color, A (crazy) inline display, One more example of colored TOC layout.* **Advanced examples:** *The TOC of TOCs, Arbitrary "Lists Of…", \etoctoccontentsline, The TOC as a tree, The TOC as a molecule, The TOC as a TikZ mind map, The TOC as a (long) table, A TOC self-adjusting widths for its typesetting, Interverting the levels*[42]*, Displaying statistics, Using depth tags, Sections styling subsections, The TOC as a (long) table (alternative).* **etoc and the world:** *Constraints on the .toc file constitution, Compatibility with document classes*[43]*, Compatibility with other packages*[44]*, TEXnical matters.* **The code:** *Timestamp, Change history, Implementation.*

The code used:

```
\begingroup
\newsavebox{\forsubsections}
\etocsetstyle{part}{\upshape. \etocskipfirstprefix}
                   {. \upshape}
                   {\bfseries\etocname:~~}
                   {}
\etocsetstyle{section}{\itshape\etocskipfirstprefix}
                      {, }
                      {\mdseries\etocname}
                      {}
\etocsetstyle{subsection}
    {\begin{lrbox}{\forsubsections}\footnotesize\upshape\etocskipfirstprefix}
    {; }
```

[33] The \etocsettocstyle command (*The \etocarticlestyle, \etocbookstyle, and others commands, The \etocinline and \etocdisplay commands*); The \etocmulticolstyle, \etocmulticol, and \etoclocalmulticol commands; The \etoctocstyle command (*The \etoctocstylewithmarks command*); The \etocruledstyle, \etocruled and \etoclocalruled commands; The \etocframedstyle, \etocframed, and \etoclocalframed commands; Customizing the pre-defined toc display styles; Headings, titles, \etocoldpar, \etocinnertopsep.

[34] The \etocskipfirstprefix and \etociffirst commands; The \etocnumber command; The \etocifnumbered switch; The \etocthename, \etocthenumber, and \etocthepage commands; The \etoclink command; The \etocthelinkedname, \etocthelinkednumber, \etocthelinkedpage and \etocthelink commands; The \etocsetlevel command; Using enumerate or itemize environments for line styles; The\etocglobaldefs and \etoclocaldefs commands.

[35] A demo of a TOC using \etocdefaultlines; Customizing the etoc pre-defined line styles.

[36] Setting up local styles; Setting up toc display styles; Displaying tables of contents; Labels and references.

[37] The hyperref option *bookmarksdepth*.

[38] The \etocobeytoctocdepth and \etocignoretoctocdepth commands.

[39] The \etocobeydepthtags and \etocignoredepthtags commands.

[40] The hyperref option *hidelinks*; Disabling protrusion in all TOCs.

[41] The \etocchecksemptiness command; The \etocnotocifnotoc command; The \etocifwasempty command.

[42] All subsections of this document.

[43] Compatibility with the KOMA-script classes; Compatibility with the memoir class; Compatibility with beamer.

[44] Compatibility with babel; Compatibility with hyperref; Compatibility with microtype; Compatibility with multicol; Compatibility with tableof; Compatibility with tocbasic; Compatibility with tocloft; Compatibility with tocbibind; Compatibility with tocvsec2.

```
    {\etocname}
    {.\end{lrbox}\footnote{\unhbox\forsubsections}}
\etocsetstyle{subsubsection}
    { (\itshape\etocskipfirstprefix}
    {, }
    {\etocname}
    {\/\upshape)}
\etocsettocstyle{Here is the inline table of contents. }{.\par}
\tableofcontents \label{toc:crazyinline}
\endgroup
```

etocsnippet-12.tex Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

## 33. One more example of colored TOC layout

The command \etocframedstyle puts the title on the top rule in a centered position. This is not very convenient for this example so we included the title as part of the ⟨*start*⟩ code at section level, to get it *inside* the frame.

```
\begingroup
\definecolor{subsecnum}{RGB}{13,151,225}
\definecolor{secbackground}{RGB}{0,177,235}
\definecolor{tocbackground}{RGB}{212,237,252}

\renewcommand{\etocbkgcolorcmd}{\color{tocbackground}}
\renewcommand{\etocleftrulecolorcmd}{\color{tocbackground}}
\renewcommand{\etocrightrulecolorcmd}{\color{tocbackground}}
\renewcommand{\etocbottomrulecolorcmd}{\color{tocbackground}}
\renewcommand{\etoctoprulecolorcmd}{\color{tocbackground}}

\renewcommand{\etocleftrule}{\vrule width 1cm}
\renewcommand{\etocrightrule}{\vrule width .5cm}
\renewcommand{\etocbottomrule}{\hrule height 12pt}
\renewcommand{\etoctoprule}{\hrule height 12pt}

\renewcommand{\etocinnertopsep}{0pt}
\renewcommand{\etocinnerbottomsep}{0pt}
\renewcommand{\etocinnerleftsep}{0pt}
\renewcommand{\etocinnerrightsep}{0pt}

\newcommand\shiftedwhiterule[2]{%
    \hbox to \linewidth{\color{white}%
    \hskip#1\leaders\vrule height1pt\hfil}\nointerlineskip\vskip#2}

\etocsetstyle{subsubsection}{\etocskipfirstprefix}
  {\shiftedwhiterule{\leftskip}{6pt}}
  {\sffamily\footnotesize
     \leftskip2.5cm\hangindent1cm\rightskip1cm\noindent
     \hbox to 1cm{\color{subsecnum}\etocnumber\hss}%
     \color{black}\etocname\leaders\hbox to .2cm{\hss.}\hfill
```

```
     \rlap{\hbox to 1cm{\hss\etocpage\hskip.2cm}}\par
     \nointerlineskip\vskip3pt}
  {}

\etocsetstyle{subsection}{\etocskipfirstprefix}
  {\shiftedwhiterule{1.5cm}{6pt}}
  {\sffamily\small
     \leftskip1.5cm\hangindent1cm\rightskip1cm\noindent
     \hbox to 1cm{\color{subsecnum}\etocnumber\hss}%
     \color{black}\etocname\leaders\hbox to .2cm{\hss.}\hfill
     \rlap{\hbox to 1cm{\hss\etocpage\hskip.2cm}}\par
     \nointerlineskip\vskip6pt}
  {}

\newcommand{\coloredstuff}[2]{%
            \leftskip0pt\rightskip0pt\parskip0pt
            \fboxsep0pt % \colorbox uses \fboxsep also when no frame!
       \noindent\colorbox{secbackground}
               {\parbox{\linewidth}{%
                    \vskip5pt
                    {\noindent\color{#1}#2\par}\nointerlineskip
                    \vskip3pt}}%
       \par\nointerlineskip}

\etocsetstyle{section}
{\coloredstuff{white}
     {\hfil \hyperref[toc:b]{\bfseries\large I am a twin of
     that other TOC (click me!)}\hfil}}
{\vskip3pt\sffamily\small}
{\coloredstuff{white}
    {\leftskip1.5cm\rightskip.5cm\parfillskip-\rightskip
     \makebox[0pt][r]{\makebox[.5cm][r]{\etocnumber\hspace{.2cm}}}%
     \etocname\hfill\makebox[.5cm][r]{\etocpage\hspace{.2cm}}}%
 \vskip6pt }
{}

\etocframedstyle[1]{}
\tableofcontents \label{toc:clone} \ref{toc:tocstyle}
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-13.tex

The TOC has been put in a float which appears on the previous page. The coding is a bit involved[45] as it does not use any additional package. Also, it was written at some early stage and I have not revised it since.

A better solution would be to use some package to set up a background color possibly extending accross pages, as the framed style (which we used to get this background color) can only deal with material short enough to fit on one page.

Regarding colors, generally speaking all color commands inside **etoc** are initially defined to do nothing, and the choice to use or not colors is left to the user.

[45] and reveals the author's preference for the TEX syntax...

# Part V.

# Advanced examples

*Here are some statistics for this part: it contains 12 sections and no subsection. The name of the first section is "*The TOC of TOCs*" and the corresponding number is "*34*". The name of the last section is "*The TOC as a (long) table (alternative)*" and its number is "*45*". The name of the first subsection is "*All subsections of this document*" and the corresponding number is "*41.1*". The name of the last subsection is "*All subsections of this document*" and its number is "*41.1*".*



## 34. The TOC of TOCs

Here is the numbered and linked list of all tables of contents which are displayed within this document:[46] 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38. And to obtain it here we just wrote:

```
Here is the numbered and linked list of all tables of contents which are
displayed within this document: \tableofcontents.
```

The preparatory work was the following. First, we defined a counter `visibletoc` whose vocation is to get incremented at each displayed toc. **etoc** has its own private counter but it counts all TOCs, even those not displayed because the `tocdepth` value was `-2` or `-3`.

We could have added manually `\refstepcounter{visibletoc}` and `\label` commands at all suitable locations in the document source, and we would then have used here `\ref` commands, but this imposes heavy manual editing of the source.

There is a much better way: there is a hook `\etocaftertitlehook` and we told it to increment the `visibletoc` counter and to write a line to the `.toc` file, in a manner analogous to what sectioning commands such as `chapter`, `section`, or `subsection` do. As **etoc** increments

[46] The TOCs put in floats may change the order: the numbers are listed in the order the TOCs are typeset in the document; but the numbering itself is from the order of the TOCs in the *source* of this document...

its own private counter even before typesetting the title of a table of contents, this provides (most of the time) a better link destination than any counter manipulated from inside \etocaftertitlehook (for which the link would target the area just after the title). So, rather than including \refstepcounter{visibletoc} inside \etocaftertitlehook, we just put there \stepcounter{visibletoc} followed by the command \etoctoccontentsline{visibletoc}{\thevisibletoc}. This **etoc** command \etoctoccontentsline{⟨*level_name*⟩}{⟨*name*⟩} has the same effect as:

\addcontentsline{toc}{⟨*level_name*⟩}{⟨*name*⟩}

but its usefulness is to circumvent[47] the patching for automatic creation of bookmarks done to \addcontentsline by the hyperref package, as pdf bookmarks don't make much sense here (and would elicit a complaint of hyperref that the bookmark level is 'unknown').[48]

Finally, the preamble of the document did \etocsetlevel{visibletoc}{6}. The level 6 (or anything with a higher number) is ignored, even if tocdepth has value 10 for example; this is independently of whether **etoc** uses the document class default line styles or its own line styles, or the ones defined by the user with the \etocsetstyle command. So there is no need to worry that something could go wrong.

*The example actually uses \etocthemaxlevel, see the 1.2a change log entry.*

Then, only here we have set \etocsetlevel{visibletoc}{0}. And to display only this kind of entries we assign temporarily to part and chapter level 1 (or anything higher than zero) and set tocdepth to the value 0. We also did

\etocsetstyle{visibletoc}{\etocskipfirstprefix}{, }{\etocname}{}

which defines an inline display with the comma (plus space) as separator. Finally, as **etoc** issues \par automatically by default just before typesetting a table of contents, we used the command \etocinline (also known as \etocnopar) which turns off this behavior.

Here are the implementation details:

```
   < in the preamble >
\newcounter{visibletoc}
\renewcommand{\etocaftertitlehook}
   {\stepcounter{visibletoc}\etoctoccontentsline{visibletoc}{\thevisibletoc}}
\etocsetlevel{visibletoc}{\etocthemaxlevel}
\begin{document}
   < document body >
\subsection{Surprising uses of etoc}
\begingroup
    \etocinline
    \etocsetlevel{part}{1}
    % \etocsetlevel{chapter}{1} % (no chapters in scrartcl class)
    \etocsetlevel{visibletoc}{0}
    \etocsetstyle{visibletoc}
        {\etocskipfirstprefix}{, }{{\color{niceone}\etocname}}{}
    \etocsettocstyle{}{}     % don't set any title, rules or frame or multicol!
    \etocsetnexttocdepth{visibletoc} % display only the 'visibletoc' entries from .toc
```

[47] using \addtocontents rather than \addcontentsline

[48] The package provides a starred variant \etoctoccontentsline*, which does allow the creation of bookmarks and has a third mandatory argument which is the Level to be used by these bookmarks; depending on the context the starred as well as the non-starred variants may be profitably preceded by \phantomsection.

```
Here is the numbered and linked list of all tables of contents which are
displayed within this document: \tableofcontents.
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-14.tex

After \etocsetstyle{visibletoc}{..}{..}{..}{..}, all future TOCs (not in compatibility mode) will use the defined style for level 0 (which is normally the level for chapters). To keep these changes strictly local the simplest manner is to put everything inside a group.

The section 41 gives another use of the shuffling of levels.

## 35. Arbitrary “Lists Of...”, \etoctoccontentsline and \etocimmediatetoccontentsline

This idea of interverting the levels is very powerful and allows to let **etoc** display lists of arbitrary things contained in the document. All of that still using nothing else than the .toc file! Example: imagine a document with dozens of exercises, perhaps defined as \newtheorem{exercise}{}[section]. Let us explain how to instruct **etoc** to display an hyperlinked list of all these exercises. For this we put in the preamble: (but see 1.2a change log entry)

```
\newtheorem{exerci}{}[section]
   % the exercice number will be recoverable via \etocname: v--here--v
\newcommand*{\exercisetotoc}{\etoctoccontentsline{exercise}{\theexerci}}
\newenvironment{exercise}{\begin{exerci}\exercisetotoc}{\end{exerci}}
\etocsetlevel{exercise}{6}
```

In this way, \etocname will give the exercise number (but \etocnumber will be empty). Had we used instead

```
\newcommand*{\exercisetotoc}
   {\etoctoccontentsline{exercise}{\protect\numberline{\theexerci}}}
```

the exercise number would then have been available via \etocnumber, and \etocname would have been empty. It doesn’t matter which one of the two methods is used. The **etoc** command \etoctoccontentsline{..}{..} is provided as a substitute to \addcontentsline{toc}{..}{..}: this is to avoid the patching which is done by hyperref to \addcontentsline in its process of creation of bookmarks. If one wants to authorize hyperref to create bookmarks at a specific level ⟨*n*⟩, one can use (here with ⟨*n*⟩=2) the starred variant \etoctoccontentsline* which has an additional argument:

```
\newcommand{\exercisetotoc}{\etoctoccontentsline*{exercise}{\theexerci}{2}}
```

The counter exerci is already incremented by the exerci theorem environment, and provides the correct destination for the link added by package hyperref. The command \exercisetotoc adds for each exercise a line to the .toc file, corresponding to a fictitious document unit

with name 'exercise'. A four-column list, including the sections, can then be typeset with the following code:

```
  \etocsetnexttocdepth{2}     % sections are at level 1 and will show up
\begingroup
  \etocsetlevel{exercise}{2}   % but:
  \etocsetlevel{chapter}{3}    %     no chapters
  \etocsetlevel{subsection}{3} %     no subsections
  \etocsetlevel{part}{3}       %     no parts
  \etocsetstyle{exercise}{}{}  % \etocname = exercise number
    {\noindent\etocname\strut\leaders\etoctoclineleaders\hfill\etocpage\par}
    {\pagebreak[2]\vskip\baselineskip}
  \etocsetstyle{section}{}{}
    {\noindent\strut{\bfseries\large\etocnumber\hskip.5em\etocname}\par
     \nopagebreak[3]}{}
  \etocruledstyle[4]{\Large\bfseries List of the exercises}
  \setlength{\columnseprule}{.4pt}
  \tableofcontents
\endgroup
```

etocsnippet-15.tex

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available.

A related command \etocimmediatetoccontentsline (and its starred version) is also provided. For discussion and the meaning of "immediate", refer to the analogous case of \etocimmediatedepthtag.toc.

## 36. The TOC as a tree

Using tikz and the package forest we shall display the table of contents of Part II as a tree. The technique is to use the **etoc** modified command \tableofcontents not for typesetting, but to prepare a macro, or rather here a \toks variable, with all the instructions to be executed later. Leslie Lamport's book has no mention whatsoever of such token lists registers, and LaTeX gives the impression to not really expect the general user to ever hear about them (or delimited macros); this whole section and the next are thus for advanced users.

Putting the \etocnumber and \etocname commands in \treetok would be of no use: to which number or name would they then refer to, in a delayed execution?

We need to store, not the macro names, but the macro contents. And also we wish to maintain the correct hyperref hyperlinks. The commands \etocname, etc…, are robust, it is easier to work with \etocthelinkednumber, \etocthelinkedname, and \etocthelinkedpage which contain the same information in an easier accessible form.

*At 1.1a the commands \etocthelinkedname, etc…, are always providing an hyperlink, so it is not true that \etocname, etc…, are always simply their robust variants.*

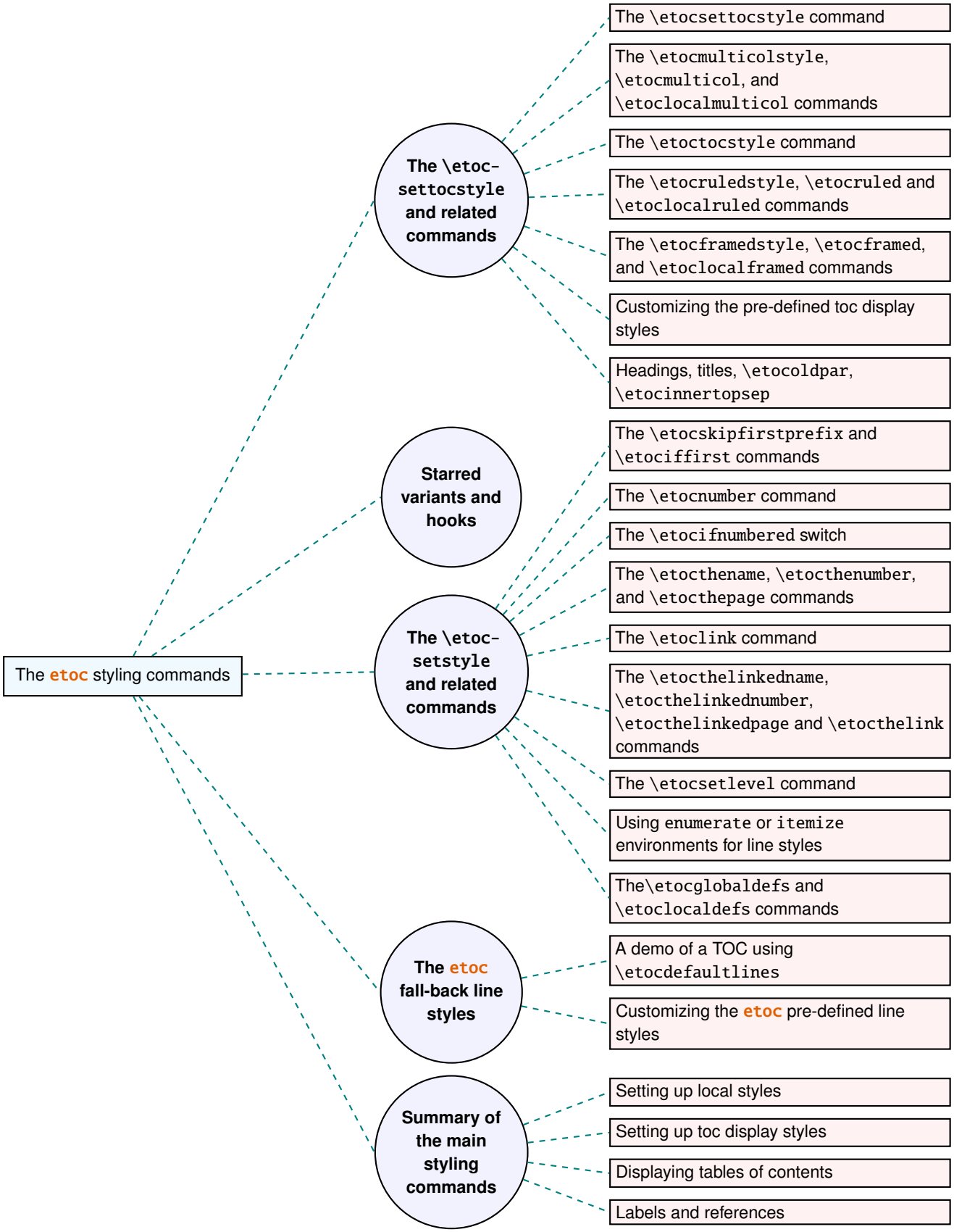
For this forest tree we have designed very special **etoc** styles for sections and subsections. They use a token list register called \treetok and a macro \appendtotok whose rôle is to append to a given token list variable the contents of a macro given as second argument. All this will happen in reaction to a \tableofcontents command, but *nothing* has yet been printed in the process.[49] This is the later job of a forest environment which will be given the contents of

[49] There is always a \par, which here is not a problem, but can be suppressed if need be via the command \etocinline or its synonym \etocnopar.

\treetok.

The resulting tree has been put in a float, which appears on the following page. Here is the code used for its production:

```
% \newtoks\treetok % put this (uncommented) preferably in the preamble
% \newtoks\tmptok  % (idem)
\begingroup
\newcommand*\appendtotok[2]{% #1=toks variable, #2=macro, expands once #2
  #1\expandafter\expandafter\expandafter
    {\expandafter\the\expandafter #1#2}}

\newcommand*\PrepareSectionNode{%
  \tmptok {\centering\bfseries}%
  \appendtotok\tmptok\etocthelinkedname
  \edef\foresttreenode{ [{\noexpand\parbox{2cm}{\the\tmptok}}]%
}

\newcommand*{\PrepareSubsectionNode}{%
  \tmptok {\raggedright}%
  \appendtotok\tmptok\etocthelinkedname
  \edef\foresttreenode{ [{\noexpand\parbox{6cm}{\the\tmptok}}]%
}

\etocsetstyle{section}
  {\etocskipfirstprefix}
  {\appendtotok\treetok{ ]}}
  {\PrepareSectionNode \appendtotok\treetok\foresttreenode}
  {\appendtotok\treetok{ ]}}

\etocsetstyle{subsection}
  {\etocskipfirstprefix}
  {\appendtotok\treetok{ ]}}
  {\PrepareSubsectionNode \appendtotok\treetok\foresttreenode}
  {\appendtotok\treetok{ ]}}

\etocsettocstyle
    {\treetok{[{\hyperref[part:styling]{The \etoc styling commands}}]}
    {\global\appendtotok\treetok{ ]}}

% forest does not like @\the\treetok if \treetok is empty. On first latex
% run, this will be the case because the TOC style defined above will not
% have been executed, as the label {toc:overview} does not refer to a valid
% TOC yet. So we must give a safe default value to \treetok
\treetok{[{run latex again}]}

\begin{figure}[htbp!]
\centering
     \etocsetnexttocdepth{subsection}
     \tableofcontents \label{toc:forest}\ref{toc:part:styling}
     \hypersetup{hidelinks}%
     \bracketset{action character=@}
% manual adjustments to fit the printed page
% \kern-1cm
```


The etoc styling commands
The \etoc-settocstyle and related commands
The \etocsettocstyle command
The \etocmulticolstyle, \etocmulticol, and \etoclocalmulticol commands
The \etoctocstyle command
The \etocruledstyle, \etocruled and \etoclocalruled commands
The \etocframedstyle, \etocframed, and \etoclocalframed commands
Customizing the pre-defined toc display styles
Headings, titles, \etocoldpar, \etocinnertopsep
Starred variants and hooks
The \etocskipfirstprefix and \etociffirst commands
The \etocnumber command
The \etocifnumbered switch
The \etocthename, \etocthenumber, and \etocthepage commands
The \etoclink command
The \etoc-setstyle and related commands
The \etocthelinkedname, \etocthelinkednumber, \etocthelinkedpage and \etocthelink commands
The \etocsetlevel command
Using enumerate or itemize environments for line styles
The\etocglobaldefs and \etoclocaldefs commands
The etoc fall-back line styles
A demo of a TOC using \etocdefaultlines
Customizing the etoc pre-defined line styles
Summary of the main styling commands
Setting up local styles
Setting up toc display styles
Displaying tables of contents
Labels and references

```
\noindent\kern-3cm
     \begin{forest}
                  for tree={anchor=center,child anchor=west,
                            grow'=east,draw,thick,
                            edge={draw,thick,dashed,color=teal}},
                  where={level()==1}{circle,thick,fill=blue!5,
                                     before computing xy={l=6cm}}{},
                  where={level()==2}{fill=red!5,
                                     before computing xy={l=6cm}}{},
                  rectangle, thick, fill=cyan!5, inner sep=6pt,
     @\the\treetok
     \end{forest}
\end{figure}
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-16.tex

Why \hypersetup{hidelinks}? as explained in subsection 20.1, I prefer the links in TOCs not to be colorized, nor framed, so this document inserts a command \hypersetup{hidelinks} in the .toc file. But at the time the \treetok contents are unpacked the \hyperlink commands originating in \etocthelinkedname, etc... will be executed in the normal environment for links (which, in this document, is to colorize them). Rather than having **etoc**'s code try to guess what the current "style" for links is (a concept not really provided by hyperref it seems) and store it in \etocthelinkedname, etc..., I opted for the simpler solution to leave it up to the user to recreate whatever conditions are desired. So here it is necessary to re-issue \hypersetup{hidelinks} in the figure environment.

There are some other examples in this documentation where \tableofcontents is used to prepare material for later typesetting:

- printing the statistics at the start of each Part (see section 42) is done using save boxes (so the problem of the appearance of the links does not arise then).
- typesetting of the TOC as a table without benefiting from \etocglobaldefs (see section 45).
- and the examples in the next section.

## 37. The TOC as a molecule

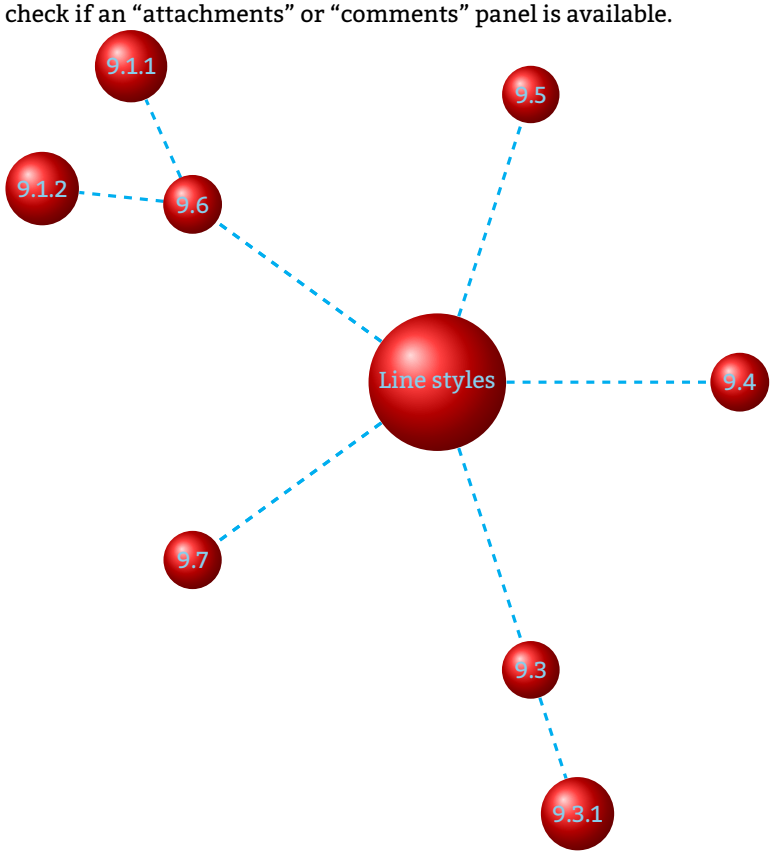
It is also possible to construct a TOC tree obeying the TikZ syntax for trees: but this is a more complicated task for the **etoc** line styles for reasons related to the way braces are handled by TEX (they need, when filling up the token list to be always balanced at each step, else complicated tricks must be employed.)

The simplest strategy is to allocate a token list (or use a macro) for each level used: we may need a \parttok, a \chaptertok, a \sectiontok and a \subsectiontok, to help in the task of filling up the total \treetok. As we are interested here in the table of contents of this (or another) document part, only a \sectiontok and a \subsectiontok will be needed.

```
% \newtoks\treetok % put this (uncommented) preferably in the preamble
% \newtoks\subsectiontok
% \newtoks\subsubsectiontok
% Attention: this code has been prepared only for subsections
% and subsubsections.

\newcommand*{\treenode}{}% only to make sure our \edef's do not overwrite
                         % an existing command

% expands 2nd argument (macro) and appends it to 1st argument (toks)
\newcommand*\appendtotok[2]{% #1=toks variable, #2=macro, expands once #2
  #1\expandafter\expandafter\expandafter
    {\expandafter\the\expandafter #1#2}}

% appends 2nd argument contents (toks) as child of first argument (toks)
\newcommand*{\appendchildtree}[2]{% token list t1 becomes: t1 child {t2}
   \edef\tmp{\the#1 child {\the#2}}%
   #1\expandafter{\tmp}%
}
% prepare the (hyperlinked) number in the "node (number)" shape
\newcommand*{\preparetreenode}{%
  \tmptok\expandafter{\etocthelinkednumber}% expanded once (needed)
  \edef\treenode{node {\the\tmptok}}%
}

\etocsetstyle{subsection}
  {\etocskipfirstprefix}
  {\appendchildtree\treetok\subsectiontok}
  {\preparetreenode
   \subsectiontok\expandafter{\treenode}}
  {\appendchildtree\treetok\subsectiontok}

\etocsetstyle{subsubsection}
  {\etocskipfirstprefix}
  {\appendchildtree\subsectiontok\subsubsectiontok}
  {\preparetreenode
   \subsubsectiontok\expandafter{\treenode}}
  {\appendchildtree\subsectiontok\subsubsectiontok}

\etocsettocstyle
  {\treetok{\node {\hyperref[sec:linestyles]{Line styles}}}}
  {\global\appendtotok\treetok{ ;}}

\centeredline{% from package centeredline (limits scope of \hypersetup)
   \etocsetnexttocdepth{subsubsection}
   \etocinline\tableofcontents \label{toc:molecule} \ref{toc:tocstyle}
   \hypersetup{hidelinks}%
   \begin{tikzpicture}
              [grow cyclic,
               level 1/.style={level distance=4cm,sibling angle=72},
               level 2/.style={level distance=2cm,sibling angle=60},
               every node/.style={ball color=red,circle,text=SkyBlue},
```

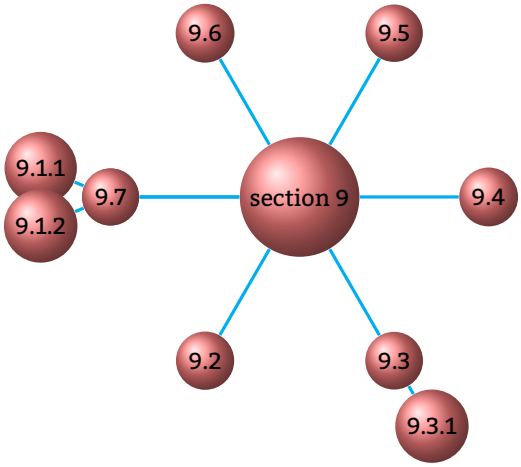
```
                edge from parent path={[dashed,very thick,color=cyan]
                            (\tikzparentnode) -(\tikzchildnode)}]
    \the\treetok
    \end{tikzpicture}%
}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-17.tex




The \tableofcontents command is executed prior to the tikzpicture environment which will actually typeset it via insertion of the accumulated data in the \toks register \treetok. This TikZ TOC is fully hyperlinked, like the previous Forest TOC.

Another example:




The above is the fully hyperlinked table of contents of section 9.

## 38. The TOC as a TikZ mind map

This is in the same spirit as the “molecule” example. The use of the $\varepsilon$-TEX primitive \unexpanded will simplify the code.


It is difficult to get everything to fit on one page. However \resizebox comes to the rescue. And it preserves hyperlinks. Nevertheless for this example I excluded some sections from the

display, using the technique of the etoc depth tags.

```
% \newtoks\treetok % done in preamble
% \newtoks\parttok
\newcommand*\partnode {}  % check with \newcommand we will not overwrite something
\newcommand*\childnode {}

\newcommand*\tmprotate {} % (idem)
\newcommand*\tmpoption {} %
\newcommand*\tmpstuff  {} %

\newcommand*\appendtotok[2]{% #1=toks variable, #2=macro, expands once #2
   #1\expandafter\expandafter\expandafter{\expandafter\the\expandafter #1#2}}

\newcommand*{\appendchildtree}[3]{%
% this is to construct "t1 child [#3]{t2}" from #1=t1 and #2=t2
% t1 and t2 are two toks variable (not macros)
% #3 = for example teal!60
   \edef\tmpstuff {\the#1 child [#3]{\the#2}}%
   #1\expandafter {\tmpstuff }%
}

\newcounter{partco}

% 1,2,3,4,5,... -> 1,2,3,1,2,3,1,2,3
\def\pseudomodthree #1{\numexpr #1 + 3 - 3*((#1+1)/3)\relax}

\etocsetstyle{part}
  {\etocskipfirstprefix}
% This updates the global tree with the data  from the previous
% part and all its children sections. Moved here because for some parts the
% sections are not displayed due to depth tags.
  {\ifnum\value{partco}=3
     \appendchildtree\treetok\parttok {branch color= green!50,level distance=10cm}%
   \else
\ifcase\pseudomodthree{\value{partco}}%
   \or \appendchildtree\treetok\parttok {branch color= teal!30}%   first
   \or \appendchildtree\treetok\parttok {branch color= yellow!80}% second
   \else\appendchildtree\treetok\parttok {branch color= green!50}% third and next ...
   \fi\fi
   }
  {\stepcounter{partco}%
% customize manually  some TikZ set-up (should be done inside the TikZ thing I guess)
   \def\tmpoption {}%
   \def\tmprotate {}% first
   %\ifnum\value{partco}=5 \def\tmprotate {[counterclockwise from =-40]}\fi
   %\ifnum\value{partco}=8 \def\tmprotate {[counterclockwise from =-50]}\fi
% define the part node
   \edef\partnode{node \tmpoption
                  {\unexpanded\expandafter{\etocthelinkednumber}.
                   \unexpanded\expandafter{\etocthelinkedname}}\tmprotate }%
% this is a starting point which will be filled it by the section children
   \parttok\expandafter{\partnode}}
```

```
  {\ifcase\pseudomodthree{\value{partco}}%
   \or \appendchildtree\treetok\parttok {branch color= teal!60}%   first
   \or \appendchildtree\treetok\parttok {branch color= yellow!80}% second
   \else\appendchildtree\treetok\parttok {branch color= green!50}% third and next ...
   \fi
   }

\etocsetstyle{section}
  {}
  {}
  {% define the section node
   \edef\childnode{child {node {\unexpanded\expandafter{\etocthelinkednumber}
                        \unexpanded\expandafter{\etocthelinkedname}}}}%
   % append it to the current \parttok
   \appendtotok\parttok\childnode
    }
  {}

\etocsettocstyle
  {\setcounter{partco}{0}%
   \treetok{\node [root concept]{\textbf{The \etoc documentation}}}}
  {\global\appendtotok\treetok{ ;}}
% The \global above is mandatory because etoc always typesets TOC inside a group

\etocsetnexttocdepth{section}
% use of depth tags to cut out sections for most parts, the sections
% are too numerous to fit well with the circular growth
\etocsettagdepth {preamble}   {part}
\etocsettagdepth {overview}   {part}
\etocsettagdepth {styling}    {part}
\etocsettagdepth {control}    {part}
\etocsettagdepth {examples}   {part}
\etocsettagdepth {advanced}   {part}
\etocsettagdepth {etocandworld}{part}
\etocsettagdepth {code}       {section}

\tikzset{
    branch color/.style={
        concept color=#1!white,
        every child/.append style={concept color=#1!white!30!white, font=\normalsize},
    }
}%

\begin{figure}[htbp!]
\etocobeydepthtags % obey the depth tags restrictions (which is default anyhow)
\tableofcontents\label{toc:mindmap}%
\centeredline{\resizebox{.85\paperwidth}{!}%
{\begin{tikzpicture}[mindmap,
                     grow cyclic,
                     text width=2cm,
                     align=flush center,
                     nodes={concept},
```

```
                    concept color=orange!60,
                    root concept/.append style={text width=4cm, font=\Large},
   level 1/.append style={level distance=5cm,sibling angle=45, text width=3cm},
   level 2/.append style={level distance=7cm,sibling angle=30, text width=3cm},
   level 1 concept/.append style={font=\normalsize},
                    ]
\the\treetok
\end{tikzpicture}}}
\end{figure}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-18.tex

The fully hyperlinked TOC appears on page 78.

An interesting alternative is to use **etoc** rather to convert the entire TOC into a TikZ tree (perhaps excluding some parts) and print it out to a file from which it can be recovered and manipulated directly by the author of the document. Things written to the `.log` file get broken into lines. Here is a technique to get non-broken output. Once the `\treetok` has been computed by **etoc** (as in the molecule example, or the current example), this demo will write it out to file with extension `.toctree`:

```
\newwrite\TOCasTree
\immediate\openout\TOCasTree=\jobname.toctree
\immediate\write\TOCasTree{\the\treetok}%
```

## 39. The TOC as a (long) table

It is possible to open a `tabular` (or `longtable`) in the title part of the TOC (first argument to `\etocsettocstyle`) and then close it after the contents (second argument to `\etocsettocstyle`), and specify in the line styles how to use the tabulation & and tabular end of row `\\`.

But there are some conditions and a few caveats:

1. it is **mandatory** to issue `\etocglobaldefs` for **etoc**'s definitions of `\etocname` et al. to have global scope, i.e. not be extinguished on encountering a &,
2. it is impossible to start one of the ⟨*start*⟩, ⟨*prefix*⟩, ⟨*contents*⟩ or ⟨*finish*⟩ specification with a sole `\hline`, *i.e.* one not preceded by a `\\`,
3. as is explained next, it is recommended to put the `\\` at the start of the ⟨*prefix*⟩ or ⟨*contents*⟩ specifications in order to close the *previous* row, rather that at the end with the idea to close the *current* row; and when the TOC is a partial one (a `\localtableofcontents`) this is (in almost all situations) mandatory.

Here is an example of a TOC as a `longtable`. Yes this is only *one* table! The code follows.

**TABLE OF CONTENTS**

## etoc (read this first)



## I. Overview



## II. The etoc styling commands

**III. Control of contents**



**IV. Examples**



**V. Advanced examples**

```
\begingroup
\etocglobaldefs % necessary for \etocname etc... to survive &
\makeatletter
% hack into longtable \hline to avoid annoying (here) stray lines at top
\def\LT@@hline{%
  \ifx\@let@token\hline
    \global\let\@gtempa\@gobble
    \global\let\@gtempb\@firstofone %%% ADDED
    \gdef\LT@sep{\penalty-\@medpenalty\vskip\doublerulesep}%
  \else
    \global\let\@gtempa\@empty
    \global\let\@gtempb\@gobble     %%% ADDED
```

```
    \gdef\LT@sep{\penalty-\@lowpenalty\vskip-\arrayrulewidth}%
  \fi
  \ifnum0=`{\fi}%
  \multispan\LT@cols
     \unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr
  \@gtempb{%                          %%% ADDED
  \noalign{\LT@sep}%
  \multispan\LT@cols
     \unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr
  \noalign{\penalty\@M}%
  }%                                  %%% ADDED
  \@gtempa}
\makeatother

% observe the locations of the \\
\etocsetstyle{part}
  {}
  {}
  {\\\hline\multicolumn{3}{c}{\bfseries\vrule height6ex depth3ex width0pt
                              \makebox[0pt]{\etocifnumbered{\etocnumber. }{}\etocname}}}
  {}

\etocsetstyle{section}
  {}
  {\etociffirst{\\\hline}{\\}}
  {\etocnumber&\etocname &\etocpage }
  {}

\etocsetstyle{subsection}
  {}
  {\\}
  {&\makebox[1cm][c]{\etocnumber}%
    \parbox[t]{\dimexpr6cm-\tabcolsep\relax}{\sloppy\itshape\etocname\strut}%
   &\itshape\etocpage }
  {}

\etocsettocstyle
    {\hypersetup{hidelinks}%
     \begin{longtable}{|>{\bfseries}c|p{7cm}|r|}
     \hline
     \multicolumn{3}{|c|}{\Large\bfseries\strut\strut TABLE OF CONTENTS}%
     }
    {\\\hline\end{longtable}}

\etocsetnexttocdepth {subsection}

\tableofcontents
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-19.tex

Examining the code above the reader will wonder why the \\ are always given first in

⟨*prefix+contents*⟩ and not, as is more intuitive, rather last. In some favorable cases (but almost never for local tables of contents) one may indeed construct TOC-as-tables with the \\ located at the end of the style specifications. The problem in the previous example was with the positioning of the \hline's.

Due to technical aspects of how TEX constructs alignments any definition or assignment done after an \\ starts a new row, and thus makes \hline an illegal token (this shows as a misplaced \noalign error.) Not only does **etoc** have to do such definitions to construct \etocname etc..., it is furthermore the case that some packages put things in the .toc file and as a result there is never any guarantee that between two \contentsline there will not be such a token like \relax which in the contexts of alignments forces TEX to start a cell and thus makes it impossible then to insert an \hline.

The safest way is thus to start with an \\ each line style specification in order to close the *previous* table row. We had a little problem with the fact that we wanted parts not only to have a rule above them (easy, they do \\\hline) but also below them: after each part there is a section, and it is these sections which are used to insert the missing \hline (this is done with the help of the \etociffirst conditional).

Last technical note: because we put the \\\hline inside the branches, there was no need to employ the expandable variants \etocxiffirst and \etocxifnumbered.

For the hardliner's old way see section 45.

Here is also a much simpler example. It is a local table of contents.

| Section title | number | page |
|---|---|---|
| \(local)tableofcontents | 1 | 4 |
| \locallistof(figures\|-tables) | 2 | 6 |
| The \etocsettocstyle command | 3 | 8 |
| The \etocsetstyle command | 4 | 9 |
| No auxiliary file is used beyond the TOC file | 5 | 9 |
| Compatibility mode | 6 | 10 |
| A list of the commands added at 1.2 | 7 | 11 |
| A partial list of the package commands | 8 | 14 |

```
\begin{center}
\etocsetstyle{section}
             {}
             {\etociffirst{\\\hline\hline}{\\\hline}}
             {\etocname & \etocnumber & \etocpage }
             {}

\etocsettocstyle
   {\hypersetup{hidelinks}\begin{tabular}{|>{\RaggedRight}p{4.5cm}|c|c|}\hline
    \multicolumn{1}{|c|}{\bfseries Section title}&
    \bfseries number&
    \bfseries page}
```

```
   {\\\hline\end{tabular}}

\etocglobaldefs % MANDATORY !!
\etocsetnexttocdepth{1}

\tableofcontents\ref{toc:overview}
\end{center}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available. etocsnippet-20.tex

## 40. A TOC self-adjusting widths for its typesetting

This is a continuation of section 29. The goal is to adjust automatically the “numwidths” used for typesetting the unit numbers in the (local) tables of contents.

```
\makeatletter
\newcommand*\TOCcompute@numwidths [2]{% #1=empty/"local", #2=minimal indent
  \begingroup
    \def\TOCnumwidthB {0pt}%
    \def\TOCnumwidthC {0pt}%
    \def\TOCnumwidthD {0pt}%
    \def\TOCnumwidthE {0pt}%
    \def\TOCnumwidthF {0pt}%
    \def\TOCnumwidthG {0pt}%
    \etocsetstyle{part}{}{}{}{}%
    \etocsetstyle{chapter}{}
      {\setbox0\hbox{\bfseries\etocthenumber\kern#2}}
      {\ifdim\wd0>\TOCnumwidthB\edef\TOCnumwidthB{\the\wd0}\fi}{}%
    \etocsetstyle{section}{}
      {\setbox0\hbox{\bfseries\etocthenumber\kern#2}}
      {\ifdim\wd0>\TOCnumwidthC\edef\TOCnumwidthC{\the\wd0}\fi}{}%
    \etocsetstyle{subsection}{}
      {\setbox0\hbox{\etocthenumber\kern#2}}
      {\ifdim\wd0>\TOCnumwidthD\edef\TOCnumwidthD{\the\wd0}\fi}{}%
    \etocsetstyle{subsubsection}{}
      {\setbox0\hbox{\etocthenumber\kern#2}}
      {\ifdim\wd0>\TOCnumwidthE\edef\TOCnumwidthE{\the\wd0}\fi}{}%
    \etocsetstyle{paragraph}{}
      {\setbox0\hbox{\etocthenumber\kern#2}}
      {\ifdim\wd0>\TOCnumwidthF\edef\TOCnumwidthF{\the\wd0}\fi}{}%
    \etocsetstyle{subparagraph}{}
      {\setbox0\hbox{\etocthenumber\kern#2}}
      {\ifdim\wd0>\TOCnumwidthG\edef\TOCnumwidthG{\the\wd0}\fi}{}%
    %
    \etocsettocstyle
     {}
     {\global\let\TOCnumwidthB\TOCnumwidthB
      \global\let\TOCnumwidthC\TOCnumwidthC
      \global\let\TOCnumwidthD\TOCnumwidthD
      \global\let\TOCnumwidthE\TOCnumwidthE
```

```
      \global\let\TOCnumwidthF\TOCnumwidthF
      \global\let\TOCnumwidthG\TOCnumwidthG
     }% make the found maximal widths have global scope
    \etocnopar
    \csname #1tableofcontents\endcsname
\typeout{Next TOCs will use \TOCnumwidthB\space for chapter number width}%
\typeout{Next TOCs will use \TOCnumwidthC\space for section number width}%
\typeout{Next TOCs will use \TOCnumwidthD\space for subsection number width}%
\typeout{Next TOCs will use \TOCnumwidthE\space for subsubsection number width}%
\typeout{Next TOCs will use \TOCnumwidthF\space for paragraph number width}%
\typeout{Next TOCs will use \TOCnumwidthG\space for subparagraph number width}%
  \endgroup % matches \begingroup at start of definition
}%
\newcommand*\TOCcomputenumwidths [1][0.5em]{%
  \TOCcompute@numwidths {}{#1}%
}%
\newcommand*\TOCcomputelocalnumwidths [1][0.5em]{%
  \TOCcompute@numwidths {local}{#1}%
}%
\makeatother
```

etocsnippet-21.tex Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

The optional parameter to `\TOCcomputenumwidths` specifies the minimal indent. In case nothing is numbered you may wish a higher value than `0.5em`. For each local table of contents to have its own width computations, the macro `\TOCcomputelocalnumwidths` is provided. As the code makes global assignments, either use (once) `\TOCcomputenumwidths` or do `\TOCcomputelocalnumwidths` for each local table of contents.

```
\TOCcomputelocalnumwidths  % may use optional argument to replace 0.5em
\localtableofcontents
```

Notes:

1. naturally these are only suggestions. For example one could put everything in single macros `\TOCtoc` and `\TOClocaltoc` to simultaneously compute the numwidths and then typeset the (local) table of contents.
2. if you want to adjust the `tocdepth` recall from subsection 17.1 that it influences hyperref hence you may need to use a group `\begingroup...\endgroup`. Or, one can use `\etocsetnexttocdepth`{⟨*level*⟩} but (with the code as here) this must then be issued twice, once for `\TOCcomputelocalnumwidths`, once for `\localtableofcontents`.
3. the bold font serves above for both chapter and section numwidth computations, but the code from section 29 uses `\bfseries` only in local TOCs. Thus the `\TOCcomputenumwidth` will set the parameter `\TOCnumwidthC` to a value slightly larger than needed in the main TOC. Hence the section style in `\TOCcompute@numwidths` should possibly insert the `\bfseries` in the box only after testing for the optional parameter `local`.

# 41. Interverting the levels

Let us display and count all subsections occurring in this document (see Part V for other uses of this technique):

```
\etocsetnexttocdepth{2}
\begingroup
\etocsetlevel{part}{3}
\etocsetlevel{section}{3}
\etocsetstyle{subsection}
    {\small\begin{enumerate}[itemsep=0pt,label=,leftmargin=0pt]}
    {\normalfont\bfseries\item}
    {\roman{enumi}. \mdseries\etocname{} (\etocnumber, p.~\etocpage)}
    {\end{enumerate}}
\renewcommand{\etoccolumnsep}{2.75em}
\renewcommand{\columnseprule}{1pt}
\etocmulticol[3]{\subsection{All subsections of this document}}
\endgroup
```

## 41.1. All subsections of this document

# 42. Displaying statistics

Each part of this document starts with a paragraph telling how many sections and subsections it has. Well, each one of this paragraph is a table of contents! We designed a macro `\thispartstats` to do that. It uses "storage" boxes to keep the information about the first and last section or subsection. Using boxes is the simplest manner to encapsulate the hyperref link for later use (whether there is one or none). However, one cannot modify then the font or the color. If such a need arises, one must switch from using boxes to using macros, and store the hyperref data for later use as was done in the code presented in section 37. We present also this second method.

But first, the code of `\thispartstats`:

```
\newsavebox\firstnamei  \newsavebox\firstnumberi
\newsavebox\lastnamei   \newsavebox\lastnumberi
\newsavebox\firstnameii \newsavebox\firstnumberii
\newsavebox\lastnameii  \newsavebox\lastnumberii
\newcounter{mycounti}   \newcounter{mycountii}
\newcommand*{\thispartstatsauxi}{} \newcommand*{\thispartstatsauxii}{}
\newcommand*{\oldtocdepth}{}
\newcommand*{\thispartstats}{%
  \setcounter{mycounti}{0}%
  \setcounter{mycountii}{0}%
  \def\thispartstatsauxi{%
        \sbox{\firstnamei}{\footnotesize\etocname}%
        \sbox{\firstnumberi}{\footnotesize\etocnumber}%
        \def\thispartstatsauxi{}}%
  \def\thispartstatsauxii{%
        \sbox{\firstnameii}{\footnotesize\etocname}%
        \sbox{\firstnumberii}{\footnotesize\etocnumber}%
        \def\thispartstatsauxii{}}%
  \begingroup
  \etocsetstyle{subsection} {} {}
    {\thispartstatsauxii
     \stepcounter{mycountii}%
     \sbox{\lastnameii}{\footnotesize\etocname}%
     \sbox{\lastnumberii}{\footnotesize\etocnumber}} {}%
  \etocsetstyle{section} {} {}
    {\thispartstatsauxi
     \stepcounter{mycounti}%
     \sbox{\lastnamei}{\footnotesize\etocname}%
     \sbox{\lastnumberi}{\footnotesize\etocnumber}}
    {{\footnotesize\itshape
    Here are some statistics for this part: it contains \arabic{mycounti}
    section\ifnum\value{mycounti}>1 s\fi{} and \arabic{mycountii}
    subsection\ifnum\value{mycountii}>1 s\fi. The name of the first section is
    \unhbox\firstnamei{} and the corresponding number is \unhbox\firstnumberi.
```

```
    The name of the last section is \unhbox\lastnamei{} and its number is
    \unhbox\lastnumberi. The name of the first subsection is \unhbox\firstnameii{}
    and the corresponding number is \unhbox\firstnumberii. The name of the last
    subsection is \unhbox\lastnameii{} and its number is \unhbox\lastnumberii.\par}}%
  \etocinline % cancels the automatic \par automatically before the TOC
  \etocsettocstyle {}{}
  \etocsetnexttocdepth{2}%
  \localtableofcontents  % to be used at the top level of a Part.
  \endgroup
}
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-22.tex

And now, the variant with macros rather than boxes (this variant as it stands here is for using within a section).

```
\makeatletter
\newcommand*\firstsubname    {}  \newcommand*\lastsubname    {}
\newcommand*\firstsubnumber {}  \newcommand*\lastsubnumber  {}
\newcommand*\thisspecialstatsaux{}
\newcommand*{\thisspecialstats}{%
  \setcounter{mycounti}{0}%
  \def\thisspecialstatsaux{%
         \let\firstsubname\etocthelinkedname
         \let\firstsubnumber\etocthelinkednumber
         \def\thisspecialstatsaux{}}
  \begingroup
  \etocsetstyle{subsection} {} {}
    {\thisspecialstatsaux
     \stepcounter{mycounti}%
     \let\lastsubname\etocthelinkedname
     \let\lastsubnumber\etocthelinkednumber }
    {Here are some statistics for this section. It contains \arabic{mycounti}
      subsections. The name of its first is \emph{\firstsubname{}} and the
      corresponding number is {\firstsubnumber}. The name of the last
      subsection is \emph{\lastsubname{}} and its number is {\lastsubnumber}.}%
   \etocsettocstyle {}{}
   \etocinline
   \etocsetnexttocdepth {1}%
   \localtableofcontents % to be used within a section
  \endgroup
}
\makeatother
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-23.tex

## 43. Using depth tags

**etoc** provides via the concept of *depth tags* the possibility to decide that, for example, a table of contents will display sections and subsections in a given chapter or part, but only sections, or

nothing at all, for the other chapters (or parts) included in the given table of contents. You will find more explanations at section 19

For this the user adds to the source usage in the document body of \etocdepthtag.toc commands, located right before the \part, or \chapter, or whatever division headings whose control is desired, for example

```
\etocdepthtag.toc{specialchapter}
```

and then when using \tableofcontents or \localtableofcontents, user will add some command such as

```
\etocsettagdepth{specialchapter}{section}
```

which will have the effect that only sections but no subsections of that chapter get displayed in that (possibly local, here to a Part) table of contents. Actually, one also has to add a depth tag at end of the desired scope of first one, something such as

```
\etocdepthtag.toc{endofspecialchapter}
```

and one will then issue

```
\etocsettagdepth{endofspecialchapter}{all}
```

to left further chapters again include in the so-configured table of contents all of their sub-units (as limited by current tocdepth). The simplest way thus is, if one decides to add a depth tag for one part or chapter to also add depth tags to all other parts, respectively chapters, and configure them appropriately.

Think of this as a way to add *inside* the .toc file some changes to the tocdepth counter, except that those changes are dynamically configurable, and also the whole thing can be ignored if \etocignoredepthtags is used.

As an example, one can imagine a document with various Parts, each part displaying a table of contents of the *complete* document, but which shows regarding other parts only their title and no finer division, whereas for the Part where it is located it will show all levels allowed by tocdepth. This is a variant of a "local" table of contents, which does display things external to the Part where it is located, but in a more limited way than regarding that Part itself.

The commands \etocobeydepthtags (which is default behavior) and \etocignoredepthtags can be used to control which TOC obey depth tags. If generally they should not, issue once \etocignoredepthtags in preamble, and then use \etocobeydepthtags for specific TOCs, either inside a scope-limiting group, or re-issue \etocignoredepthtags after each such (local or not) table of contents.

Here is a demonstration using the contents of this documentation. We want a TOC which will have a heading for each \part, with two Parts being handle especially: the Part III will show all its divisions down to subsubsection (if present) and the Part VII will even show paragraphs (turns out that this document used there some \paragraph mark-up). To achieve this we added various \etocdepthtag.toc commands inside the source of this document prior to each \part command. It only remains now to set appropriately the depth of each such depth tag, which will be done like this:

```
\etocsettagdepth {preamble}    {part}
\etocsettagdepth {overview}    {part}
\etocsettagdepth {styling}     {part}
\etocsettagdepth {control}     {subsubsection}
```

```
\etocsettagdepth {examples}    {part}
\etocsettagdepth {advanced}    {part}
\etocsettagdepth {etocandworld}{part}
\etocsettagdepth {code}        {paragraph}
```

As said, we want here to display even paragraph entries in a specific part. The standard document class paragraph TOC line styles give too much vertical spacing in this context. So we cook up our own, quickly designed, line styles, a bit like what we did for section 24, but with a way (see the command `\EndParWithPageNumberInMarginAndLeaders`) to put page numbers on the right which is more like the method used by LaTeX2e's `\@dottedtocline`; headings occupying more than one line will now wrap in a way achieving some hanging indentation relative to the entry number (but there is no such long heading in this example).

## A TOC using depth tags

```
\etocsetnexttocdepth {all}
\begingroup
\parindent 0pt \leftskip 0cm \rightskip .75cm \parfillskip -\rightskip
\newcommand*{\EndParWithPageNumberInMargin}
    {\nobreak\hfill
     % I initially added \etocnoprotrusion after \etocpage, but
     % finally switched to using monospace font for page number,
     % and it seems to have deactivated the protrusion anyhow.
     \makebox[0.75cm][r]{\mdseries\normalsize\ttfamily\etocpage}%
     \par}
\renewcommand*\etoctoclineleaders
    {\hbox{\normalfont\normalsize\hbox to .75ex {\hss.\hss}}}
\newcommand*{\EndParWithPageNumberInMarginAndLeaders}
    {\nobreak\leaders\etoctoclineleaders\hfill
     \makebox[0.75cm][r]{\mdseries\normalsize\ttfamily\etocpage}%
     \par }
\etocsetstyle {part}
              {}
              {\leavevmode\leftskip 1cm\relax}
              {\bfseries\large\llap{\makebox[1cm][r]{\etocnumber\ \ }}%
               \etocname\EndParWithPageNumberInMargin\smallskip}
              {}
\etocsetstyle {section}
              {}
              {\leavevmode\leftskip 1.75cm\relax}
              {\bfseries\normalsize\llap{\makebox[.75cm][l]{\etocnumber}}%
               \etocname\EndParWithPageNumberInMarginAndLeaders}
              {}
\etocsetstyle {subsection}
              {}
              {\leavevmode\leftskip 2.75cm\relax }
              {\mdseries\normalsize\llap{\makebox[1cm][l]{\etocnumber}}%
               \etocname\EndParWithPageNumberInMarginAndLeaders}
              {}
\etocsetstyle {subsubsection}
              {}
              {\leavevmode\leftskip 4cm\relax }
              {\mdseries\normalsize\llap{\makebox[1.25cm][l]{\etocnumber}}%
               \etocname\EndParWithPageNumberInMarginAndLeaders}
              {}
\etocsetstyle {paragraph}
              {}
              {\leavevmode\leftskip 5.5cm\relax }
              {\mdseries\normalsize\llap{\makebox[1.5cm][l]{\etocnumber}}%
               \etocname\EndParWithPageNumberInMarginAndLeaders}
              {}
\etocsettagdepth {preamble}    {part}
\etocsettagdepth {overview}    {part}
\etocsettagdepth {styling}     {part}
\etocsettagdepth {control}     {subsubsection}
\etocsettagdepth {examples}    {part}
\etocsettagdepth {advanced}    {part}
```

```
\etocsettagdepth {etocandworld}{part}
\etocsettagdepth {code}        {paragraph}
\renewcommand\etoctoprule {\hrule height 3pt\relax }
\renewcommand\etoctoprulecolorcmd {\color{blue}}
\renewcommand\etocaftercontentshook
   {\medskip\begingroup \color{blue}\hrule height 3pt \endgroup }
\etocruledstyle [1]{\Large\bfseries
                    \fbox{\makebox[8cm]{A TOC using depth tags}}}
\sloppy
\etocobeydepthtags % let's not forget to activate this (default anyhow)
\tableofcontents
\endgroup
```

Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an "attachments" or "comments" panel is available. etocsnippet-24.tex

## 44. Sections styling subsections

Here is a subtler example where one only marginally modifies the sections (adding color to the number, removing the hyperref link, changing the color for one specific section) but let them decide almost one by one of the style which will be followed by their subsections. Furthermore one configures some sections to decide they will not display subsections at all or only count how many there are!

```
\makeatletter
\newcommand*{\MyQuasiStandardTOC}[2][]{%
  % #1 is an optional "\ref{somelabeltoanothertoc}"
  % #2 is the number of some exceptional section
  \begingroup
  \etocsetstyle{section}
    {}
    {\etociffirst{% Suppress display of subsections for the first section!
                  \etocsetlevel{subsection}{\etocthemaxlevel}}
                 {\etocsetlevel{subsection}{2}}%
     \ifnum\etocthenumber=#2 % Handle especially section number #2 !
      \etocsetstyle{subsection}
        {\def\foo{}\par\nopagebreak\begingroup
         \leftskip2em \rightskip\@tocrmarg
         \parfillskip \@flushglue
         \parindent 0pt
         \normalfont\normalsize\rmfamily\itshape
         \etocskipfirstprefix}
        {\allowbreak\,-\,}
        {\edef\foo{\the\numexpr\foo+1}\etocname\ \textup{(\etocnumber)}}
        {.\par \upshape My AI counted circa \foo\space subsections,
                        was it right?\par\endgroup}%
     \else
      \ifnum\etocthenumber>#2 % Only count subsections in those sections !
        \etocsetstyle{subsection}
        {\def\foo{}}%
```

```
        {\edef\foo{\the\numexpr\foo+1}}%
        {}%
        {\leftskip2em \emph{There are \foo\space subsections here,
                      but I will need payment to display them.}\par}%
      \else
        \etocsetstyle{subsection}
        {}%
        {}%
        {\l@subsection{\numberline{\etocnumber}\etocname}{\etocpage}}%
        {}%
      \fi
    \fi
    }%
    {% Display in a special color the number of the special section!
     \l@section{\numberline{{\ifnum\etocthenumber=#2
                              \color{red}\else\color{cyan}\fi\etocthenumber}}%
                \etociffirst{\etocname\space (SUBSECTIONS SKIPPED)}{\etocname}}
               {\etocpage}}%
    {}%
  \etocclasstocstyle % will use the ambient document class
     % special KOMA-script customization as this document uses scrartcl
     % and we need to enlarge numwidth for some subsections
     \DeclareTOCStyleEntry[numwidth=2em,indent=0pt]{tocline}{section}
     \DeclareTOCStyleEntry[numwidth=2.5em,indent=2em]{tocline}{subsection}
  \etocsetnexttocdepth {subsection}%
  \tableofcontents #1
  \endgroup
}
\makeatother
```

etocsnippet-25.tex Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

The optional argument stands for a suitable `\ref`, see section 15. Here is what

```
\MyQuasiStandardTOC[\ref{toc:part:styling}]{11}% treat especially section 11
```

gives:

## Contents

My AI counted circa 9 subsections, was it right?



The page heading (on the page where this TOC appears) may have been modified as is expected from usage of `\etocclasstocstyle` in the code.

Sections are printed exactly as in the default, *except* that a special section number is displayed especially and that its subsections are then displayed inline! And other sections may only count subsections or ignore them altogether!

Note: for this code to work the macro `\thesection` *must* not have been modified from its default which produces simply arabic digits. In other terms it must be usable as TEX number denotation. Typically this would fail if we were handling subsections as their number in the `.toc` file will be typically something like `3.7` which does not work in a TEX integer-only context. But it is possible to use a counter and configure the line styles to increment it, and use its value.

## 45. The TOC as a (long) table (alternative)

Due to, among other things, the fact that alignment cells create and close groups, and that by default definitions of `\etocname`, `\etocnumber`, `\etocpage` made by **etoc** are local, it was not easy to typeset a TOC as table with **etoc**, prior to release to the addition of `\etocglobaldefs` at `1.08`.

Not only `\etocname` etc... caused a problem, but also the basic redefinition of `\contentsline` was made by **etoc** only after the first argument to `\etocsettocstyle` had been executed, hence if this argument were to open a tabular, the **etoc** redefinition of `\contentsline` would be done in the first cell of the first row and get lost thereafter.

Thus one had to resort to the technique explained in section 36 of using the execution of `\tableofcontents` as a way to store data which was then displayed later.

For the record, here is how the TOC from section 39 was coded in the old days.[50] We don't have here the problems with the positioning of `\hline`'s we face with the newer method; on the other hand we must manipulate token registers which are not familiar to most LATEX users (macros could be used, but would be more cumbersome, except perhaps if using the $\varepsilon$-TEX `\unexpanded`).

The method here is the most powerful because it filters out of the `.toc` file only the data we want (the other things are not ignored, they are executed but hopefully do not create havoc; typically they are language changing instructions, etc...), and we are less susceptible to fall potential victims of various external macros inserted in the `.toc` file by other packages.

Note: rather than `\toks` registers it would be easier here to use $\varepsilon$-TEX `\unexpanded` primitive. See for example section 38.

```
\newtoks\toctabletok
```

[50] At release `1.09f` the design anb contents of the TOC from section 39 were modified; the code here, if executed, which will not, reproduces the former looks.

```
\newcommand*\appendtotok[2]{% #1=toks variable, #2=macro, expands once #2
  #1\expandafter\expandafter\expandafter {\expandafter\the\expandafter #1#2}}

\newcommand*\PreparePart{%
  \toks0 \expandafter{\etocthelinkednumber}%
  \toks2 \expandafter{\etocthelinkedname}%
  \toks4 \expandafter{\etocthelinkedpage}%
  \edef\toctablepiece {\noexpand\hline
                       \noexpand\strut\the\toks0 &\noexpand\bfseries\the\toks2
                                 &\the\toks4 \noexpand\\\noexpand\hline}%
}
\newcommand*\PrepareSection{%
  \toks0 \expandafter{\etocthelinkednumber}%
  \toks2 \expandafter{\etocthelinkedname}%
  \toks4 \expandafter{\etocthelinkedpage}%
  \edef\toctablepiece {\the\toks0 &\the\toks2 &\the\toks4 \noexpand\\}%
}
%
\newcommand*{\PrepareSubsection}{%
  \toks0 \expandafter{\etocthelinkednumber}%
  \toks2 \expandafter{\expandafter\itshape\etocthelinkedname\strut}%
  \toks4 \expandafter{\expandafter\itshape\etocthelinkedpage}%
  \edef\toctablepiece{&\noexpand\makebox[1cm][c]{\the\toks0}%
                       \noexpand\parbox[t]{\dimexpr6cm-\tabcolsep\relax}
                                          {\noexpand\sloppy\the\toks2}%
                       &\the\toks4 \noexpand\\}%
}

\begingroup
\etocsetstyle{part}{}{}{\PreparePart \appendtotok\toctabletok\toctablepiece}{}

\etocsetstyle{section}{}{}{\PrepareSection \appendtotok\toctabletok\toctablepiece}{}

\etocsetstyle{subsection}{}{}{\PrepareSubsection\appendtotok\toctabletok\toctablepiece}{}

\etocsettocstyle
    {\toctabletok{\hypersetup{hidelinks}%
                  \begin{longtable}{|>{\bfseries}c|p{7cm}|r|}\hline
                  \multicolumn{3}{|c|}{\Large\bfseries\strut TABLE OF CONTENTS}%
                  \\\hline\hline}}
    {\global\toctabletok\expandafter{\the\toctabletok\hline\end{longtable}}}
\etocsettocdepth {subsection}
\tableofcontents
\the\toctabletok
\endgroup
```

etocsnippet-26.tex Depending on the PDF viewer, a click (or CTRL-click) on the filename in the margin may allow to extract it. Or check if an “attachments” or “comments” panel is available.

# Part VI.

# etoc and the world



## 46. Constraints on the .toc file constitution

The contents of the .toc file (if it already exists) are read into memory by etoc once, at the time of \begin{document}.[51]

The .toc file remains available to other packages for read operations until the location of the first table of contents at which time a write stream is opened by etoc and from that point the file is erased until its contents are again written to the disk by LaTeX at the end of the compilation.

Don't use \if<condition> stuff \tableofcontents\fi, but:

```
\if<condition> stuff \expandafter\tableofcontents\fi
```

Also a \else immediately following \tableofcontents or \localtableofcontents requires a previous \expandafter.

etoc can not really cohabit with packages modifying the \tableofcontents command: some sort of truce can be achieved if etoc is loaded last, hence is the winner.

> Do not modify the \tableofcontents command like this:
>
> ```
> \let\oldtableofcontents\tableofcontents
> ```

[51] Versions earlier than 1.07m read the .toc file at the time of \usepackage{etoc}. Thanks to Denis Bitouzé who signaled a Babel related problem, which turned out to be caused by this.

```
\renewcommand\tableofcontents{\oldtableofcontents foo}
```

as this will make the `\label`/`\ref` mechanism impossible.

Rather, redefine `\etocaftertochook`

```
\renewcommand\etocaftertochook{foo}
```

and there is also `\etocaftercontentshook` which is executed a bit earlier[52] just before the closing part of the toc display style (and thus still within a group.)

Prepending is less of a problem (and anyhow there is also `\etocbeforetitlehook` available to the user).

Under certain circumstances **etoc** imposes its views on `\tableofcontents` at the time of `\begin{document}`. You may thus have to use `\AtBeginDocument` to delay your (necessarily ugly and non-recommendable) patches. Patching after `\begin{document}` is naturally possible but I feel almost a rebel to mention this to LaTeX users!

**etoc requires** the `.toc` file to use the `\contentsline` macro. It **can not** work if there is no `.toc` file or if the `.toc` file does not use the `\contentsline` macro or if the `\contentsline` macro second argument mixes number and name in a manner unexpected by **etoc**.[53]

# 47. Compatibility with document classes

**etoc** has mainly been tested with the `article` and book standard classes. Some compatibility layer with the KOMA-script and memoir classes was added at `1.05` of `2012/12/01`.[54] A compatibility layer is required for what this document designates as the "global display style": **etoc** tries to emulate the default behaviour of the document class when `\etocsettocstyle` has not been used, so that it is still in "compatibility mode". There does not seem to be an easy way to extract this in an automated manner dynamically, so it is basically some manual work which the author initiated in 2012 and which got sporadically updated since.

## 47.1. Compatibility with the KOMA-script classes

Not really tested... well, tested by this document with its dozens of **etoc** TOCs and which uses scrartcl! The package code contains

```
\@ifclassloaded{scrartcl}
    {\renewcommand*\etocclasstocstyle{\etocscrartclstyle}}{}
```

with `\etocscrartclstyle` trying to emulate the global display style of the `\tableofcontents` within the class scrartcl. Thus **etoc** is ready for basic usage in compatibility mode. See further subsection 48.6.

[52] Contrarily to `\etocaftertochook`, `\etocaftercontentshook` is not executed if the `tocdepth` did not allow the printing of the TOC.

[53] Prior to `1.1a`, **etoc** required the `\contentsline` macro expansion to invoke `\l@section` et al. It now does not expand `\contentsline` hence is immune to whatever would happen during this expansion, and does not use nor modify the `\l@section` et al. macros.

[54] That this is still in the doc on `2023/02/22` shows the frequency of **etoc** syncing with upstream changes... but let me reassure the reader there has been indeed some occasional, although admittedly rare, internal updates in the more than ten years elapsed since.

### 47.2. Compatibility with the memoir class

Release 1.07l has also improved the compatibility with the memoir class: its appendix level has been made known to etoc. It is at the same level as chapter, thus the chapter line style should possibly do a test for some user defined boolean whose activation may be added to the .toc file at the suitable location via \addtocontents{toc}{..}, if one needs to distinguish the two kinds of divisions.

The memoir mechanism relative to \tableofcontents versus \tableofcontents* is obeyed automatically, and applies with \localtableofcontents too.

With release 1.2 some issues which were previously described here when the "to toc" feature of memoir was left acting on local TOCs have been resolved, as an after-effect of the support for localtoctotoc etc..., see \etocsetup.

### 47.3. Compatibility with beamer

For the reasons mentioned already regarding the constraints on the .toc file constitution, etoc is incompatible with the beamer class. However, if beamer is used in an article mode, i.e., with the article class in conjunction with the beamerarticle package, then etoc should work.

## 48. Compatibility with other packages

### 48.1. Compatibility with babel

One must load etoc *after* babel. This is in order for babel's shorthands to be active at the time when etoc loads the .toc file.

### 48.2. Compatibility with hyperref

Please inform the author in case of issues: etoc was from the start designed to be 100% compatible with package hyperref.

The macros \etocname, \etocnumber, and \etocpage contain the hyperref links, if present (note that the *linktoc=all* option of hyperref tells it to put a link also in the page number corresponding to a given toc entry). For example, the tables of contents of the present document are all fully linked. It doesn't matter whether etoc or hyperref is loaded first.

### 48.3. Compatibility with microtype

(this doc added at 1.2b)

microtype patches the LaTeX \numberline command (which is used in .toc file) to execute \leftprotrusion. The reason microtype does this patch to \numberline is that the default LaTeX contents line styles do some \hskip which inhibit left protrusion of the (explicit digits or letters of the) argument of \numberline, which is the first typeset material on the contents line.

Some other contexts than a previous \hskip inhibit left protrusion, in particular a non-zero paragraph indentation.

One could imagine that etoc user only has to add \leftprotrusion if so desired at start of the defined line styles. But usage of the \leftprotrusion command is delicate, it must be followed with an explicit character or with a very limited set of LaTeX commands. In particular, and imagining here that the defined line style starts a paragraph with \etocnumber, and that this paragraph is indented so that automatic left protrusion does not apply, an attempt to force reactivation of left protrusion via:

```
\leftprotrusion\etocnumber
```

fails. One has to resort to use:

```
\expandafter\leftprotrusion\etocthenumber
```

This would not work with \etocthelinkednumber, so to get the protruding number to also be hyperlinked one has to use

```
\etoclink{\expandafter\leftprotrusion\etocthenumber}
```

The package does not provide yet (nor is it planned in near future in absence of user requests) extra support such as an option to insert internally \leftprotrusion *inside* \etocnumber which would avoid such gymnastics. But the user can easily add a definition such as

```
\newcommand\leftprotrudingetocnumber{\etoclink{\expandafter\leftprotrusion\etocthenumber}}
```

to the preamble, and use the defined wrapper or a variant built upon \etocthename in custom \etocsetstyle line style definitions.

If on the contrary you want to avoid left protrusion, it is coherent to load microtype this way:

```
\usepackage[nopatch=toc]{microtype}
```

This avoids the \leftprotrusion addition done by microtype to \numberline; but this is of relevance only if you use etoc in compatibility mode, because user-defined etoc line styles will not execute \numberline except if explicitly added to the line style, and even if they do, as explained above something such as

```
\numberline{\etocnumber}
```

will inject a \leftprotrusion which will remain without effect.

etoc provides since 1.2b a macro \etocnoprotrusion which is a copy of the LaTeX \noprotrusion command available since its 2018/12/01 release, and is used since by LaTeX default TOC code with page numbers. You can use this if a style definition, perhaps modeled on the standard LaTeX layout from \@dottedtocline, also locates \etocpage at right margin preceded on the line by some dot leaders. Indeed protrusion may cause an extra dot, compared to other lines, which is really ugly visually. Use \noprotrusion in the style after \etocpage (or whatever is at end of the line style contents) if you can assume that LaTeX is recent enough, or use \etocnoprotrusion which will work with all LaTeX versions, even old ones.

You may also consider turning protrusion off completely for tables of contents. For this, see subsection 20.2.

Notice that some problem with page number alignment look like protrusion-related ones but may simply be caused by the fact that the font used does not assign the same width to all digits, and the line style pushes these digits to the right end of the line (to confirm this in case of suspicion, temporarily make package microtype inactive or remove it from your document). If

for example 20 is, in the used font, wider than 27, the latter will look shifted a bit to the right compared to the former. Centering the page number in a fixed width box (itself flushed-right to end of line) is one solution, but it may not be to everyone's taste due to single-digit page number; in that case the line style code could make a test on the page number (or its width, if it is not assumed it is purely numerical) and make appropriate decisions. I will leave the matter to your LaTeX coding skills. Using a font with tabular lining figures fixes such problem.

## 48.4. Compatibility with multicol

etoc loads the package multicol.

## 48.5. Compatibility with tableof

It is possible to use simultaneously etoc and tableof. Release 1.08 of etoc requires at least version 1.4a of tableof. If \etocglobaldefs is put in the preamble, this must be after the loading of package tableof. tableof command \nexttocwithtags should work as expected.

tableof commands \tableof, \tablenotof, ... will typeset the (a priori global) table of contents according to the document class defaults, obeying the etoc depth tags; as explained in the tableof documentation they do not typeset a TOC title. They should *not* be used in case \etocglobaldefs was issued before, except if its scope has been terminated since then, or \etoclocaldefs has cancelled its influence.

## 48.6. Compatibility with tocbasic

(doc last updated for 1.2)

This has only been tested to the extent of production of this PDF file (which does have dozens of TOCs, among them a few in compatibility mode). I noticed that some things required the global display style to also be set to compatiblity mode for some KOMA-script user interface to produce the desired effect. As an example, the local table of contents of section 27 uses \KOMAoptions {toc=left} and \etocstandardlines but it had to use also \etocclasstocstyle:

```
\begingroup % to keep in particular toc=left with local effect
\KOMAoptions{toc=left}
\etocclasstocstyle % necessary for the display to obey toc=left
\etocstandardlines
\tableofcontents \ref{toc:tocstyle}
\endgroup
```

On the other hand I could use some other interface without having to activate the compatibility for the global display style. See the example from section 15 or the one from the Part III which uses

```
\begingroup
\etocsettocstyle{}{}
\DeclareTOCStyleEntry[numwidth=2em,indent=0pt]{tocline}{section}
\DeclareTOCStyleEntry[numwidth=2.5em,indent=2em]{tocline}{subsection}
\DeclareTOCStyleEntry[numwidth=3em,indent=4.5em]{tocline}{subsubsection}
```

```
\etocstandardlines
\localtableofcontents
\endgroup
```

and demonstrates that this interface works also when not using the compatibility mode from \etocclasstocstyle. Of course, here the display style is a bit plain... but the example of section 15 less so; it uses about the same TOCStyleEntry lines as above with some more width for the numbers of subsections.

## 48.7. Compatibility with tocloft

Release 1.07k added compatibility with package tocloft: steps are taken to prevent the redefinition of \tableofcontents done by tocloft at \begin{document}. As long as etoc is left in compatibility mode the customization done by tocloft will be obeyed, for both the line styles and the TOC title. One may still benefit from the *depth tags* management by etoc, from its \localtableofcontents, from its \label+\ref mechanism. One may use \etocsetstyle to define via etoc the layout for one TOC and then use rather tocloft for another one, if \tableofcontents follows \etocstandardlines and \etocclasstocstyle. In this compatibility mode \etocsetlevel{⟨*division unit*⟩}{\etocthemaxlevel} will render invisible the chosen division level, but exchanging levels is otherwise not possible.

> One should load etoc *after* tocloft. A warning is issued if otherwise, because if etoc is loaded before it will realize that at the time of \begin{document} and trick tocloft into believing having been loaded with the titles option.
>
> Sadly, 1.2 and 1.2a had a bug and loading tocloft before etoc caused \tableofcontents to not be the etoc one. One had to use explicitly \etoctableofcontents (there was no issue with \localtableofcontents). The regression was fixed at 1.2b.

It is possible to modify midway in the document the macros \l@section, \l@subsection ... but the effect will be seen only in table of contents typeset by etoc in compatibility mode (and of course after those customizations). It will have no effect on true etoc TOCs.

## 48.8. Compatibility with tocbibind

Added at 1.2. See also \etocsetup for a discussion of the package options which are all related to this (they will achieve the tocbibind "to toc" features without requiring the package).

Thanks to Denis Bitouzé for feature request.

## 48.9. Compatibility with tocvsec2

etoc used to be incompatible with package tocvsec2; it now cohabits, sort of, as it deactivates tocvsec2's modification of \tableofcontents and also cancels its other toc-related macros, but reimplements partially their functionality with \etocsettocdepth.toc. By the way, at least two latex runs are necessary for new uses of this command in a document to have an effect in tables of contents.

# 49. TEXnical matters

The \etocname, \etocnumber, \etocpage commands are protected against premature expansion. They are hyperlinks if package hyperref is loaded and depending on its option linktoc value; under the default linktoc=section, only name and number are hyperlinked, not the page number.

On the other hand \etocthename, \etocthenumber, \etocthepage are *not* protected against expansion. And neither are \etocthelinkedname, \etocthelinkednumber, \etocthelinkedpage. They were modified at 1.1a and now are always hyperlinks (except for the latter if the page number is empty), if hyperref is present, independently of linktoc status.

The commands \etoclink and \etocifnumbered are also protected against premature expansion. Also \etociffirst and \etoctoccontentsline.

Commands such as \etocsetstyle, \etocsetlevel, \etocsettocstyle, \etocmulticolstyle, \etocruledstyle, \etocframedstyle obey LaTeX's groups. All TOCs are typeset inside groups.

When a \localtableofcontents is inserted by the user in the document, a line containing an **etoc** inner command and an identification number is added to the .toc file on first subsequent compilation. The correct local table of contents will be displayed only on the second compilation.

After using \etocsetstyle for one level, the remaining uncustomized levels use the **etoc** default styles (those which are activated by \etocdefaultlines). One has to make sure that **all levels** needed for the next table of contents are mutually compatible: in particular the **etoc** default line styles expect each to be started in "vertical mode".

When using multiple \tableofcontents commands in a document, one should beware from adding typesetting instructions directly in the .toc file, as they will be executed by **etoc** for **all TOCs**: even for a \localtableofcontents it doesn't matter if that instruction seems to concern material outside of its scope, it will get executed nevertheless. If absolutely necessary to add extra commands to the .toc file, make it in such a way that they can be activated or deactivated easily from the document source, e.g. via some booleans.

As is usual with toc and labels, after each change, one has to run latex a certain number of times to let the produced document get its final appearance (at least twice).

# Part VII.

# The code



## 50. Timestamp

This is the documentation as of 2023/10/28, printed from the source file with the time stamp Time-stamp: <04-07-2023 at 22:42:41 CEST>. The package version is 1.2c, of 2023/10/28.

## 51. Change history

1.2c [2023/10/28]

Compatibility hotfix with hyperref v7.01c; etoc had copied over a now unneeded internal hyperref test, and since v7.01c the result was that etoc produced empty tables of contents. Thanks to Denis Bitouzé for report.

1.2b [2023/07/01]

bugfix: a refactoring at 1.2 accidentally removed a needed \AtBeginDocument and as a result the tocloft redefinition of \tableofcontents was not undone, if etoc got loaded after tocloft, as is advised by the documentation. Thanks to user691586 for report.

Add section documenting compatibility aspects with usage of package microtype. Add \etocnoprotrusion an alias to LaTeX 2018/12/01 \noprotrusion and use it after \etocpage inside the package built-in fallback section and subsection toc line styles.

Fix some of the documentation (in particular the sections "A first example" and "A second example") as some of its highly pedagogical comments had lost sync with other changes. Don't hesitate getting in touch with the author to obtain further improvements.

1.2a [2023/05/01]

The requirements (added respectively at 1.1a and 1.2) of a LaTeX kernel at least as recent as 2020-10-01 and of availability of \expanded engine primitive have been lifted. Due to lack of time and ressources, etoc is not retro-tested on old LaTeX installations, though.

Boolean option 'deeplevels' extends the maximal allowed numerical level from 6 to 12. Adapted from a patch courtesy of Matthew Trescott. See

https://github.com/doxygen/doxygen/pull/9936

for context.

Using this option does not change anything to tocdepth and secnumdepth LaTeX counters, nor does it do any of the needed extras to let a sectioning level be known to LaTeX (if etoc is used in "compatibility mode", the suitable \l@<levelname> macros must have been declared to LaTeX for TOC rendering). Whether or not using etoc in compatibility mode (i.e. whether or not using \etocsetstyle{<levelname>}{...}{...}{...}{...}), recall that all new levels must be declared to etoc at least once via \etocsetlevel{<levelname>}{<number>}.

The macro \etocthemaxlevel expands accordingly to either 6 (default) or 12 (if option 'deeplevels'). So \etocsetlevel{foo}{\etocthemaxlevel} should now be used in place of \etocsetlevel{foo}{6} when one wants to make foo entries invisible in a document context which may have used, or not, the 'deeplevels' option.

The auto-selection-of-heading-by-local-tocs from 1.2 uses a macro \etocdivisionnameatlevel which will need most probably user adaptation with 'deeplevels', as etoc can not know what are the standard names chosen by user, and how many deep levels were actually added. So it opted to the choices done by the Doxygen project. Check the etoc source code for this macro and redefine it appropriately. Note though that this only matters if your documents has local TOCs located under division units at least as deep as subsubsection.

Fix a 1.2 bug: using 'six' as first argument of '\etocsetlevel' caused an internal control sequence to be redefined, with various strange potential aftereffects.

1.2 [2023/03/01]

\locallistoffigures and \locallistoftables.

MAKE SURE TO READ THE FRAMED TEXT IN THE DOCUMENTATION AT START OF SECTION 2 WHICH DESCRIBES THESE NEW COMMANDS.

It is for time being required to pass options 'lof', resp. 'lot' at package loading time to activate these features. Whether or not these features are enabled, etoc interferes in absolutely no way with \listoffigures and \listoftables and their respective auxiliary files.

Boolean options (and utility \etocsetup) 'maintoctotoc', 'localtoctotoc', 'localloftotoc', 'locallottotoc', and 'ouroboros'. For above reasons, no 'mainloftotoc' or 'mainlottotoc'. With 'ouroboros=false' local TOCs whose heading has been added to the '.toc' file due to 'localtoctotoc' will show deeper ones and the local 'lists of' from their own level, but not themselves.

Compatibility layer with 'tocbibind'. Thanks to Denis Bitouzé for feature request many years ago. Sorry for the delay!

New feature (and breaking change): local tables of contents in the default package configuration auto-adapt their titles to use the appropriate unnumbered division heading (for example a \subsection* if inside a \section).

Formerly, it was intended one would use \etocsettocstyle (or higher level commands), else local TOCs were rendered using the same emulation of the document class as for the main TOC. One can still require this document class emulation but then etoc tries at least to adapt to the division unit one level below \part (i.e. \chapter or \section). E.g., for KOMA-script the option 'leveldown' is used. But this is not appropriate for deeper levels in general.

Ultimate control of toc titling can still be achieved via \etocsettocstyle. Many commands have been added to ease its usage. It is also possible to customize the default behavior without \etocsettocstyle.

Details of the user interface and produced output from these novel commands and options are susceptible to change. Use at own risk.

Bugfix: the emptiness check could fail for \localtableofcontentswithrelativedepth as it ignored tocdepth changes originating from inside the .toc file, contrarily to the actual typesetting (and behavior with other local tocs)

Bugfix: the \tableofcontents\ref{foo} syntax caused a "Missing number" after \vpageref{foo} from package varioref in certain circumstances due to \r@foo having then acquired a value with an unexpected non-numerical first chunk.

Breaking change: the \etocsavedsectiontocline et al. macros which raised a deprecation warning since 1.1a now use an error message.

\expanded primitive is required. [reverted at 1.2a]

1.1d (not released, part of 1.2)

Complete internal refactoring of the legacy core code behind the concept of "start" and "finish" parts for line styles.

\etocifunknownlevelTF and \etoclevel abstract away some internals and may be useful for class or package authors.

\etocsetlevel filters out forbidden keywords 'all', 'none' and others. Its second argument can be an expression.

\etocsetstyle checks and warns appropriately if the level is unknown to etoc or has a non-accepted numerical value.

Extensive refactoring of the documentation.

1.1c [2023/01/20]

Fix a brace removal bug in the construction of \etocname. It remained without visible effects in documents using hyperref and default settings, thanks to the hyperlink wrapper, but e.g. \section{{\color{blue}Blue}} in a document not using hyperref, and not using etoc only in "compatibility mode", could cause a color leak in the table of contents.

With the KOMA-script numberline toc feature, unnumbered entries in TOCs typeset via etoc user-defined or package provided line styles but using compatibility mode for the global display style were (knowingly) considered to be numbered with an empty number. They are now considered by \etocifnumbered to be not numbered and the empty \etocnumber will carry no hyperlink.

Fix a 1.1a regression in the context of KOMA-script unnumbered TOC entries: \etocthelinkedname could lose its hyperlink.

Continue internal trimming of old code branches which became un-needed after the 1.1a refactoring. Add relatively decent code comments to accompany the 1.1a-c refactoring. Update warning messages to use more consistently LaTeX's templates.

1.1b [2023/01/15]

Documentation fix, 1.1a forgot to mention the following change: \etocthelinkedname, \etocthelinkednumber, \etocthelinkedpage are now always hyperlinks independently of linktoc status.

1.1a [2023/01/14]

This version brings no new functionality, despite the number bump. It implements a complete rewrite of old legacy core internals. Formerly, etoc waited for hyperref (if present) to have added hyperlinks via its patch to LaTeX's \contentsline. etoc examined the arguments of \l@section and other commands to extract hyperlinking information, if any. With this release etoc decides earlier according to hyperref linktoc status whether section names and page numbers should be hyperlinked, and adds links itself via \hyperlink. etoc is thus now immune to the details of how hyperref patches the \contentsline command, which is not executed anymore. Overall, the code is greatly simplified.

\etoclink now wraps its argument in an hyperlink even if hyperref is configured via linktoc=none. Formerly no hyperlink was added then.

Deprecation of \etocsavedsectiontocline and similarly named commands. They are not needed as \l@section et al. are with this release left unmodified during the table of contents typesetting.

LaTeX kernel from 2020-10-01 or later is required (to allow assuming the \contentsline entries in the TOC file always have four arguments). [reverted at 1.2a]

1.09i [2022/11/21]

Fix bug showing when a document uses both \etocchecksemptiness and \etocsetlocaltop.toc: the start and finish parts of some levels were executed possibly causing extra printed output.

More hyperlinking in the implementation part of the documentation.

1.09h [2022/11/20]

Documentation improvements. In particular, attached code snippets are now visible via their filenames in the page margins. Also, command names are doubly hyperlinked: first half links to the devoted part of the user manual, second half links to the implementation part.

1.09g [2022/11/17]

Compatibility hotfix with recent hyperref 7.00u of 2022-11-13. Thanks to Denis Bitouzé for signaling the breakage to the author.

1.09f [2022/08/30]

No more shipping of a German translation of the documentation, as it was last updated in April 2015.

(etoc.pdf) User level commands hyperlink from their code source definitions to their descriptions in the documentation part. Macros used in the code source hyperlink to where they first got defined there.

Wrap the \etocpartname (from etoc's package provided toc line style) together with the part number in a potential common hyperlink.

Try to sync the emulation of the global display style with KOMA-script v3.37 (in particular regarding the noparskipfake KOMA toc feature).

Improve documentation of some aspects under memoir class.

Remove the \nonumberline token, even though empty, from the meaning of \etocthename (KOMA-script classes).

Add \etocimmediatedepthtag.toc to work around problems related to \include (see user doc). Thanks to Norman Ramsey who reported the problem and proposed a workaround in July 2016. Apologies for the somewhat longish delay in incorporating it...

Also add \etocimmediatesettocdepth.toc.

Also add \etocimmediatetoccontentsline and its starred variant.

Also add \etocimmediatesetlocaltop.toc.

Fix an obscure bug (see source code comments) in the \etocsetlocaltop.toc mechanism.

1.09e [2021/09/23]

Needed (if etoc is used without hyperref) updates to internal macros to prepare for the upcoming LaTeX November 2021 change to \contentsline.

Related updates to the user macro \etoctoccontentsline.

1.09d [2021/07/13]

Some minor synching with tableof 1.4c.

Add \etockeeporiginaltableofcontents to provide a workaround to a compatibility issue with listings's \lstlistoflistings, which abuses \tableofcontents for doing something unrelated to the actual contents. Thanks to Denis Bitouzé for report.

Usage: \usepackage{etoc}\etockeeporiginaltableofcontents, then however you must employ \etoctableofcontents, not \tableofcontents.

1.09c [2020/05/15]

Syncs with KOMA-script deprecation of \iftocfeature.

1.09b [2019/11/17]

1.09a

Sync with memoir v3.7i which has a better location of the TOC hyperref anchor. The \etocaftertitlehook can now freely be used also with memoir class (formerly its usage in case of memoir class was preempted by etoc itself). For more details refer to the section "Compatibility with the memoir class".

1.09 [2019/03/09]

New features: \etoclocaltop, \localtableofcontentswithrelativedepth. Thanks to Tony Roberts for feature request.

Note to hackers: internal control sequence \Etoc@localtop is gone.

etoc now requires e-TeX (\numexpr, \unless).

1.08p [2018/07/04]

Fixed bug surfacing in case of linktoc=page option of hyperref. Thanks to Denis Bitouzé for report. Cf:
https://github.com/ho-tex/hyperref/issues/65
https://github.com/dbitouze/yathesis/issues/61

1.08o [2018/06/15]

Fixed bug showing up if an unnumbered TOC entry starts with a brace, and document uses hyperref. Caused by a typo in a macro name at previous release.

1.08n [2018/02/23]

Refactoring of core macros detecting \numberline and its variants.

1.08m [2018/02/07]

Fix to 1.08k's introduced incompatibility with KOMA-script and tocbasic's \nonumberline.

1.08l [2017/10/23]

Workaround an issue with Emacs/AUCTeX wrongly reporting about actually non-existent LaTeX errors, which was triggered by some strings written (indirectly) to log file by etoc under some circumstances.

1.08k [2017/09/28]

Adds \etocsetlocaltop.toc. See corresponding manual section for details.

Adds \etocsavedparttocline, \etocsavedchaptertocline, \etocsavedsectiontocline, ... They can be used in the context of the technique explained in section "Another compatibility mode".

Formerly, etoc redefined for the duration of the TOC the memoir macro \chapternumberline and its likes to have

same meaning as \numberline (of course, not when executed in compatibility mode), for the sake of extraction of \etocnumber.

New method detects presence of any \<foo>numberline macro without any change to originals; they can thus be used as is when applying the approach of "Another compatibility mode" section from manual.

1.08j [2017/09/21]

Since 1.08a-2015/03/13 \etocname, \etocnumber, \etocpage contain, if hyperref is present and configured for using hyperlinks in the TOC, the link destination in already expanded form. This means one can use them even if the style closes a group (for example from a & in a tabular), if \etocglobaldefs was issued; also one can save their meaning for delayed usage (with for example \LetLtxMacro as they are robust).

But for some legacy reason \etoclink, contrarily to \etocthelink, was handled differently. Now, \etoclink also contains the link destination in already expanded form, and can thus be used even if the line style issues a &, as long as \etocglobaldefs is issued.

Also, bugs dating back to the early days of the package, but surfacing only under relatively rare conditions such as usage of hyperref with its option "linktoc=page" got fixed.

1.08i [2016/09/29]

This fixes an issue dating back to 1.08e-2015/04/17: under \etoccheckseemptiness regime, some circumstances (such as adding to an already compiled document a \localtableofcontents before the main \tableofcontents) created an "Undefined control sequence \Etoc@localtop" error. Thanks to Denis Bitouzé for reporting the problem.

On this occasion, \etocdoesnotcheckemptiness has been added to unset the flag.

A rather more exotic issue was fixed: the emptiness check for local tocs could get confused if the tocdepth counter was varying in some specific ways from inside the toc file.

After adding to a document a \localtableofcontents, two LaTeX passes are needed for etoc to get a chance to print the correct local contents. Formerly, etoc issued a Warning on the first pass; it now also induces LaTeX into announcing "There were undefined references", as this is nearer to the end of the log file and console output.

1.08h [2016/09/25]

New functioning of \etocsetnexttocdepth: the tocdepth counter is modified only at the time of the table of contents, not before. This fixes an issue which arose when \etocsetnexttocdepth was used multiple times with no intervening table of contents. Thanks to Denis Bitouzé for reporting the problem.

The PDF documentation includes about 25 LaTeX code snippets also as file attachment annotations, additionally to their verbatim typesetting. The ordering of the documentation contents has been slightly re-organized.

A previous documentation-only update on 2016/09/09 added a new section with the (approximate) translation into etoc lingua of the book class toc style, for easy customizability.

1.08g [2015/08/29]

Downgraded to a mere info message the etoc-issued warning (relative to \settocdepth/\maxtocdepth) under class memoir.

1.08f [2015/04/28]

Minor changes to the documentation. \etocsetlevel more economical.

1.08e [2015/04/17]

The command \etoccheckseemptiness tells etoc to not print, from that point on, the headings of the local tables of contents if they have empty contents. This is mainly for class authors who might want to have their \section or \chapter automatically do a \localtableofcontents. Could prove also useful for batch conversions of documents. Thanks to Paul Gaborit who asked for such a feature.

The command \etocnotocifnotoc extends this behaviour to global TOCs: indeed why should documents with no sectioning units take this as an excuse not to use package etoc ?

The command \etocifwasempty{yes}{no} can be used for suitable extra action.

A \tableofcontents\ref{foo} now expects foo to be a label to a _local_ TOC. The use with foo a label to a _global_ TOC is not supported anymore as it had no utility and made the code more complex.

The syntax \localtableofcontents\ref{foo} is now accepted as a synonym to the earlier syntax \tableofcontents\ref{foo}.

1.08d [2015/04/09]

Translation into German of the additions made to the documentation for the 1.08x series of releases.

Thanks to Christine Römer!

1.08c [2015/03/30]

- removed a few unneeded \long from the code.

- removed use of \arabic at one location of the code, as it may get redefined by some language modules for babel or polyglossia.

1.08b [2015/03/18]

Bug fixes:

- extra space token removed from ‘\localtableofcontents‘ (showed only for inline TOCs.)

- \etocpartname (a macro used by the package own default line styles) was defined to be \partname, but this is not compatible at least with babel+french context. Now simply expands to Part.

- some problems fixed in the German documentation.

- [2015/03/28] some more problems fixed in the documentation. Added mention of \etocarticlestyle and \etocbookstyle.

1.08a [2015/03/13]

\etocname, \etocnumber and \etocpage are now the robust variants of \etocthelinkedname, \etocthelinkednumber

and \etocthelinkedpage. This should arguably have been done since the addition of the latter to etoc with 1.07f [2013/03/07]. The earlier robust commands \etocname etc... contained the hyperlink destination only in an unexpanded form.

The documentation has a brand new title page and a new section The TOC as a TikZ mind map both illustrating further uses of etoc to display tables of contents as trees in an automatic manner.

1.08 [2015/03/10]

\etocskipfirstprefix may now appear anywhere in the <start> part of a level style.

New commands \etociffirst, \etocxiffirst, \etocxifnumbered, \etocglobaldefs and \etoclocaldefs.

It is now possible to issue line style specifications directly with & and \\ tokens, in order to typeset a TOC as a tabular or longtable with the opening for example in the first argument of \etocsettocstyle and the closing in its second argument.

It is mandatory for such uses to issue \etocglobaldefs which tells etoc to proceed globally for certain definitions. This is also useful in the context of the inline environments of package enumitem.

On this occasion, various old parts of the code have been improved.

1.07n [2015/03/05]

No more use of \toks@ when etoc constructs \etocthelinkedname etc... Thus \toks@ can be put in the line styles in order to accumulate information. Only useful if it is certain nothing else will change \toks@ either.

In the documentation: list of main commands now in alphabetic order.

1.07m [2015/01/23]

Reading of .toc file is delayed to \begin{document} to account for possible Babel active characters used therein. Thanks to Denis Bitouzé who reported a Babel related problem.

Improved global toc display emulation under KOMA-script classes.

New command \etocbeforetitlehook. New command \etocdisplay.

1.07l [doc of 2014/04/29]

Added to the documentation an example of use of \etocthelinkedname together with an enumitem inline itemize* environment; moved main TOC to immediately after the title, and license to the first pages.

Incorporation of the translation into German done on the initiative of Christine Römer by Felix Baral-Weber, Jenny Rothkrämer-Vogt, Daniel Büttner, Claudia Dahl, Christian Otto and Christine Römer (FSU Jena). My grateful thanks to all!

1.07l [2014/04/22]

Fixes a bug with the 1.07k compatibility layer with tocloft which had broken the 1.07k (sic) compatibility with memoir (yes, memoir class 1.07k testing had been done before adding the tocloft thing to the source code . . . ). Also, etoc when detecting tocvsec2 now checks if this is under the memoir class, as then nothing special needs to be done to rescue \tableofcontents, contrarily to the situation with the native tocvsec2.

1.07k [2014/03/06]

Compatibility with package tocloft; and improved compatibility with class memoir. Novel TOC example in Overview.

1.07j [2013/12/03]

Some issues with the documentation formatting (now two-sided) have been addressed, and a novel documentation section “Typesetting the TOC as a table” has been added. Very minor code change (\Etoc@readtoc).

1.07i [2013/10/21]

Changes to the \etocmulticolstyle and \etocruledstyle codes to lessen the risk of a page break after the title (in the one-column case).

1.07h [2013/10/16]

New commands \etocdepthtag.toc, \etocsettagdepth, \etocobeydepthtags, \etocignoredepthtags.

1.07g [2013/10/13]

New commands \etocsettocdepth, \etocsettocdepth.toc, \etocobeytoctocdepth, \etocignoretoctocdepth which emulate part of tocvsec2 functionality ; measures to make tocvsec2 partially compatible with etoc.

New commands \etocsetnexttocdepth, \invisibletableofcontents, \invisiblelocaltableofcontents.

Switched from tikz-qtree to forest for the first ‘toc as tree’ example.

Command names are linked to their descriptions, and many other changes in the documentation.

Removed printing of temporary message when the local toc id is not yet stabilized; indeed \localtableofcontents can have many uses, such as filling up some token list register and one may wish to not have anything typeset, even in an intermediate run.

All of tex etoc.dtx, etex etoc.dtx, xetex etoc.dtx, latex etoc.dtx, pdflatex etoc.dtx are now possible, and the extracted file etoc.tex allows easy customization of compilation options for the documentation (default is via dvipdfmx which produces the smallest file).

1.07f [2013/03/07]

New macros \etocthelinkedname, \etocthelinkednumber, \etocthelinkedpage, and \etocthelink.

1.07e [2013/03/01]

Improvements in the package own line styles with regards to penalties and vertical spaces.

Addition to the documentation of an example of a tree-like table of contents (uses tikz).

More such examples added 2013/03/03.

1.07d [2013/02/24]

Minor code improvements and new documentation section "Another compatibility mode".

1.07b [2013/02/02]

Removal of the \xspace from the macros \etocname, \etocnumber, \etocpage.

Additional examples in the documentation.

1.07 [2013/01/29]

New commands:

\etocthename, \etocthenumber, \etocthepage, \etoclink,

\etoctoccontentsline, \etoctoccontentsline*

\etocnopar, \etocaftercontentshook

Modified command: \etocmulticolstyle

New documentation section "Surprising uses of etoc" which explains how to do "Lists of arbitrary things", in addition to the tables of contents.

1.06 [2012/12/07]

The standard macros \l@section etc... are modified only during the calls to \tableofcontents; they can thus be customized as will by the user (with the help of a package like tocloft) and this will be taken into account by etoc for the TOCs typeset in compatibility mode.

1.05 [2012/12/01]

\localtableofcontents replaces \tableofcontents* (for compatibility with the memoir class).

Compatibility with KOMA-script and memoir document classes.

1.04 [2012/11/24]

A (possibly local) table of contents can be labeled:

\tableofcontents \label{toc:1}

and reproduced elsewhere in the document (with a possibly completely different layout):

\tableofcontents \ref{toc:1}

1.02 [2012/11/18]

Initial version.

# 52. Implementation

**About the syntax highlighting of the code:** Control sequences are mostly hyperlinks. When a user level command gets defined it hyperlinks to the user documentation with bold face and **using this colour**. Further instances if they occur will use this colour to link to their place of first definition inside the code lines. Package macros with no user level documentation hyperlink to their first place of definition in the code. And such non-user level macros, at the location of their first definitions will have their names displayed using bold face and **this colour** (it is not an hyperlink then). Comments located *inside* the code source (very little are left at 1.1a) have been configured to be rendered in their own colour, and non commented-out and non-control sequences tokens use this colour. Other tokens use the fall-back normal colour. A package macro mentioned in code comments also hyperlinks to the location of its first definition using this colour.

*The "syntax highlighting" was added at release 1.09f of 2022/08/30.*

Known limitations:

1. The location of first definition may be disappointing as it may be a provisory definition.
2. Macros such as `\Etoc@next` are there for matters of code branching only, and the first encountered definition has no relevant significance.
3. Macros defined using `\csname...\endcsname` or alike constructs are not detected by the syntax highlighting automatization. Apart from manual intervention, this appears complex to solve as moreover the macro name may be there indirectly as an argument such as #1.

**An apology:** the code comments served mainly as a record for the author's benefit of the historical evolution of the package and rarely as a description of what the macros do. At 1.1a I have removed almost all code comments which had accumulated as in a palimpsest. As a result, very few comments actually remain. 1.1c re-added comments to those parts of the code which got refactored at and since 1.1a, revigorating the palimpsest stratification.

**About etoc and the processing of `.toc` file data:** etoc, when not left in compatibility mode, hijacks the `\contentsline` expansion so as to not execute `\l@chapter`, `\l@section` etc..., but rather to parse the data and extract from it the *name*, *number*, and *page number*. The LaTeX `.toc` data is *not structured*, but contains already typesetting mark-up. The etoc maneuvers to disentangle *name* and *number* are somewhat fragile as they expect the `.toc` file to contain the `\contentsline` arguments to be arranged in a certain manner. Of course etoc can be easily broken if changes happen to how data is stored there. Things would have been much easier for etoc in 2012 if the `\contentsline` arguments had considered the section titles

*This paragraph and the next were added very late in the history of the package, at 1.09h and later.*

(aka *name* for etoc) and their numbers (which are not numbers in the sense of things with which TEX can compute, in general) separately, each providing an argument to \contentsline. But some mix is prepared, which may depend on the document class also, and besides usually handles \part levels very differently. Fortunately upstream changes happen rarely.

*Prior to 1.1a, etoc aliased \l@chapter, \l@section, etc... (at the time of TOC typesetting only) to its own \Etoc@lxyz in order to leave time to hyperref to add its mark-up. Thus the disentangling of name from number was more complex than it is now. With 1.1a, etoc leaves \l@chapter etc... unmodified (and unused) and only hijacks \contentsline.*

The other core part of etoc present from day one of the package is that it creates a tree-like structure of the sectioning levels present in the .toc file. But this is purely virtual, and handled via a notion of "level" and TEX conditionals. It could be fun to implement officially such a tree (where the children of a sectioning title are the sectioning levels at a greater depth such as subsections versus a section). Let us recall that LATEX provides zero means to know from a subsection for example, what is the title of the section containing it, or chapter, or part. To do this one has to create a really structured document which neither core LATEX nor the main document classes do. This remark was given for document body, but it also applies to the .toc data. But etoc adds at least some kind of follow-up to the successive encountered sectioning titles, and is thus able, to "on-the-fly" add some kind of structure and follow the chaining of levels. Ultimately this is why the \etocsetstyle offers {⟨*start*⟩} and {⟨*finish*⟩} parts in additions to {⟨*contents*⟩} (which I divided into a {⟨*prefix*⟩} and a {⟨*contents*⟩}). At some point one could imagine that a really *structured* document (in opposition to what core LATEX from thirty years ago up to nowadays realizes) would store in the .toc data directly a tree structure, where each node would have attributes name, number, page number, completely separated from any typesetting. Once this exists then basically etoc disappears. In brief, once etoc ideas will have permeated the society, it will disappear as its was born only to palliate the absence of real structure in the .toc file (which is sort of inherited from the absence of real structure in a LATEX document body).

1.1a implements a radical change to legacy core internals for compatibility with some (future) hyperref changes. In order to facilitate this overhaul, it required LATEX 2020-10-01 for the fourth argument to \contentsline lines in the .toc file to be always present.

At 1.2a, this requirement has been lifted so that we re-incorporated compatibility with old \contentsline fetching 4 or only 3 arguments depending on presence or not of hyperref. Fortunately the code refactoring completed at 1.2 made this easier. Testing on "old" LATEX was limited to one single check on a TL2019 install.

Also 1.2 use \expanded, we now test for its existence and provide alternative code if it is not provided by the engine.

TeXLive started producing LATEX format incorporating by default the ε-TEX extensions I think twenty years ago with TL2003, so let's require at least the LATEX of December of 2003 (but its is no guarantee that it will actually be with engine providing \ifdefined, \unless, \numexpr or \unexpanded or maybe others yet that we use).

```
\NeedsTeXFormat{LaTeX2e}[2003/12/01]
\ProvidesPackage{etoc}[2023/10/28 v1.2c Completely customisable TOCs (JFB)]
```

Gentle Info message in the log to mention no testing is done of current etoc on old LATEX installations.

```
\newif\ifEtoc@oldLaTeX
\@ifl@t@r\fmtversion{2020/10/01}
  {}
  {\Etoc@oldLaTeXtrue
   \PackageInfo{etoc}{Old LaTeX (\fmtversion) detected!\MessageBreak
   Since 1.1a (2023/01/14), etoc prefers LaTeX at least\MessageBreak
   as recent as 2020-10-01, for reasons of the .toc file,\MessageBreak
   and used to require it (from 1.1a to 1.2).\MessageBreak
   This etoc (1.2c) does not *require* it, but has not been\MessageBreak
   tested thoroughly on old LaTeX (especially if document\MessageBreak
   does not use hyperref) and retrofitting was done only\MessageBreak
   on basis of author partial remembrances of old context.\MessageBreak
   Reported}}
```

1.2 adds experimental support (only tested with standard classes and few packages) for \localtableofcontents, \localtableofcontents and \localtableofcontents. Did I say this is experimental? When LATEX will have added official hook to \addcontentsline, I will probably revise the way etoc hacks into it for this *experimental* functionality.

As it is **experimental**, I think we can all agree I don't have to spend too much space documenting it in user manual. So I shall be brief when I will get to it. And I will remain brief here too.

Now that etoc uses package options, I will use kvoptions as I am familiar with it. I understand upstream

LaTeX now has support for key-value input, put I simply have had no time to read the interface. Besides not sure it was there for the 2020-10-01 required release.

```
\RequirePackage{kvoptions}
\SetupKeyvalOptions{prefix=Etoc@}
\newif\ifEtoc@lof
\DeclareVoidOption{lof}{\Etoc@loftrue
  \PackageInfo{etoc}{Experimental support for \string\locallistoffigures.\MessageBreak
                     Barely tested, use at own risk}%
}
\newif\ifEtoc@lot
\DeclareVoidOption{lot}{\Etoc@lottrue
  \PackageInfo{etoc}{Experimental support for \string\locallistoftables.\MessageBreak
                     Barely tested, use at own risk}%
}
\@ifclassloaded{memoir}{
\PackageInfo{etoc}
   {As this is with memoir class, all `...totoc' options\MessageBreak
    are set true by default.  Reported}
\DeclareBoolOption[true]{maintoctotoc}
\DeclareBoolOption[true]{localtoctotoc}
\DeclareBoolOption[true]{localloftotoc}
\DeclareBoolOption[true]{locallottotoc}
}{
\DeclareBoolOption[false]{maintoctotoc}
\DeclareBoolOption[false]{localtoctotoc}
\DeclareBoolOption[false]{localloftotoc}
\DeclareBoolOption[false]{locallottotoc}
}
\DeclareBoolOption[true]{ouroboros}
```

The **deeplevels** option added at 1.2a. Adapted from an intial patch contributed by Matthew Trescott in the context of the Doxygen project. It sets the maximum level usable with \etocsetlevel (and never displayed) at 12 rather than 6.

Such an extension of the number of levels handled by **etoc** would have been much more cumbersome to achieve prior to the 1.2 refactoring which replaced usage of a booleans by a stack storage of the succession of levels seen in the .toc file (from which **etoc** creates virtual nesting structure), which actually is completely scalable and can handle unlimited number of levels. Basically we only needed to replace the formerly used \Etoc@@six@@ by a \Etoc@maxlevel set to have value 12 but this simple change caused also many modification to messages involved in **etoc**-user interactions, and a disproportionate quantity of time passed on updating the documentation, even though in a minimal way, and of course a minimal amount of testing but **etoc** is lacking a strong regression test suite which for lack of time has not yet been put into place. As it is so simple we could do some option maxlevel=<number> but well, there is no real need, because the extra potential levels do not cause any overhead to the actual **etoc** handling of tables of contents, so it is even tempting to adopt 12 (which is way beyond any realistic needs of a real document) as default, but this would break the documents in the wild which have used the advanced techniques based on hiding one or more levels via setting it at numerical depth 6.

```
\DeclareBoolOption[false]{deeplevels}
\DeclareDefaultOption{\PackageWarning{etoc}{Option `\CurrentOption' is unknown.}}
\ProcessKeyvalOptions*
\DisableKeyvalOption[action=error,package=etoc]{etoc}{lof}
\DisableKeyvalOption[action=error,package=etoc]{etoc}{lot}
\DisableKeyvalOption[action=error,package=etoc]{etoc}{deeplevels}
```

The real verbosity problem of LaTeX is not so much the log, which should actually be as detailed as possible (and the default \errorcontexlines setting of LaTeX is to my view misguided), but the humongous output to console, most of it being perfectly useless in 99% of cases. Despite \PackageInfo adding stuff to the log only, I finally decided not to use this next hunk.

```
% \PackageInfo{etoc}{Status of options at loading time:\MessageBreak
```

```
%   lof = \ifEtoc@lof true\else false\fi\MessageBreak
%   lot = \ifEtoc@lot true\else false\fi\MessageBreak
%   maintototoc   = \ifEtoc@maintoctotoc true\else false\fi\MessageBreak
%   localtoctotoc = \ifEtoc@localtoctotoc true\else false\fi\MessageBreak
%   localloftotoc = \ifEtoc@localloftotoc true\else false\fi\MessageBreak
%   locallottotoc = \ifEtoc@locallottotoc true\else false\fi\MessageBreak
%   ouroboros     = \ifEtoc@ouroboros true\else false\fi\MessageBreak
%   deeplevels    = \ifEtoc@deeplevels true\else false\fi\@gobble
% }
```

For many many years etoc had no options. Let's be modern and provide an \etocsetup.

```
\def\etocsetup#1{\setkeys{etoc}{#1}}
\def\etocifmaintoctotoc{\ifEtoc@maintoctotoc
                          \expandafter\@firstoftwo
                        \else
                          \expandafter\@secondoftwo
                        \fi}
\def\etociflocaltoctotoc{\ifEtoc@localtoctotoc
                          \expandafter\@firstoftwo
                        \else
                          \expandafter\@secondoftwo
                        \fi}
\def\etociflocalloftotoc{\ifEtoc@localloftotoc
                          \expandafter\@firstoftwo
                        \else
                          \expandafter\@secondoftwo
                        \fi}
\def\etociflocallottotoc{\ifEtoc@locallottotoc
                          \expandafter\@firstoftwo
                        \else
                          \expandafter\@secondoftwo
                        \fi}
\RequirePackage{multicol}
\def\etoc@{\etoc@}
\long\def\Etoc@gobtoetoc@ #1\etoc@{}
\newtoks\Etoc@toctoks
\def\Etoc@par{\par}
\def\etocinline{\def\Etoc@par{}}
\let\etocnopar\etocinline
\def\etocdisplay{\def\Etoc@par{\par}}
```

\etocglobaldefs should be used only for special things such as TOC as a table; it should be put in a group to limit its scope. If used in the preamble, it must come *after* tableof if the latter is loaded too.

```
\let\Etoc@global\@empty
\def\etocglobaldefs{\let\Etoc@global\global\let\tof@global\global}
\def\etoclocaldefs {\let\Etoc@global\@empty\let\tof@global\@empty}
```

Some have been renamed at 1.2.

```
\newif\ifEtoc@numbered
\newif\ifEtoc@hyperref
\newif\ifEtoc@parskip
\newif\ifEtoc@tocwithid
\newif\ifEtoc@standardlines
```

These next three added at 1.2. The latter two for handling compatibility layer with tocloft and tocbibind.

```
\newif\ifEtoc@etocstyle
\newif\ifEtoc@classstyle
\newif\ifEtoc@keeporiginaltoc
\newif\ifEtoc@skipprefix
```

```
\newif\ifEtoc@isfirst
\newif\ifEtoc@localtoc
\newif\ifEtoc@skipthisone
\newif\ifEtoc@stoptoc
\newif\ifEtoc@notactive
\newif\ifEtoc@mustclosegroup
\newif\ifEtoc@isemptytoc
\newif\ifEtoc@checksemptiness
\def\etoccheckssemptiness       {\Etoc@checksemptinesstrue }
\def\etocdoesnotcheckemptiness {\Etoc@checksemptinessfalse }
\newif\ifEtoc@notocifnotoc
\def\etocnotocifnotoc {\Etoc@checksemptinesstrue\Etoc@notocifnotoctrue }
\newcounter{etoc@tocid}
\def\Etoc@tocext{toc}
\def\Etoc@lofext{lof}
\def\Etoc@lotext{lot}
\let\Etoc@currext\Etoc@tocext
\def\etocifislocal{\ifEtoc@localtoc\expandafter\@firstoftwo\else
                                    \expandafter\@secondoftwo\fi
                   }
\def\etocifislocaltoc{\etocifislocal{\ifx\Etoc@currext\Etoc@tocext
                                      \expandafter\@firstoftwo\else
                                      \expandafter\@secondoftwo\fi}%
                                     {\@secondoftwo}%
                      }
\def\etocifislocallof{\etocifislocal{\ifx\Etoc@currext\Etoc@lofext
                                      \expandafter\@firstoftwo\else
                                      \expandafter\@secondoftwo\fi}%
                                     {\@secondoftwo}%
                      }
\def\etocifislocallot{\etocifislocal{\ifx\Etoc@currext\Etoc@lotext
                                      \expandafter\@firstoftwo\else
                                      \expandafter\@secondoftwo\fi}%
                                     {\@secondoftwo}%
                      }
```

Formerly a LaTeX counter `etoc@tocdepth` was declared here but at 1.2 it has been replaced by macro storage.

```
\expandafter\def\csname Etoc@-3@@\endcsname {-\thr@@}
\expandafter\def\csname Etoc@-2@@\endcsname {-\tw@}
\expandafter\let\csname Etoc@-1@@\endcsname \m@ne
\expandafter\let\csname Etoc@0@@\endcsname  \z@
\expandafter\let\csname Etoc@1@@\endcsname  \@ne
\expandafter\let\csname Etoc@2@@\endcsname  \tw@
\expandafter\let\csname Etoc@3@@\endcsname  \thr@@
\expandafter\chardef\csname Etoc@4@@\endcsname  4
\expandafter\chardef\csname Etoc@5@@\endcsname  5
\expandafter\chardef\csname Etoc@6@@\endcsname  6
```

**deeplevels** option needs a few extra declarations.

```
\ifEtoc@deeplevels
  \expandafter\chardef\csname Etoc@7@@\endcsname  7
  \expandafter\chardef\csname Etoc@8@@\endcsname  8
  \expandafter\chardef\csname Etoc@9@@\endcsname  9
  \expandafter\chardef\csname Etoc@10@@\endcsname  10
  \expandafter\chardef\csname Etoc@11@@\endcsname  11
  \expandafter\chardef\csname Etoc@12@@\endcsname  12
\fi
```

1.2a adds `\Etoc@maxlevel` which replaces formerly used `\Etoc@@six@@`. It adds also `\etocthemaxlevel`

as user interface to its explicit numerical value.

```
\expandafter\let\expandafter\Etoc@maxlevel
  \csname Etoc@\ifEtoc@deeplevels12\else6\fi @@\endcsname
\edef\etocthemaxlevel{\number\Etoc@maxlevel}
```

For \etocsetlevel, 1.2a uses a comparison with \Etoc@minf so that only levels suitable for the document class can be used. So etoc now will not define unneeded 'book' levelname with numerical level -2 except if with memoir.

```
\@ifclassloaded{memoir}{\def\Etoc@minf{-\thr@@}}{\def\Etoc@minf{-\tw@}}
\let\Etoc@none@@  \Etoc@minf
\expandafter\let\expandafter\Etoc@all@@
  \csname Etoc@\ifEtoc@deeplevels11\else5\fi @@\endcsname
```

\Etoc@dolevels will hold the names of all declared levels (independently of their numerical levels), in a traditional LaTeX \do separated manner, except that for some reason I used \Etoc@do. TODO: maybe use \do.

```
\let\Etoc@dolevels\@empty
\def\Etoc@newlevel #1{\expandafter\def\expandafter\Etoc@dolevels\expandafter
        {\Etoc@dolevels\Etoc@do{#1}}}
```

1.2 has done some refactoring here (and above), reducing the number of definitions. Formerly each defined sectioning level got two macros assigned to it: one holding the numerical level (as chardef or count, or even tokens for -\tw@), the other in a textual representation such as minusone. The latter do not get defined anymore.

MEMO: the legacy name space here is rather poor as anything can happen after \Etoc@ with the sole and only characteristic that it terminates with @@. So this means no package control sequence is allowed to end in @@ else it is in risk of being overwritten, were it not for the special filtering done by \etocsetlevel. For matters of \etocsetnexttocdepth allowing both name and numerical arguments, we employed the cheap device to define ourselves special levels named by explicit integers for internal usage, hence we need to filter them out here from being redefined. We also filter out names starting with @ for other reasons, but still it was not true at 1.2 that the package could safely use \Etoc@@...@@ macros, as \etocsetlevel creates since then also a \Etoc@@⟨*name*⟩@@. There was in particular a chardef \Etoc@@six@@ hence if user chose "six" as level name \Etoc@@six@@ acquired a new meaning and this could cause strange symptoms, the main one being the disappearance from TOCs of levels at same numerical depth than the one now given by the modified \Etoc@@six@@. (There was no \Etoc@six@@ so the bug was really only one of 1.2 not of earlier releases).

1.2 adds here \etocifunknownlevelTF and \etoclevel for a higher level interface, which may be used by third parties such as the yathesis class and will allow etoc at some point to modify its internal naming conventions.

The -\tw@ case needed for \Etoc@minf (or for memoir) means that two expansions of \etoclevel do not always deliver a single token, it is at any rate always self-delimiting in assignments and \ifnum tests. I hesitated using only explicit digit tokens by the way.

The dummy sectioning levels "all" and "none" play a special role and will be declared by \etocifunknownlevelTF as "known".

TODO: is it worthwile to still allow '7' to '12' as level "names"? This is done here for perfact backward compatibility but they have to be excluded under deeplevels option, and it would be better to do something not depending on whether that option is used or not.

```
\ifdefined\expanded
  \def\etocsetlevel#1#2{\expanded{\noexpand\etoc@setlevel{#1}{#2}}}%
\else
  \def\etocsetlevel#1#2{{\edef\Etoc@tmp{\noexpand\etoc@setlevel{#1}{#2}}\expandafter}\Etoc@tmp}%
\fi
\def\etoc@setlevel#1#2{%
  \edef\Etoc@tmp{\the\numexpr#2}%
  \if1\ifnum\Etoc@tmp>\Etoc@maxlevel0\fi\unless\ifnum\Etoc@minf<\Etoc@tmp;\fi1%
    \ifEtoc@deeplevels
      \in@{.#1,}{.none,.all,.figure,.table,.-3,.-2,.-1,.0,.1,.2,.3,.4,.5,.6,%
                                          .7,.8,.9,.10,.11,.12,}%
    \else
```

```
        \in@{.#1,}{.none,.all,.figure,.table,.-3,.-2,.-1,.0,.1,.2,.3,.4,.5,.6,}%
      \fi
```

Letter or other agnostic test. First time I ever use `\@car` as I never paid attention to its existence and in more macro-coding intensive context such as xint I use my own.

```
      \ifin@\else\if\@car#1\@nil @\in@true\fi\fi
      \ifin@
         \PackageWarning{etoc}
            {Sorry, but `#1' is forbidden as level name.\MessageBreak
             \if\@car#1\@nil @%
                (because of the @ as first character)\MessageBreak\fi
             Reported}%
      \else
        \etocifunknownlevelTF{#1}{\Etoc@newlevel{#1}}{}%
        \expandafter\let\csname Etoc@#1@@\expandafter\endcsname
                        \csname Etoc@\Etoc@tmp @@\endcsname
```

This is to allow "to toc" entries to not break the core mechanism used to let local tables of contents (now also lists of) know their scope, in case for example we have one after the other `\localtableofcontents`, `\locallistoffigures`, `\locallistoftables` and each has created its "to toc" entry: main TOC will handle the `@<division>` as aliases to `<division>` and local TOCs and Lists Of will ignore them.

So we create `@<levelname>` as aliases to `<levelname>`. This is really needed only for the level names actually involved by "to toc" thing, but let's do it systematically to avoid implementation complications.

```
        \expandafter\edef\csname Etoc@@#1@@\endcsname
                        {\expandafter\noexpand\csname Etoc@#1@@\endcsname}%
```

As "to toc" will use some `\addcontentsline`, we must make hyperref happy and also define the suitable `toclevel@` prefixed extra macros.

```
        \expandafter\edef\csname toclevel@@#1\endcsname
                        {\expandafter\noexpand\csname toclevel@#1\endcsname}%
```

But we don't set hyperref's `\toclevel@#1` to be numerically `\Etoc@tmp`, with some hesitation. The level defined by `\etocsetlevel` should only be for interpretation by **etoc** for the contents of the `.toc` file. If it also influences how hyperref hooks into `\section` and like commands influence the PDF bookmarks, unexpected results could follow. It is up to user to set-up by themself possibly needed extra hyperref configuration in this regard.

```
      \fi
   \else
      \PackageWarning{etoc}
         {Argument `\detokenize{#2}' of \string\etocsetlevel\space should
          represent one of\MessageBreak
          \ifnum\Etoc@minf=-\thr@@-2, \fi-1, 0, 1, 2, \ifEtoc@deeplevels ...\else3, 4\fi,
          \the\numexpr\Etoc@maxlevel-1, or \number\Etoc@maxlevel\space
          but evaluates to \Etoc@tmp.\MessageBreak
          The level of `#1' will be set to \number\Etoc@maxlevel.\MessageBreak
          Tables of contents will ignore `#1' as long\MessageBreak
          as its level is \number\Etoc@maxlevel\space (=\string\etocthemaxlevel).%
          \MessageBreak
          Reported}%
      \etocifunknownlevelTF{#1}{\Etoc@newlevel{#1}}{}%
      \expandafter\let\csname Etoc@#1@@\endcsname\Etoc@maxlevel
   \fi
}
```

Maybe let it first use `\etocifunknownlevelTF` and raise in the "unknown" branch a suitable expandable error message (and return say `-3`)?

I may also need for internal usage a variant for only numerical `#1` which be submitted to a `\the\numexpr`. Not so far.

Unfortunately the name "level" here does not convey to user the fact that the argument of `\etocifunknownlevelTF` is a "name" not a numerical thing.

```
\def\etoclevel#1{\csname Etoc@#1@@\endcsname}
\def\etocthelevel#1{\number\csname Etoc@#1@@\endcsname}
\def\etocifunknownlevelTF#1{\@ifundefined{Etoc@#1@@}}
\@ifclassloaded{memoir}{\etocsetlevel{book}{-2}}{}
\etocsetlevel{part}{-1}
\etocsetlevel{chapter}{0}
\etocsetlevel{section}{1}
\etocsetlevel{subsection}{2}
\etocsetlevel{subsubsection}{3}
\etocsetlevel{paragraph}{4}
\etocsetlevel{subparagraph}{5}
```

Prior to 1.2, only under class memoir was \etocsetlevel used with appendix. It does not seem to hurt to do it generally, with a check whether document class provides chapters.

```
\ifdefined\c@chapter
  \etocsetlevel{appendix}{0}
\else
  \etocsetlevel{appendix}{1}
\fi
```

The "to toc" mechanism will add to the .toc file \contentsline entries with first argument such as @section or @subsection. They will generally behave as section, resp. subsection, etc... This special mark-up is needed for "to toc" inclusions to not break the etoc mechanism for delimiting the scope of local tables of contents.

```
\def\Etoc@do#1{\@namedef{l@@#1}{\csname l@#1\endcsname}}
\Etoc@dolevels
```

We do not issue \etocsetlevel{figure}{6} (or {12}) as anyhow the \etocsetlevel interface forbids figure and table as first argument.

```
\let\Etoc@figure@@\Etoc@maxlevel
\let\Etoc@table@@ \Etoc@maxlevel
```

1.09g adapts to hyperref depending on whether the latter is at 7.00u or earlier. Indeed internal changes to hyperref at 7.00u broke etoc. Thanks to Denis Bitouzé for reporting the issue.

1.1a radically simplifies matters at etoc core, and if these changes had been in place earlier there would have been no incompatibility with the hyperref 7.00u release.

At 1.2a we drop the 1.1a requirement of LaTeXfrom 2020-10-01 and must cater for \contentsline with 3 or 4 arguments. Fortunately, the code refactorings engaged at 1.1a and completed at 1.2 made such a retro-fit relatively simple. Let's hope nothing was overlooked, though.

But we can not assume anymore \@gobblethree exists... hesitation here whether to use a \Etoc@gobblethree or directly name it \@gobblethree.

```
\let\Etoc@gobblethreeorfour\@gobblefour
\ifdefined\@gobblethree
  \let\Etoc@gobblethree\@gobblethree
\else
  \long\def\Etoc@gobblethree#1#2#3{}%
\fi
\AtBeginDocument{%
\@ifpackageloaded{parskip}{\Etoc@parskiptrue}{}%
\@ifpackageloaded{hyperref}
    {\Etoc@hyperreftrue}
    {\ifEtoc@oldLaTeX
        \let\Etoc@gobblethreeorfour\Etoc@gobblethree
        \let\Etoc@etoccontentsline@fourargs\Etoc@etoccontentsline@
        \long\def\Etoc@etoccontentsline@#1#2#3{%
            \Etoc@etoccontentsline@fourargs{#1}{#2}{#3}{}%
        }%
     \fi
    }%
```

```
}
\def\etocskipfirstprefix {\global\Etoc@skipprefixtrue }
```

Start of heart of etoc's hacks into the execution of the .toc file commands. It goes via a redefinition of \contentsline which will launch an extraction process leading to the construction of \etocname, \etocnumber, and \etocpage, then the styles as defined by user via \etocsetstyle get executed in accordance to the levels.

In passing etoc is witness to the linear succession of sectioning levels and executes the {⟨*start*⟩} and {⟨*finish*⟩} parts of each used level at the right time (they are rather called "begin" and "end" in the code though).

1.2 did a complete rewrite of how etoc creates virtually a nesting structure out of the flat succession of the \contentsline's for various levels. Ever since the first release this was based on using boolean flags, one for each level. The flag was on if the level had been seen, hence its "begin" macro executed, but not yet its "end" macro. This is now replaced by a stack storage \Etoc@stackofends which is simply, e.g. {2}{0}{-3}{}, to mean that first a chapter (0) was seen then a subsection (2). The {-3}{} trail is for matters of avoiding brace removal in the implementation next (I could have replaced it by a single token being numerically -3). This structure made implementing the deeplevels painless (but time-consumingly documented) at 1.2a.

When a new level is encountered and set in \Etoc@level, it is compared to the left most entry in the stack. If higher, the "begin" macro is executed and the \Etoc@level is pushed (as a digit with possibly a minus sign) to the left of the stack to record that the "end" macro is now on the queue for execution (sic). If equal, nothing has to be done, if lower, the "end" macro of the left-most stored level is executed then this level is removed, and one proceeds with the next one, etc... legacy code was using a bunch of TeX conditionals rather, and I recall how in 2012 I was unfamiliar with its strange syntax and had lots of troubles; once I got something working it got frozen and basically did not change since then. The 1.2 implementation has replaced all of this by maintenance of a single "stack", which is more economical in terms of used macros and potentially more scalable too.

About \Etoc@level I have hesitated using only digit and sign tokens, but it is currently let to some \chardef (in exceptional memoir case of book level it can expand to -\tw@, and there is also the \m@ne count). 1.2 thus keeps defining such \chardef's but has at least dropped auxiliary macros (see the definitions prior to \etocsetlevel) which held some alphabetical denotations such as "minusone" or "zero", and the *begin*, *prefix*, *content* and *end* macros associated with each level now use a digit (and sign perhaps) (see \etocsetstyle). With \Etoc@level already storing directly such a digit, one would avoid a \number or \the\numexpr at some places, but would have to be more careful in the various \ifnum.

This (always globally defined) \Etoc@level must now never be set to the numerical value 6 (or 12 if option deeplevels): it is legal to add to the .toc file dummy sectioning levels associated to the maximal numerical level \etocthemaxlevel (such dummy sectioning will be ignored but can be assigned locally a non-ignored level for special effects) and if \Etoc@level was, as prior to 1.2, systematically set to the numerical level of the last seen \contentsline, this could cause \Etoc@@startlocaltoc to fail to correctly set the "local top". Indeed it now only has \Etoc@level at his disposal, the legacy boolean flags being gone.

The \Etoc@isfirsttrue was formally incorporated as last token of the "begin" macros as defined by \etocsetstyle, but has been displaced by 1.2 to the code below.

```
\def\Etoc@updatestackofends#1\etoc@{\gdef\Etoc@stackofends{#1}}
\def\Etoc@stackofends{{-3}{}}
\def\Etoc@doendsandbegin{%
    \expandafter\Etoc@traversestackofends\Etoc@stackofends\etoc@
}
```

We compare the new level with those for which the {⟨*start*⟩} parts of the \etocsetstyle declarations have been executed to decide if it is time to execute their {⟨*finish*⟩} parts. In passing we set the boolean \ifEtoc@isfirst which is needed for \etociffirst and \etocxiffirst.

```
\def\Etoc@traversestackofends#1{%
  \ifnum#1>\Etoc@level
    \csname Etoc@end@#1\endcsname
    \expandafter\Etoc@traversestackofends
  \else
    \Etoc@traversestackofends@done{#1}%
  \fi
```

```
}
\def\Etoc@traversestackofends@done#1#2{#2%
  \ifnum#1<\Etoc@level
    \csname Etoc@begin@\the\numexpr\Etoc@level\endcsname
    \Etoc@global\Etoc@isfirsttrue
    \edef\Etoc@tmp{{\the\numexpr\Etoc@level}}%
  \else
    \Etoc@global\Etoc@isfirstfalse
    \let\Etoc@tmp\@empty
  \fi
  \expandafter\Etoc@updatestackofends\Etoc@tmp{#1}%
}
```

Ever since the first release of etoc, the code has to be careful that the \Etoc@end@<level> user defined macros may close groups. This is the reason why some assignments have to be done globally (2015/03/08).

1.1a of 2023/01/14 implements a radical change to legacy core internals for compatibility with (attow future) hyperref. Formerly \Etoc@etoccontentsline@ fetched only the first argument. It now also fetches all four (the fourth argument of \contentsline is always present since LaTeX 2020-10-01). The \Etoc@lxyz used to receive only the two arguments (possibly hacked by hyperref) of \l@chapter, \l@section, etc..., (these macros had been \let to \Etoc@lxyz), and examined them to see if they were carrying hyperlinking data. The 1.1a and later version receives as third argument the fourth one of \contentsline, i.e. the hyperlinking target, and adds the hyperlinking according to the status of hyperref's \Hy@linktoc.

This is a breaking change if a user hacked \contentsline to do some specific pre-processing of the data, as this extra will now be ignored. The kind of hack one can think of is perhaps to pre-process the section title to turn it into uppercase, this kind of things, but why do such things when one is using etoc which precisely provides a general interface for such customization? Besides as the LaTeX legacy set-up already mixes up in various ways name and number in the second argument of \contentsline, doing such hacks in a non-breaking way was not easy, and could have broken etoc easily anyhow.

The major hacker was hyperref... Indeed in 2012 when I started work on etoc, it was not clear to me how hyperref would end up using the fourth argument of \contentsline and I did not want to spend too much time tracing hyperref code. So I simply let hyperref do its stuff, and added specific post-processing branches to unravel it. It looks quite dumb in retrospect (at this time the .toc file lines with \contentsline had either three or four arguments which contributed for the design decisions back then).

All hyperref specific branches are now gone, replaced by extra code added depending on the status of the \ifEtoc@hyperref boolean. We also check the \Hy@linktoc \chardef status and, imitating hyperref, do not hyperlink the page number argument if it turns out empty. This maintains backwards-compatibility with earlier releases of etoc.

TODO: should this code ensure #1 is actually a legit level name and if not issue some nice error message? This would add system-level overhead only for careless people who do not read docs...

```
\def\Etoc@etoccontentsline #1{%
  \let\Etoc@next\Etoc@gobblethreeorfour
  \ifnum\csname Etoc@#1@@\endcsname=\Etoc@maxlevel
  \else
   \Etoc@skipthisonefalse
```

MEMO: the mechanism added to make added toc entries from "list of" titles invisible to the local tocs, goes via such a #1 starting with an @. In that case \Etoc@#1@@ is not a \chardef but expands to one (or to a count or -\tw@ perhaps, or a \numexpr...\relax).

```
    \global\expandafter\let\expandafter\Etoc@level\csname Etoc@#1@@\endcsname
```

The trick goes through a slight overhead here to filter out such special "@"-level names and not make them update what will serve as top level for local tocs or listsof. Formerly this was managed by booleans, then for 1.1d (released as 1.2) it got replaced by sole usage of \Etoc@level, and finally a specific \Etoc@virtualtop which gets its updates here.

```
    \if @\@car#1\@nil\else\global\let\Etoc@virtualtop\Etoc@level\fi
    \ifEtoc@localtoc
     \ifEtoc@stoptoc
       \Etoc@skipthisonetrue
     \else
```

```
      \ifEtoc@notactive
        \Etoc@skipthisonetrue
      \else
       \unless\ifnum\Etoc@level>\etoclocaltop
         \Etoc@skipthisonetrue
         \global\Etoc@stoptoctrue
       \fi
      \fi
     \fi
    \fi
    \ifEtoc@skipthisone
    \else
     \unless\ifnum\Etoc@level>\c@tocdepth
      \ifEtoc@standardlines
          \let\Etoc@next\Etoc@savedcontentsline
      \else
          \let\Etoc@next\Etoc@etoccontentsline@
      \fi
     \fi
    \fi
   \fi
   \Etoc@next{#1}%
}
```

Hesitation at 1.2 about having \Etoc@level being always explicit digit and perhaps negative sign. For now is a \chardef or \m@ne \count (perhaps -\tw@ also).

```
\def\Etoc@etoccontentsline@ #1#2#3#4{%
    \Etoc@doendsandbegin
    \Etoc@global\edef\Etoc@prefix  {\expandafter\noexpand
         \csname Etoc@prefix@\the\numexpr\Etoc@level\endcsname }%
    \Etoc@global\edef\Etoc@contents{\expandafter\noexpand
         \csname Etoc@contents@\the\numexpr\Etoc@level\endcsname }%
    \ifEtoc@skipprefix \Etoc@global\def\Etoc@prefix{\@empty}\fi
    \global\Etoc@skipprefixfalse
    \Etoc@lxyz{#2}{#3}{#4}%
    \Etoc@prefix
    \Etoc@contents
}
```

A **breaking change** is made at 1.1a: \etoclink will always create an hyperlink, even in case of hyperref being (possibly locally) configured to obey linktoc=none. Formerly, in such case, \etoclink added no hyperlink because etoc identified the hyperlink target from the hyperref hacked arguments of \l@section et al, rather than picking it from the fourth argument of \contentsline.

Another **breaking change** (documented only at 1.1b): all three of \etocthelinkedname, \etocthelinkednumber, and \etocthelinkedpage are always hyperlinks (for the latter, only if page number is not empty to match hyperref ways). Formerly they obeyed the linktoc status, somewhat counterintuitively, but this meant that \etocname etc... were their robust variants, which meant one could store easily for later usage (see the documentation examples with "treetoks") their precise meaning. The breaking change happened in part because I was fooled myself by the macro names, and refactored the code in two steps separated by months so in second step I forgot I had only provisory code. And I decided finally to keep the breaking change.

Under linktoc=page option, hyperref has a "feature" to add one level of bracing to first argument of the \l@section etc macros. So for etoc <1.1a this meant some extra work to dig into such a possible brace pair to check if the entry was numbered. At 1.1a, the hyperref modified \contentsline is not executed, hence there is no such complication. But the trimming of the now unneeded branches was not yet done at 1.1a, 1.1b, and got completed only at 1.1c, together with some renamings and refactoring.

```
\def\Etoc@lxyz #1#2#3{%
    \ifEtoc@hyperref
```

```
      \Etoc@global\def\etocthelink##1{\hyperlink{#3}{##1}}%
   \else
      \Etoc@global\let\etocthelink\@firstofone
   \fi
   \Etoc@global\def\etocthepage {#2}%
```

Prior to 1.1a an hyperlink was incorporated into \etocthelinkedpage only if hyperref added an hyperlink to the page number, i.e. under linktoc=page or linktoc=all (and a non empty page number). With 1.1a, the hyperlink is always added (if a non empty page number).

```
   \ifEtoc@hyperref
      \ifx\etocthepage\@empty
        \Etoc@global\let\etocthelinkedpage\@empty
      \else
        \Etoc@global\def\etocthelinkedpage{\hyperlink {#3}{#2}}%
      \fi
   \else
      \Etoc@global\let\etocthelinkedpage\etocthepage
   \fi
```

Define \etocthename but this will perhaps be adjusted later if it is found out that the entry was numbered.

```
   \Etoc@global\def\etocthename{#1}%
```

Now we check if the entry is numbered and disentangle the number from the name to define correctly \etocthename and \etocthenumber. The delimiter tokens were modified at 1.1c for a slight optimization. And secondary macros have less to do since the 1.1a initiated refactoring.

```
   \futurelet\Etoc@getnb@token\Etoc@@getnb #1\hspace\etoc@
```

Even if \etocthenumber was let to \@empty, it may happen in special circumstances (related to KOMA-script, see below) that \etocthename got redefined. We will thus use its current contents to define appropriately \etocthelinkedname.

In presence of hyperref we let always \etocthelinkedname and \etocthelinkednumber (for a numbered entry) carry an hyperlink. This is a **breaking change** at 1.1a: formerly if the TOC (or the specific entry in the .toc file, as it is always possibly to inject \hypersetup) was typeset under linktoc=none or linktoc=page status, then no hyperlinks were incorporated. This is how \etocthename and \etocthenumber are configured, but \etocthelinkedname and \etocthelinkednumber will since 1.1a always be hyperlinked in presence of hyperref.

```
   \ifEtoc@hyperref
     \def\Etoc@tmp##1##2{\Etoc@global\def##2{\hyperlink{#3}{##1}}}%
     \expandafter\Etoc@tmp\expandafter{\etocthename}\etocthelinkedname
     \ifEtoc@numbered
        \expandafter\Etoc@tmp\expandafter{\etocthenumber}\etocthelinkednumber
     \else
        \Etoc@global\let\etocthelinkednumber\@empty
     \fi
   \else
     \Etoc@global\let\etocthelinkedname  \etocthename
     \Etoc@global\let\etocthelinkednumber\etocthenumber
   \fi
```

Defaults in absence of hyperref. We externalize to another macro the hyperref case switch.

```
   \Etoc@global\expandafter\let\csname etoclink \endcsname  \etocthelink
   \Etoc@global\expandafter\let\csname etocname \endcsname  \etocthename
   \Etoc@global\expandafter\let\csname etocnumber \endcsname\etocthenumber
   \Etoc@global\expandafter\let\csname etocpage \endcsname  \etocthepage
   \ifEtoc@hyperref
     \Etoc@lxyz@linktoc
   \fi
}
```

In presence of hyperref, **etoc** 1.1a imports the hyperref own logic and tests \Hy@linktoc to decide if *name*, *number* and *page* get hyperlinks. This adds a dependency that \Hy@linktoc should exist and have the expected interpretation.

**MEMO:** *Matters of tagging will have to wait for LaTeX itself to show me what it does in \l@section etc... so that I can imitate.*

Updated 2023/02/26: this is now partly available and gave me an idea of what will be needed here. As the **etoc** way goes through none of the LaTeX hook points, I will have to add the suitable \UseHook at various places, after having stored the \contentsline original arguments in the same macros as the LaTeX new kernel code will do.

Some difficulties though in perspective as **etoc** separates name and number and has no concept akin to \@dottedtocline. Also if the user employs \etocsetstyle with \l@section, hooks may be executed twice if I put them not in \etocname but in the {⟨*prefix*⟩} and {⟨*content*⟩} parts for each level.

```
\def\Etoc@lxyz@linktoc{%
    \ifcase\Hy@linktoc
```

none: nothing to do

```
    \or
```

section (aka name for etoc): link name and number

```
      \Etoc@global\expandafter\let\csname etocname   \endcsname\etocthelinkedname
      \Etoc@global\expandafter\let\csname etocnumber \endcsname\etocthelinkednumber
    \or % page
      \Etoc@global\expandafter\let\csname etocpage   \endcsname\etocthelinkedpage
    \else % all
      \Etoc@global\expandafter\let\csname etocname   \endcsname\etocthelinkedname
      \Etoc@global\expandafter\let\csname etocnumber \endcsname\etocthelinkednumber
      \Etoc@global\expandafter\let\csname etocpage   \endcsname\etocthelinkedpage
    \fi
}
```

Now for the disentangling of the "number" from the "name".

At some point we will pick up the first token and check if it is \numberline or like token to identify a numbered entry. But this step can cause brace removal so we \futurelet to peek ahead.

Prior to 1.1a it could be possible that the first token following \Etoc@@getnb was an opening brace and nevertheless the entry was numbered, because of a hyperref "feature" in case of linktoc=page option. But at 1.1a, **etoc** handles directly the argument of \contentsline so the presence of an opening brace implies the entry is not numbered. For some reason 1.1a kept the extra code to check in case the next token was an opening brace whether the whole entry was braced, which would have been indicative in the past (but not at 1.1a) of a linktoc=page context (**etoc** prior to 1.1a never tested the value of \Hy@linktoc as it did not want to be dependent on details of hyperref handling of options). This legacy, now superfluous, code branch is removed at 1.1c, bringing some simplification here, in particular the removal of an \ifEtoc@bracedname boolean. Also, when branching from here to the \Etoc@getnb@nonbr, we won't need to check if the entry had a special "Part" syntax, which is another simplification.

The @nonbr means "no number", and is not to be misinterpreted as "non braced"... (this was more confusing to the author on return to **etoc** code, when it still had branches handling issues described above with an extra brace pair).

```
\def\Etoc@@getnb {%
    \let\Etoc@next\Etoc@getnb
    \ifx\Etoc@getnb@token\@sptoken\let\Etoc@next\Etoc@getnb@nonbr\fi
    \ifx\Etoc@getnb@token\bgroup  \let\Etoc@next\Etoc@getnb@nonbr\fi
    \Etoc@next
}
```

1.08n tries to handle reasonably the \nonumberline of KOMA-script. If it expands to \numberline{}, **etoc** will consider the line numbered with an empty number (afaict, the meaning of \nonumberline is either empty or \numberline{}). This got modified at 1.1c (see below).

At 1.09f we get rid of the \nonumberline from inside \etocthename when it has empty meaning, so the expansion of the latter can safely be delayed by custom section styles (for example if the build up some token list to be executed later not immediately). 1.1c fixes a regression committed at 1.1a: for a \nonumberline with empty meaning the \etocthelinkedname did not end up hyperlinked.

A change, almost a bug fix, but the former behavior was actually deliberate, at 1.1c regarding the KOMA-script \nonumberline token: formerly, when it expanded to \numberline{} (this can happen only when the TOC is typeset in compatibility mode for the global display) then \ifEtoc@numbered was set to true. But this is only a KOMA-script typesetting thing, and should not have influenced etoc's decisions when its (user or package) own line styles are used: at 1.1c it is thus decided that in such circumstances the \etocifnumbered will pick the false branch, and the empty \etocthelinkednumber will not be hyperlinked.

No brace removal of the #1 here a priori possible because we took care to check that \Etoc@getnb was not followed by either a space or an opening brace.

The \Etoc@@getit branch for "Parts" used to be executed from inside \Etoc@lxyz. At 1.1c we jump directly to it from here.

```
\def\Etoc@getnb #1{%
    \in@{#1}{\numberline\chapternumberline\partnumberline\booknumberline}%
    \ifin@
       \let\Etoc@next\Etoc@getnb@nmbrd
    \else
       \ifnum\Etoc@level=\m@ne
           \let\Etoc@next\Etoc@@getit
       \else
           \let\Etoc@next\Etoc@getnb@nonbr
       \fi
```

Remove a KOMA-script \nonumberline token if present and process the entry always as not numbered (see above comments). Prior to 1.1c, the code branched according to the meaning of the \nonumberline token, which was a bit silly.

```
       \in@{#1}{\nonumberline}%
       \ifin@
         \let\Etoc@next\Etoc@getnb@nonumberline
       \fi
    \fi
    \Etoc@next #1%
}
```

Prior to 1.1a, \etocthelinkedname and \etocthelinkednumber (for a numberd entry) were defined to carry links only if an hyperlink was actually found, now they are defined in \Etoc@lxyz to always provide the hyperlinking to the target title in the document.

1.1a and 1.1b still had here some superfluous code which was trimmed at 1.1c.

Also, 1.1c fixes here a brace removal bug (which had always been there I guess): if the numbered heading title was braced one level of bracing was removed. The bug had no effect in a document using hyperref (and its default linktoc setting) as the \hyperlink wrapper limited the scope. But in a document without hyperref it would have been seen with input such as \section{{\color{blue}Stuff}}.

```
\def\Etoc@getnb@nmbrd #1#2{%
    \Etoc@global\Etoc@numberedtrue
    \Etoc@global\def\etocthenumber {#2}%
    \Etoc@getnb@nmbrd@getname\@empty
}%
```

We added an \@empty token to prevent brace removal.

```
\def\Etoc@getnb@nmbrd@getname #1\hspace\etoc@ {%
    \Etoc@global\expandafter\def\expandafter\etocthename\expandafter{#1}%
}
```

Not numbered entry.

```
\def\Etoc@getnb@nonbr #1\etoc@ {%
    \Etoc@global\Etoc@numberedfalse
    \Etoc@global\let\etocthenumber \@empty
```

```
}
```

Special KOMA branch: #1 starts with \nonumberline (prior to 1.1c the #1 would have already lost this token, and this branch was executed only in case \nonumberline had empty meaning). We need to remove this token from #1 and redefine \etocthename.

1.1a code still had here some complications with a "braced name" branch which was in fact never executed at 1.1a, as the hyperref hacks into the expansion of \contentsline were not executed. These now unneeded complications got removed at 1.1c.

1.1c also fixes a regression caused by 1.1a in this branch: the \etocthelinkedname had lost its hyperlink.

The \nonumberline extra token guarantees no brace stripping here.

```
\def\Etoc@getnb@nonumberline #1\hspace\etoc@ {%
    \Etoc@global\Etoc@numberedfalse
    \Etoc@global\let\etocthenumber \@empty
    \Etoc@global\expandafter\def\expandafter\etocthename\expandafter{\@gobble#1}%
}
```

This branch handles the peculiar "Part" syntax. No brace stripping possible here when grabbing #1, due to previous checks that it does not start by a space token or an opening brace.

1.1c handles this as a sub-branch from \Etoc@@getnb which brings simplifications. Also the code has been somewhat strengthened so as to avoid in later processing a brace removal issue on the name (which was a bug of legacy earlier code), when it turns our we are handling a numbered Part indeed.

The whole thing is anyhow quite fragile due to LaTeX's handling by standard classes of .toc file entries for parts being even more already pre-formatted for typesetting than for other levels.

```
\def\Etoc@@getit #1\hspace#2{%
    \ifx\etoc@#2%
      \Etoc@global\Etoc@numberedfalse
      \Etoc@global\let\etocthenumber \@empty
    \else
      \Etoc@global\Etoc@numberedtrue
      \Etoc@global\def\etocthenumber {#1}%
      \expandafter\Etoc@getit@getname \expandafter\@empty
    \fi
}
```

Chain of \expandafter's to get rid of the added \@empty token to avoid a brace removal. And this is the end of the 1.1a refactoring, completed at 1.1c.

```
\def\Etoc@getit@getname #1\hspace\etoc@ {%
    \Etoc@global\expandafter\def\expandafter\etocthename\expandafter{#1}%
}
```

place-holder

```
\let\etocthename    \@empty
\let\etocthenumber  \@empty
\let\etocthepage    \@empty
\let\etocthelinkedname    \@empty
\let\etocthelinkednumber  \@empty
\let\etocthelinkedpage    \@empty
\let\etocthelink    \@firstofone
\DeclareRobustCommand*{\etocname}  {}
\DeclareRobustCommand*{\etocnumber}{}
\DeclareRobustCommand*{\etocpage}  {}
\DeclareRobustCommand*{\etoclink}  {\@firstofone}
\DeclareRobustCommand*{\etocifnumbered}
   {\ifEtoc@numbered\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
\expandafter\let\expandafter\etocxifnumbered\csname etocifnumbered \endcsname
\DeclareRobustCommand*{\etociffirst}
   {\ifEtoc@isfirst\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
\expandafter\let\expandafter\etocxiffirst\csname etociffirst \endcsname
```

```
\def\Etoc@readtoc {%
  \ifeof \Etoc@tf
  \else
     \read \Etoc@tf to \Etoc@buffer
     \Etoc@toctoks=\expandafter\expandafter\expandafter
       {\expandafter\the\expandafter\Etoc@toctoks\Etoc@buffer}%
     \expandafter\Etoc@readtoc
  \fi
}
```

1.07m moves the reading of the toc file At Begin Document. Needed for Babel activated characters.

```
\Etoc@toctoks {}% (superfluous, but for clarity)
\AtBeginDocument{\IfFileExists{\jobname.toc}
    {{\endlinechar=\m@ne
      \makeatletter
      \newread\Etoc@tf
      \openin\Etoc@tf\@filef@und
      \Etoc@readtoc
      \global\Etoc@toctoks=\expandafter{\the\Etoc@toctoks}%
      \closein\Etoc@tf}}
    {\typeout{No file \jobname.toc.}}}
```

1.2c removes a legacy test it had copied over from hyperref internals, whose purpose was to check if the .toc file had been produced in an earlier no-hyperref pass, which in the past would trigger failure. For specifics see https://github.com/latex3/hyperref/issues/305.

```
\def\Etoc@openouttoc{%
  \if@filesw
   \newwrite \tf@toc
   \immediate \openout \tf@toc \jobname .toc\relax
  \fi
  \global\let\Etoc@openouttoc\empty
}
```

The \gdef\Etoc@stackofends{{-3}{}} is in theory superfluous as normally this stack should always have been restored on exit to its initial state, but...

It would perhaps be better to issue here \Etoc@notactivetrue, prior to \the\Etoc@toctoks but it is up to the caller macros to do it. \Etoc@minf is a level which is lower (i.e. more encompassing than all others), so numerically -2 for standard classes and -3 for memoir class.

The \Etoc@doendsandbegin will not try to execute a (non-existing) "begin" macro for the level -3.

```
\def\Etoc@toctoc{%
  \gdef\Etoc@stackofends{{-3}{}}%
  \global\let\Etoc@level\Etoc@minf
  \global\let\Etoc@virtualtop\Etoc@minf
  \the\Etoc@toctoks
  \ifEtoc@notactive
  \else
   \gdef\Etoc@level{-\thr@@}%
   \Etoc@doendsandbegin
  \fi
}
```

Memo: \etoclocaltop has only meaningful meaning when the toc is local and is "active". Except that I used a "notactive" flag to torture myself, so: has the \ifEtoc@notactive flag set to false.

The 1.2 removal of usage of boolean flags associated to the levels has made the next definition more succinct. Their former rôle is picked up by \Etoc@level.

MEMO: the \etoclocaltableofcontentshook has been added during development 1.2 probably for reasons of symmetry with handling of the local "lists of" code, but at time of release I can't remember clearly. Hesitation where to put it exactly.

1.2 adds a \Etoc@@startlocaltochook for a refactoring of \localtableofcontentswithrelative-depth. It also serves to make the emptiness check with local "lists of", as will be seen further down.

```
\def\Etoc@@startlocaltoc#1#2{%
    \ifEtoc@localtoc
       \ifnum #1=#2\relax
          \global\let\etoclocaltop\Etoc@virtualtop
          \Etoc@@startlocaltochook
          \etoclocaltableofcontentshook
```

The \ifEtoc@etocstyle boolean is true when etoc is left in its default mode (no usage of \etocsettocstyle or \etocclasstocstyle). It inhibits \Etoc@tableofcontents from using the specified {⟨*before_toc*⟩} and {⟨*after_toc*⟩} from usage of \etocsettocstyle. And here we can insert the code we wish to do the title.

**MEMO**: for legacy reason etoc shares a lot of code between global TOC and local TOCs. But it would probably have been better to separate the two and provide an \etocsetlocaltocstyle that the user can use in preamble. As it stands \etocsettocstyle if used in preamble also influences the global TOC, so basically one has to use it again after it (if it comes first in the document).

The \ifEtoc@ouroboros mechanism when it is set to false appears rather audacious.

```
          \ifEtoc@etocstyle
              \etocetoclocaltocmaketitle
          \fi
          \ifx\Etoc@aftertitlehook\@empty
          \else
            \ifEtoc@localtoctotoc
              \ifEtoc@ouroboros
              \else
                \let\Etoc@tmp\contentsline
                \def\contentsline{\let\contentsline\Etoc@tmp\Etoc@gobblethreeorfour}%
              \fi
            \fi
          \fi
          \global\Etoc@notactivefalse
       \fi
    \fi
}
\let\etoc@startlocaltoc\@gobble
\let\Etoc@@startlocaltoc@toc\Etoc@@startlocaltoc
\let\Etoc@@startlocaltochook\@empty
\unless\ifEtoc@deeplevels
```

This only hard-codes the built-in LaTeX defaults.

```
  \def\etocdivisionnameatlevel#1{%
    \ifcase\numexpr#1\relax
              \ifdefined\c@chapter chapter\else section\fi%
         \or section%
         \or subsection%
         \or subsubsection%
         \or paragraph%
         \or subparagraph%
         \or empty%
    \else\ifnum\numexpr#1<\m@ne
              book%
         \else
              part%
         \fi
    \fi
  }
\else
```

The definition given to \etocdivisionnameatlevel in the **deeplevels** branch is only needed if one wants to locate a local table of contents at a deep place, and one has not made use of \etocsettocstyle to configure the heading by oneself. I have used in the default definition the headings as in the Doxygen LaTeX templates originating in the Doxygen#9936 PR.

```
  \def\etocdivisionnameatlevel#1{%
    \ifcase\numexpr#1\relax
              \ifdefined\c@chapter chapter\else section\fi%
          \or section%
          \or subsection%
          \or subsubsection%
          \or subsubsubsection%
          \or subsubsubsubsection%
          \or subsubsubsubsubsection%
          \or paragraph%
          \or subparagraph%
    \else\ifnum\numexpr#1>\z@
              empty%
          \else\ifnum\numexpr#1=\m@ne
              part%
          \else
              book%
          \fi\fi
    \fi
  }
\fi
\def\etoclocalheadtotoc#1#2{\addcontentsline{toc}{@#1}{#2}}
\def\etocglobalheadtotoc{\addcontentsline{toc}}
```

I have coded this so that it can be copied pasted to a user document and customized at will. 1.2a uses a better test for knowing if the starred command of previous line created an inline heading with \everypar: 1.2 simply tested if \etoclocaltop was at least 3 but this does not scale well with **deeplevels**. Only worry here is whether the \if@noskipsec thing really works with all document classes. At least it is in LaTeX kernel.

```
\providecommand*\UseName{\@nameuse}
\def\etocetoclocaltocmaketitle{%
    \UseName{\etocdivisionnameatlevel{\etoclocaltop+1}}*{\localcontentsname}%
    \if@noskipsec\leavevmode\par\fi
    \etociflocaltoctotoc
      {\etocifisstarred
         {}% star variant, do not add to toc
         {\etoclocalheadtotoc
           {\etocdivisionnameatlevel{\etoclocaltop+1}}%
           {\localcontentsname}%
         }%
      }%
      {}%
}%
```

In dev, I used \etoc prefix for the next two, but I decided to drop it for release.

```
\def\localcontentsname  {\contentsname}%
\let\etoclocaltableofcontentshook\@empty
```

The big thing at 1.2: experimental support code for \locallistoffigures and \locallistoftables! There were two possibilities to implement the 'localtoc' mechanism: either add some extra things to the .lof and .lot files, or get their data duplicated in the .toc file. I have chosen for time being the latter path, hence this goes via hacking into \addcontentsline and it must be the case that the document class uses it (from \caption). If some package has modified the implementation of captions of figures and tables and insertes data in the .lof or .lot files directly via \addtocontents for example and not via \addcontentsline, then the **etoc** mechanism will fail.

```
\if1\ifEtoc@lof0\fi\ifEtoc@lot0\fi1%
\else
\AtBeginDocument{%
  \let\Etoc@originaladdcontentsline\addcontentsline
  \def\addcontentsline{\Etoc@hackedaddcontentsline}%
}%
\fi
```

For optimization of execution speed we define the macro conditionnally on the option status (wihch is frozen). In the first case, `#1=lof,.lot` would work but this is not realistic.

```
\ifEtoc@lof
  \ifEtoc@lot
    \def\Etoc@hackedaddcontentsline#1{%
      \expanded{\noexpand\in@{.#1,}}{.lof,.lot,}%
      \ifin@\expandafter\Etoc@hackedaddcontentsline@i
      \else\expandafter\Etoc@originaladdcontentsline
      \fi {#1}}
  \else
    \def\Etoc@hackedaddcontentsline#1{%
      \expanded{\noexpand\in@{.#1,}}{.lof,}%
      \ifin@\expandafter\Etoc@hackedaddcontentsline@i
      \else\expandafter\Etoc@originaladdcontentsline
      \fi {#1}}
  \fi
\else
  \def\Etoc@hackedaddcontentsline#1{%
    \expanded{\noexpand\in@{.#1,}}{.lot,}%
    \ifin@\expandafter\Etoc@hackedaddcontentsline@i
    \else\expandafter\Etoc@originaladdcontentsline
    \fi {#1}}
\fi
```

This business of `\protected@file@percent` is not really needed as anyhow **etoc** reads the `.toc` file with no space token from end of lines.

```
\def\Etoc@hackedaddcontentsline@i#1#2#3{%
    \expanded{\noexpand\in@{.#1;#2,}}{.lof;figure,.lot;table,}%
    \ifin@
    \addtocontents {toc}{%
      \protect\contentsline{#2}{#3}{\thepage}{\ifEtoc@hyperref\@currentHref\fi}%
      \ifdefined\protected@file@percent\protected@file@percent\fi
    }%
    \fi
    \Etoc@originaladdcontentsline{#1}{#2}{#3}%
}
```

The two macros need to be redefined if `\expanded` is not provided by the engine. Here we leave some tests done at execution time although they could have been done (as above) here at definition time.

```
\unless\ifdefined\expanded
  \def\Etoc@hackedaddcontentsline#1{%
    {\edef\Etoc@tmp{\noexpand\in@{.#1,}{\ifEtoc@lof.lof,\fi\ifEtoc@lot.lot,\fi}}\expandafter}%
    \Etoc@tmp
    \ifin@\expandafter\Etoc@hackedaddcontentsline@i
    \else\expandafter\Etoc@originaladdcontentsline
    \fi {#1}%
  }
  \def\Etoc@hackedaddcontentsline@i#1#2#3{%
    {\edef\Etoc@tmp{\noexpand\in@{.#1;#2,}}\expandafter}%
    \Etoc@tmp{.lof;figure,.lot;table,}%
    \ifin@
```

```
    \addtocontents {toc}{%
      \protect\contentsline{#2}{#3}{\thepage}{\ifEtoc@hyperref\@currentHref\fi}%
      \ifdefined\protected@file@percent\protected@file@percent\fi
    }%
    \fi
    \Etoc@originaladdcontentsline{#1}{#2}{#3}%
  }
\fi
```

We will simply let \locallistoffigures and \locallistoftables use \localtableofcontents. We need some dedicated variant of \Etoc@@startlocaltoc. Hesitation where to put \Etoc@@startlocaltochook which has multiple usages.

```
\def\Etoc@@startlocallistof#1#2#3{%
    \ifEtoc@localtoc
       \ifnum #2=#3\relax
          \global\let\etoclocaltop\Etoc@virtualtop
          \global\Etoc@notactivefalse
          \Etoc@@startlocaltochook
          \csname etoclocallistof#1shook\endcsname
          \ifEtoc@etocstyle
              \csname etocetoclistof#1smaketitle\endcsname
          \fi
       \fi
    \fi
}
\def\Etoc@@startlocallistof@setlevels#1{%
```

#1 is figure or table.

\etoclocaltop represents the level which will stop the local "list of" contents. I hesitated whether local "lists of" should obey the tocdepth, especially a varying one. Finally at last minute I opted for "Yes", so in the end no alteration of tocdepth will be done here. We will set the level to have value

```
min(1,\etoclocaltop+1)
```

At least 1, because it really would not make sense to show figure entries as chapter entries for a local list of figures in a Part (if the user has for example set the \etoclocallistoffigureshook to do nothing, so that the line styles as configured via \etocsetstyle are not avoided). And it must be greater than \etoclocaltop else such an entry would terminate the scope of the local contents.

In the default context which has issued via \etoclocallistoffigureshook a local \etocstandardlines, the actual value of the level only has a "all or nothing" meaning: if it is greater than tocdepth then the \Etoc@etoccontentsline will not call \contentsline, else it will and then the code from document class will kick in. The article class defines \l@figure as

```
\@dottedtocline {1}{1.5em}{2.3em}
```

And we are here in a context where necessarily tocdepth is at least 1 (if it was 0 or less, as the level has been set to at least 1, the entry would have been filtered out earlier by \Etoc@etoccontentsline), so the line will show. And if \etoclocallistoffigureshook was modified and did not issue \etocstandardlines and we are in this situation that the tocdepth comparison did not inhibit the entry, then it will show too, at least if the line style does not say to do nothing (notice that the etoc fall-back line styles are configured to do nothing for paragraph and subparagraph entries).

On the other hand if the tocdepth inhibits entries of level \etoclocaltop+1, then a \localtableofcontents at this level would display no contents, so why should it not also mute \locallistoffigures?

This change to the level \Etoc@figure@@ or \Etoc@table@@ is done only locally, no need to worry about collaterals after the "List of".

```
          \ifnum\etoclocaltop<\z@
              \expandafter\let\csname Etoc@#1@@\endcsname\@ne
          \else
              \expandafter\let\csname Etoc@#1@@\expandafter\endcsname
                              \csname Etoc@\the\numexpr\etoclocaltop+\@ne @@\endcsname
          \fi
```

We now make invisible all level names whose numerical level would have allowed them to show in these

contents. figure and table are never included in the \Etoc@dolevels no danger to cancel them out. The special level names @section etc... which are inserted by headings of local "lists of" or local TOCs under **localloftotoc** and other options have been made invisible by \Etoc@listofhook, no need to do anything about them here.

```
            \def\Etoc@do##1{%
                \ifnum\etoclevel{##1}>\etoclocaltop
                       \expandafter\let\csname Etoc@##1@@\endcsname\Etoc@maxlevel
                \fi}%
            \Etoc@dolevels
}
```

Let's for time being configure the figure and table lines to be rendered as in class default.

Ah, I remember why I added \etoclocaltableofcontentshook above. It is by symmetry as I had defined those next two already.

```
\def\etoclocallistoffigureshook{\etocstandardlines}
\def\etoclocallistoftableshook {\etocstandardlines}
```

Here is some info about usage of \etoclocallistoffigureshook, for those who end up here from having clicked on the name from the user manual. Its default as above means to use the class default for lines in \listoffigures. If you redefine it to be empty, the effect is that (except of course if \etocstandardlines has been issued globally) the figure lines will adopt the style configured for level \etoclocaltop+1 (or more precisely the minimum of that and of 1). You can definitely put into the hook some

```
\etocsetstyle{\number\etoclevel{figure}}{}{}{...}{}
```

where the dots represent some code with \etocnumber, \etocname, \etocpage, itself possibly querying \etoclevel{figure}, or perhaps \etoclocaltop to know the actual depth (which may the one of a Part which can thus be distinguished from being in a Chapter, whereas \number\etoclevel{figure} will give 1 in both cases).

There is variant which is to maintain the default \etocstandardlines in the hook and then re-define the kernel macro \l@figure to do the desired thing. For example the default with article is for \l@figure to do \@dottedtocline {1}{1.5em}{2.3em} so you can do some variant where the second and third argument (indent and numwidth) are set according to \etoclocaltop.

I hope the above explanations help, they appear too advanced for inclusion in the user manual so I give them here.

In dev, I used \etoc prefix here, but I decided to drop it for release.

```
\def\locallistfigurename{\listfigurename}
\def\locallisttablename {\listtablename}
```

Same observations as for \etocetoclocaltocmaketitle.

```
\def\etocetoclistoffiguresmaketitle{%
    \UseName{\etocdivisionnameatlevel{\etoclocaltop+1}}*{\locallistfigurename}%
    \ifnum\etoclocaltop>\tw@\mbox{}\par\fi
    \etociflocalloftotoc
      {\etocifisstarred
         {}% star variant, do not add to toc
         {\etoclocalheadtotoc
           {\etocdivisionnameatlevel{\etoclocaltop+1}}%
           {\locallistfigurename}%
         }%
      }%
      {}%
}%
\def\etocetoclistoftablesmaketitle{%
    \UseName{\etocdivisionnameatlevel{\etoclocaltop+1}}*{\locallisttablename}%
    \ifnum\etoclocaltop>\tw@\mbox{}\par\fi
    \etociflocallottotoc
      {\etocifisstarred
         {}% star variant, do not add to toc
         {\etoclocalheadtotoc
           {\etocdivisionnameatlevel{\etoclocaltop+1}}%
```

```
            {\localisttablename}%
          }%
      }%
      {}%
}%
```

The local lists of do support the `\label`/`\ref` syntax as we are careful here to position `\localtableofcontents` as last token.

The reset to `\@empty` of the `\Etoc@listofreset` is not strictly needed as the other things can always be done with no harm.

```
\let\Etoc@listofreset\@empty
```

Memo: `\ext@toc` defined in KOMA and memoir but not in the standard classes.

```
\ifEtoc@lof
  \def\locallistoffigures{%
      \def\Etoc@listofreset{%
          \let\Etoc@currext\Etoc@tocext
          \let\Etoc@@startlocaltoc\Etoc@@startlocaltoc@toc
          \let\Etoc@@startlocaltochook\@empty
          \let\Etoc@listofreset\@empty
```

The `\Etoc@listofhook` is executed by `\localtableofcontents`. It will be used here to let local "lists of" ignore all the special `.toc` entries whose level names start with a `@`. This is to avoid the scope limiting detection of the local list of figures or tables from being influenced by another list of, such as a list of tables following a list of figures which has put its title inside the `.toc` due to `locallottotoc` option of **etoc**.

```
          \let\Etoc@listofhook\@empty
          }%
      \let\Etoc@currext\Etoc@lofext
      \def\Etoc@@startlocaltoc{\Etoc@@startlocallistof{figure}}%
      \def\Etoc@@startlocaltochook{\Etoc@@startlocallistof@setlevels{figure}}%
      \def\Etoc@listofhook{%
        \def\Etoc@do####1{%
          \expandafter\let\csname Etoc@@####1@@\endcsname\Etoc@maxlevel
        }%
        \Etoc@dolevels
      }%
      \localtableofcontents
  }
\else
  \def\locallistoffigures{%
      \PackageError{etoc}{%
        \string\locallistoffigures \on@line\space but\MessageBreak
        package was loaded without `lof' option}%
        {Try again with \string\usepackage[lof]{etoc}}%
      }
\fi
\ifEtoc@lot
  \def\locallistoftables{%
      \def\Etoc@listofreset{%
          \let\Etoc@currext\Etoc@tocext
          \let\Etoc@@startlocaltoc\Etoc@@startlocaltoc@toc
          \let\Etoc@@startlocaltochook\@empty
          \let\Etoc@listofreset\@empty
          \let\Etoc@listofhook\@empty
          }%
      \let\Etoc@currext\Etoc@lotext
      \def\Etoc@@startlocaltoc{\Etoc@@startlocallistof{table}}%
      \def\Etoc@@startlocaltochook{\Etoc@@startlocallistof@setlevels{table}}%
      \def\Etoc@listofhook{%
```

```
        \def\Etoc@do####1{%
          \expandafter\let\csname Etoc@@####1@@\endcsname\Etoc@maxlevel
        }%
        \Etoc@dolevels
      }%
      \localtableofcontents
  }
\else
  \def\locallistoftables{%
      \PackageError{etoc}{%
        \string\locallistoftable \on@line\space but\MessageBreak
        package was loaded without `lot' option}%
        {Try again with \string\usepackage[lot]{etoc}}%
      }
\fi
```

\Etoc@tocid is the number of the toc (possibly gotten via a \ref following a \tableofcontents), or it is \z@ if the emptiness test is from a global toc. Until the compilations stabilize, some local TOCs can get printed at wrong locations naturally and emptiness tests can not be trusted either.

Note: (1.08i 2016/09/29) the code has to handle both local and total toc. Hence the flag \ifEtoc@notactive has to be set prior to it. For a global toc, the \Etoc@tocid was set to \z@, and the \ifnum in \etoc@startlocaltoc did always fail, but I now prefer to simply nullify the \etoc@startlocaltoc. As its default fallback is \@gobble I simply test here for the \ifEtoc@localtoc flag. The \Etoc@tocid will be undefined for a global toc but it is not tested anymore.

The initialization such as \global\let\Etoc@level\Etoc@minf is needed in case the .toc file contains an \etoc@startlocaltoc before any \contentsline.

Local list of figures and tables set especially the tocdepth and a hook is added here for emptiness check to work correctly with them.

MEMO: Should I also execute \etoclocaltableofcontentshook? Then I would need to set \etoclocallistoffigureshook to redefine it, rather than be inserted as itself.

```
\def\Etoc@checkifempty {%
    \global\Etoc@isemptytoctrue
    \global\Etoc@stoptocfalse
    \global\let\Etoc@level\Etoc@minf
    \global\let\Etoc@virtualtop\Etoc@minf
    \gdef\Etoc@stackofends{{-3}{}}%
    \begingroup
      \ifEtoc@localtoc
        \def\etoc@startlocaltoc##1{%
          \ifnum##1=\Etoc@tocid\relax
              \global\let\etoclocaltop\Etoc@virtualtop
              \Etoc@@startlocaltochook
              \global\Etoc@notactivefalse
          \fi
        }%
```

The mechanism is to check if any \contentsline triggers execution. For this we replace the **etoc** replacement by another replacement.

```
        \let\contentsline\Etoc@testingcontentslinelocal
      \else
        \let\contentsline\Etoc@testingcontentsline
      \fi
      \Etoc@storetocdepth
```

This here is a 1.09i added work-around to avoid usage of \etocsetlocaltop.toc which could cause the test of emptiness of a global TOC to actually execute some start and finish parts in some cases.

```
        \let\Etoc@setlocaltop@doendsandbegin\@empty
        \the\Etoc@toctoks
      \Etoc@restoretocdepth
```

```
    \endgroup
}
\DeclareRobustCommand*\etocifwasempty
  {\ifEtoc@isemptytoc\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi }
\expandafter\let\expandafter\etocxifwasempty\csname etocifwasempty \endcsname
\def\Etoc@testingcontentslinelocal #1{%
  \ifEtoc@stoptoc
  \else
   \ifnum\csname Etoc@#1@@\endcsname=\Etoc@maxlevel
   \else
    \global\expandafter\let\expandafter\Etoc@level\csname Etoc@#1@@\endcsname
    \if @\@car#1\@nil\else\global\let\Etoc@virtualtop\Etoc@level\fi
    \ifEtoc@notactive
    \else
```

\ifEtoc@notactive is \iffalse in case of a local TOC once it has encountered the \etoc@startlocaltoc with matching id. So here \etoclocaltop has been set by \Etoc@@startlocaltoc and we compare to the new level encountered. If the latter is at most equal to \etoclocaltop this means the local TOC ends there and is empty, so we set the \ifEtoc@stoptoc to true, which will cause the subsequent parsing to abort immediately. If it is greater we check how it compares to the local required toc depth. If lower or equal we conclude the toc is not empty, and toggle the flag to stop the parsing; if greater, we are still in doubt and must continue.

```
     \ifnum\Etoc@level>\etoclocaltop
      \unless\ifnum\Etoc@level>\c@tocdepth
       \global\Etoc@isemptytocfalse
       \global\Etoc@stoptoctrue
      \fi
     \else
      \global\Etoc@stoptoctrue
     \fi
    \fi
   \fi
  \fi
```

The \@gobblethree was added to the LaTeX kernel for its 2020-02-02 PL 5 release and we require 2020-10-01 since 1.1a.

```
  \Etoc@gobblethreeorfour{}%
}
```

Test of emptiness of the global TOC (according to current setting of the levels and a possibly evolving tocdepth).

```
\def\Etoc@testingcontentsline #1{%
  \ifEtoc@stoptoc
  \else
   \ifnum\csname Etoc@#1@@\endcsname=\Etoc@maxlevel
   \else
    \unless\ifnum\csname Etoc@#1@@\endcsname>\c@tocdepth
     \global\Etoc@isemptytocfalse
     \global\Etoc@stoptoctrue
    \fi
   \fi
  \fi
  \Etoc@gobblethreeorfour{}%
}
```

1.08e lets \localtableofcontents do a first scan of the .toc file (as stored in \Etoc@toctoks) to determine if the table of contents will in fact end up empty. Only, however if user has added \etoccheckscontentsempty to document.

If this optional emptiness check is positive, nothing is typeset. The command \etocaftertochook is still executed though. Other ways were envisioned (like delimited macros) to determine this potential

emptiness, but in the end I opted for execution of the .toc file with suitable definitions for \contentsline and \etoc@startlocaltoc. Notice though that if emptiness would result from empty line styles, this can not be detected. Emptiness means "no \contentsline under the TOC scope".

For this detection of emptiness, assignments (here and in \Etoc@testingcontentsline) are made globally, I think this is the best (just in case some portions of the .toc file turn out to be inside some groups — perhaps for some silly color assignments, etc... — whose boundaries do not necessarily respect unit levels).

The flag \ifEtoc@tocwithid discriminates between a \localtableofcontents and a \tableofcontents \ref{foo}; the latter could so far possibly refer to a local or also to a global table of contents but release 1.08e has deprecated the latter use as it complicated the code, for something truly silly. Thus \ref{foo} must now be with foo a label of a *local* TOC. As a result \ifEtoc@tocwithid is less used now.

In the case of a \refed-to toc whose label was just added hence is not yet in the .aux file, \Etoc@tocid is 0. etoc used to issue a warning to run latex again and did no printing at all. Release 1.08e in such cases prints the heading (this may gain one compilation step). Emptiness test is not executed as it would necessarily turn out positive and can not be trusted anyhow. The TOC is declared non empty, which it probably is...

Emptiness detection for local tables of contents (either from a \localtableofcontents or from a \tableofcontents\ref{localtoc}) can be trusted only when the .toc file has stabilized.The emptiness status of a local TOC whose Id is not yet in the .toc is by necessity undecided yet (and not to be trusted really as the numbering may have changed; only when compilation runs settle is the emptiness status to be trusted). The code declares the TOC non empty, as it will be in 95% of use cases.

Dropping support for \tableofcontents \ref{globaltoc} means here that when a TOC id is not found in the .toc file we can assume it definitely has to be a local TOC needing more compilations. The emptiness status is undecided, the code declares the TOC non empty.

1.08i 2016/09/29 now does \Etoc@localtoctrue right at the start (the earlier code could have to handle table of contents which were actually global, via the \label/\ref mechanism.) It does not rely on the \ifnum automatically false in \Etoc@@startlocaltoc due to the special values 0 or \z@ for \Etoc@tocid, but simply leaves \etoc@startlocaltoc to its default \@gobble. The \Etoc@isemptytocfalse is upfront in case some code using \etocifwasempty is in user hooks. The default is to assume the TOC non-empty as its contents are actually still unknown. Under the \Etoc@stoptoctrue flag, the \Etoc@etoccontentsline is more efficient now.

The \ifEtoc@notactive flag needs to be set before calling \Etoc@checkifempty.

I hesitated with 1.08i to write something to aux file in order to let LaTeX prompt the user for extra pass, after insertion of some new \localtableofcontents, but finally I prefer to only trick LaTeX into telling about undefined references.

The \PackageWarning approach has the advantage that at least in Emacs/AUCTeX the C-cC-c will propose LaTeX, not View. But perhaps some automated scripts checking aux file will not like the extra line which is then removed in next pass, and could possibly do one extra unneeded compilation to check aux file remains identical. Hence the second approach. (edit 2017/10/23: good thing I documented that! I had completely forgotten that rationale, but I wonder if it is correct.)

Also the \PackageWarning does not trigger a visible message near the end of the log file or console output, contrarily to a

```
LaTeX Warning:  There were undefined references.
```

followed by a

```
LaTeX Warning:  Label(s) may have changed.  Rerun to get cross-references right.
```

Method used here seems to work fine also with latexmk: it does not seem to induce it into making too many runs.

```
\def\Etoc@localtableofcontents#1{%
    \gdef\etoclocaltop{-\@m}%
    \Etoc@localtoctrue
    \global\Etoc@isemptytocfalse
    \edef\Etoc@tocid{#1}%
    \ifnum\Etoc@tocid<\@ne
      \setbox0\hbox{\ref{Unknown toc ref \@secondoftwo#1. \space Rerun LaTeX}}%
```

Do only heading, skip all the rest.

```
      \global\Etoc@stoptoctrue
      \gdef\etoclocaltop{-\thr@@}%
```

```
      \Etoc@tableofcontents
      \expandafter\Etoc@gobtoetoc@
    \fi
    \global\Etoc@notactivetrue
    \ifEtoc@checksemptiness
       \Etoc@checkifempty
    \fi
    \ifEtoc@isemptytoc
       \ifEtoc@notactive
         \setbox0\hbox{\ref{Unknown toc ID \number\Etoc@tocid. \space Rerun LaTeX}}%
```

Assume real one will be non-empty and print only heading for this pass.

```
         \global\Etoc@isemptytocfalse
         \global\Etoc@stoptoctrue
         \gdef\etoclocaltop{-\thr@@}%
         \Etoc@tableofcontents
         \expandafter\expandafter\expandafter\Etoc@gobtoetoc@
       \fi
    \else
       \global\Etoc@stoptocfalse
       \global\Etoc@notactivetrue
```

We can end up here either if the emptiness check was done and turned negative (then `\etoclocaltop` has the correct level for usage in first argument of `\etocsettocstyle`), or if the emptiness check was not done. For the latter case `\etoclocaltop` has setting `-\@m`.

```
       \edef\etoc@startlocaltoc##1%
           {\noexpand\Etoc@@startlocaltoc{##1}{\Etoc@tocid}}%
       \Etoc@tableofcontents
    \fi
    \@gobble\etoc@
    \endgroup\ifEtoc@mustclosegroup\endgroup\fi
    \Etoc@tocdepthreset
    \Etoc@listofreset
    \etocaftertochook
}% \Etoc@localtableofcontents
```

2013/03/07: I discover a `\@namedef` trick to construct the `\Etoc@again` space delimited macro:

```
 \@namedef {Etoc@again} {...stuff...}
```

Original version was (copied from analogous stuff in source2e):

```
 {\def\1{\Etoc@again}\expandafter\gdef\1 {...stuff...}}
```

and in the end (now that I think about it) I simply use `\@firstofone`.

1.2 has a removed a `\Etoc@getrefno` as the author now knows about `\@car` so need to define a similar utility here!

Much more importantly (and embarrassingly) 1.2 fixes a bug which had been here for ever. The code in its innocence 2012 birth year assumed that the first entry of `\r@foo`, if the latter is defined, is always numerical! But this is broken by varioref if one used for example `\vpageref{foo}` before! The first brace pair will then be `{??}`... causing in **etoc** a "Missing number" error.

About the current fixed code, there is no strong reason that `\Etoc@getref` should work expandably. Later `\Etoc@localtableofcontents` does

```
\edef\Etoc@tocid{#1}\iffnum\Etoc@tocid<\@ne
```

but I certainly could organize things differently.

Maybe I should investigate more on what nasty things car be in first argument. Or submit it to an `\expanded`?

```
\def\Etoc@getref #1{%
    \@ifundefined{r@#1}
         {0}
         {\expandafter\Etoc@getref@i\romannumeral-`0%
          \expandafter\expandafter\expandafter
          \@car\csname r@#1\endcsname0\@nil\@etoc
```

```
          }%
}
\def\Etoc@getref@i#1#2\@etoc{\ifnum9<1\string#1 #1#2\else 0\fi}
\def\Etoc@ref#1{\Etoc@localtableofcontents{\Etoc@getref{#1}}}
\def\Etoc@label#1{\label{#1}\futurelet\Etoc@nexttoken\Etoc@t@bleofcontents}
\@firstofone{\def\Etoc@again} {\futurelet\Etoc@nexttoken\Etoc@t@bleofcontents}
```

`\ref{foo}` expects foo to be a label to a *local* TOC.
The syntax `\localtableofcontents\ref{foo}` is supported.

```
\def\Etoc@dothis #1#2\etoc@ {\fi #1}
\def\Etoc@t@bleofcontents{%
    \gdef\etoclocaltop{-\@M}%
    \ifx\Etoc@nexttoken\label\Etoc@dothis{\expandafter\Etoc@label\@gobble}\fi
    \ifx\Etoc@nexttoken\@sptoken\Etoc@dothis{\Etoc@again}\fi
```

`\Etoc@ref` will hand over directly to `\Etoc@localtableofcontents`. Argument will be (or rather expand to) zero if the reference is non-existent yet.

```
    \ifx\Etoc@nexttoken\ref\Etoc@dothis{\expandafter\Etoc@ref\@gobble}\fi
```

Flag to check if we were called from a `\localtableofcontents`.

```
    \ifEtoc@tocwithid\Etoc@dothis{\Etoc@localtableofcontents{\c@etoc@tocid}}\fi
```

From now on we are handling a global TOC. Earlier, I used the trick of setting `\Etoc@tocid` to `\z@` for compatibility with expansion of `\etoc@startlocaltoc`. But since 1.08i `\etoc@startlocaltoc` is left to be `\@gobble`, and `\Etoc@tocid` is never tested. We don't need to set the `\ifEtoc@notactive` flag as now `\Etoc@testingcontentsline` tests first the `\ifEtoc@localtoc` flag (was already the case of `\Etoc@etoccontentsline`). I change a bit the style of conditionals here for clarity of code.

```
    \global\Etoc@isemptytocfalse
    \ifEtoc@checksemptiness\Etoc@checkifempty\fi
    \ifEtoc@isemptytoc
     \ifEtoc@notocifnotoc
      \expandafter\expandafter\expandafter\@gobble
     \fi
    \fi
    \Etoc@tableofcontents
    \endgroup
    \ifEtoc@mustclosegroup\endgroup\fi
    \Etoc@tocdepthreset
    \Etoc@listofreset
    \etocaftertochook
    \@gobble\etoc@
    }% \Etoc@t@bleofcontents
```

1.08c does not use `\arabic` in the `\addtocontents` since I have seen that in some circumstances (for some right to left languages with polyglossia or babel), one can not rely on `\arabic` having its default definition. As the number written here will be used later in an `\ifnum`, I should not have used it in the first place (done 2015/03/30).

```
\def\Etoc@table@fcontents{%
    \refstepcounter{etoc@tocid}%
    \Etoc@tocwithidfalse
    \futurelet\Etoc@nexttoken\Etoc@t@bleofcontents
}
\def\Etoc@localtable@fcontents{%
    \refstepcounter{etoc@tocid}%
    \addtocontents{toc}{\string\etoc@startlocaltoc{\the\c@etoc@tocid}}%
    \Etoc@tocwithidtrue
    \futurelet\Etoc@nexttoken\Etoc@t@bleofcontents
}
```

Attention that there could be a `\ref` following, thus we don't yet know whether this is a local or global table of contents.

The \Etoc@tocdepthset is for \etocsetnexttocdepth mechanism.

```
\def\etoctableofcontents{%
   \Etoc@openouttoc
   \Etoc@tocdepthset
   \begingroup
```

This group will be closed in \Etoc@t@bleofcontents or \Etoc@localtableofcontents.

Prior to its release 1.4c, tableof added a group pair via \tof@begin and \tof@finish. It was removed at 1.4c, so no need to do anything now here about silencing \tof@begingroup and \tof@endgroup: they are inserted only in the tableof private copy of the .toc file which is used by its own table of contents typesetting command.

1.09b uses a \def in non-starred variant for allowing tricks to recognize later on if we are in a starred or non-starred case, whatever the user definition of \etocaftertitlehook may be.

```
      \@ifstar
      {\let\Etoc@aftertitlehook\@empty\Etoc@table@fcontents}
      {\def\Etoc@aftertitlehook{\etocaftertitlehook}\Etoc@table@fcontents}%
}% \etoctableofcontents
\def\etocifisstarred{\ifx\Etoc@aftertitlehook\@empty
                          \expandafter\@firstoftwo\else
                          \expandafter\@secondoftwo
                     \fi}
\let\etocoriginaltableofcontents\tableofcontents
\let\tableofcontents\etoctableofcontents
```

The \Etoc@listofhook is configured by \locallistoffigures and \locallistoftables. The latter two use hooks rather than \begingroup...\endgroup to manage local configuration, in part because they want to support the \label/\ref special etoc mechanism, so need to issue \localtableofcontents last for it to be able to pick up a following \label or \ref.

```
\let\Etoc@listofhook\@empty
\newcommand*\localtableofcontents{%
   \Etoc@openouttoc
   \Etoc@tocdepthset
```

This group closed in \Etoc@t@bleofcontents or \Etoc@localtableofcontents. Same comment relative to tableof. No need to do anything here.

```
   \begingroup
      \Etoc@listofhook
      \@ifstar
      {\let\Etoc@aftertitlehook\@empty\Etoc@localtable@fcontents}
      {\def\Etoc@aftertitlehook{\etocaftertitlehook}\Etoc@localtable@fcontents}%
}% \localtableofcontents
```

1.09 adds \localtableofcontentswithrelativedepth. Note that changes to the tocdepth from inside the .toc file during duration of local toc will remain without effect as a substitute is used which gets a frozen non-dynamical value.

It seems 1.09 actually forgot a change to \Etoc@etoccontentsline and the previous sentence was false and described only what happened during the emptiness check which thus could give false positives or false negatives. But if I had done the needed change in \Etoc@etoccontentsline to keep the two in sync, even if tocdepth evolves from the .toc file, I would have created a divergence between local TOCs (not only the \localtableofcontentswithrelativedepth ones) under \etocstandardlines and those not with standard lines.

Redoing the whole thing at 1.2. I needed to insert a suitable hook in \Etoc@@startlocaltoc (and also in the version used by \Etoc@checkifempty). And I have now at my disposal \Etoc@listofreset to reset it (it will be expanded after both the emptiness check and the typesetting are completed, and at same grouping level as the trigger command defined here). I do not want to add tokens after \localtableofcontents to not break the \label/\ref thing, which is one reason for this way of proceeding.

```
\newcommand*\localtableofcontentswithrelativedepth[1]{%
   \def\Etoc@@startlocaltochook{%
       \global\c@tocdepth\numexpr\etoclocaltop+#1\relax
```

```
    }%
    \def\Etoc@listofreset{\let\Etoc@@startlocaltochook\@empty
                          \let\Etoc@listofreset\@empty}%
    \localtableofcontents
}% \localtableofcontentswithrelativedepth
```

Prior to 1.2, `\Etoc@tableofcontents` was constructed by `\etocsettocstyle` as one big macro where #1 and #2 had been inserted. It now simply stores #1 and #2 each in a macro and `\Etoc@tableofcontents` is defined once and for all. Also the boolean `\ifEtoc@etocstyle` is added which helps keeping the emulation of the document class only for the global document TOC but use a better choice for local tables of contents (especially those in deeper sub-levels), which is activated via `\etocetoclocaltocstyle` (near end of code).

`\etocstoretocstyleinto` added at 1.2. Too lazy to check if #1 pre-exists. Well, rather, too annoying to do this check as one may want to redefine #1 without further ado.

```
\newcommand\etocsettocstyle[2]{%
    \Etoc@etocstylefalse
    \Etoc@classstylefalse
    \def\Etoc@tableofcontents@user@before{#1}%
    \def\Etoc@tableofcontents@user@after {#2}%
}%
\def\etocstoretocstyleinto#1{%
%%      \@ifdefinable#1{%
        \edef#1{\noexpand\Etoc@etocstylefalse\noexpand\Etoc@classstylefalse
                \def\noexpand\Etoc@tableofcontents@user@before{%
                    \unexpanded\expandafter{\Etoc@tableofcontents@user@before}%
                  }%
                \def\noexpand\Etoc@tableofcontents@user@after{%
                    \unexpanded\expandafter{\Etoc@tableofcontents@user@after}%
                  }%
               }%
%%      }%
}%
```

The macro names added here at 1.2 are provisory. Next release will probably do a complete renaming of various internals because this is currently a complete mess with some names differing only by an @ versus a voyel, but not necessarily being at a deeper level of expansion the more @'s they have... the deepest one being this `\Etoc@tableofcontents`...

```
\def\Etoc@tableofcontents {%
    \Etoc@tableofcontents@etoc@before
    \ifEtoc@localtoc\ifEtoc@etocstyle\expandafter\expandafter\expandafter\@gobble\fi\fi
    \Etoc@tableofcontents@user@before
    \Etoc@tableofcontents@contents
    \ifEtoc@localtoc\ifEtoc@etocstyle\expandafter\expandafter\expandafter\@gobble\fi\fi
    \Etoc@tableofcontents@user@after
    \Etoc@tableofcontents@etoc@after
    \@gobble\etoc@
}
\def\Etoc@tableofcontents@etoc@before{%
    \ifnum\c@tocdepth>\Etoc@minf
    \else
     \expandafter\Etoc@gobtoetoc@
    \fi
    \Etoc@par
    \Etoc@beforetitlehook
    \etocbeforetitlehook
    \Etoc@storetocdepth
    \let\Etoc@savedcontentsline\contentsline
    \let\contentsline\Etoc@etoccontentsline
    \ifEtoc@standardlines
```

```
    \else
```

1.1a's \Etoc@lxyz now fetches 3 not 2 arguments and the \l@section etc... are not \let to it anymore, as they used to be here formerly.

For backwards compatibility the \etocsavedchaptertocline etc... are still defined but issue a warning since 1.1a and an error since 1.2.

```
        \def\Etoc@do##1{%
            \expandafter\def\csname etocsaved##1tocline\endcsname
             {\PackageError{etoc}{%
              \expandafter\string\csname etocsaved##1tocline\endcsname\space
              has been deprecated\MessageBreak
              at 1.1a and is removed at 1.2.\MessageBreak
              Use \expandafter\string\csname l@##1\endcsname\space directly.\MessageBreak
              Reported \on@line}%
                               {I will use \expandafter\string
              \csname l@##1\endcsname\space myself for this time.%
                              }%
              \csname l@##1\endcsname
            }%
        }%
        \Etoc@dolevels
    \fi
}%
```

1.09 makes \etocsetnexttocdepth usable in #1 (but this is not 100% compatible with the emptiness check). It makes an \etoclocaltop usable in #1 if under checksemptiness regime.

(#1 above now means \Etoc@tableofcontents@user@before since 1.2)

```
\def\Etoc@tableofcontents@contents{%
    \Etoc@tocdepthset
    \ifEtoc@parskip\parskip\z@skip\fi
    \Etoc@aftertitlehook
```

1.09 has replaced former \Etoc@localtop (*minus one*) by \etoclocaltop. Under checksemptinesstrue regime its value is already known, but it will be obtained again from the toc file execution. As it is used only if TOC is active, resetting it here this way is decorative and could be removed.

```
    \gdef\etoclocaltop{-\thr@@}%
    \Etoc@toctoc
    \etocaftercontentshook
}%
\def\Etoc@tableofcontents@etoc@after{%
    \@nobreakfalse
    \Etoc@restoretocdepth
```

\contentsline was set to \Etoc@etoccontentsline by a non-global \let, and it will recover its normal value from exiting a scope limiting group. But tableof (since 1.4a) under \etocglobaldefs does a global redefinition of \contentsline. Its \tof@finish then does a global restore of \contentsline, but it will be to the etoc set value. \tof@finish is active only if either the table of contents was typeset using \tableof, \tablenotof, \tableoftaggedcontents, or \nextocwithtags was used. If not active it is either undefined (no package tableof) or \@empty. Prior to tableof 1.4c, the \tof@finish closed a group and could be undefined as well, but not if \etocglobaldefs.

If rather than \@empty the \tof@finish fall-back was \relax we could use here \@ifundefined to check in one go (matters of speaking because expansion of \@ifundefined is not in "one-go"). Maybe I should update tableof, but for time being I will simply add an extra test. All this is probably lots of time on irrelevant issue.

```
    \ifx\Etoc@global\global
     \@ifundefined{tof@finish}
     {}
     {\ifx\tof@finish\@empty
      \else
```

```
        \global\let\contentsline\Etoc@savedcontentsline
       \fi
      }%
     \fi
}
```

Refactored at 1.2 to check if the style is actually known and its level is from -1 (-2 in memoir class) to 5 inclusive. If not raise a warning.

```
\def\etocsetstyle#1{\ifcsname Etoc@#1@@\endcsname
                      \expandafter\Etoc@setstyle@a
                    \else
                      \expandafter\Etoc@setstyle@error
                    \fi {#1}%
}
\def\Etoc@setstyle@error #1{%
    \PackageWarning{etoc}{`#1' is unknown to etoc. \space Did you\MessageBreak
                    forget some \string\etocsetlevel{#1}{<level>}?\MessageBreak
                    Reported}%
    \@gobblefour
}
\def\Etoc@setstyle@a #1{%
    \edef\Etoc@tmp{\the\numexpr\csname Etoc@#1@@\endcsname}%
    \if1\unless\ifnum\Etoc@tmp<\Etoc@maxlevel 0\fi
        \unless\ifnum\Etoc@tmp>\Etoc@minf 0\fi1%
      \Etoc@standardlinesfalse
      \expandafter\Etoc@setstyle@b\expandafter\Etoc@tmp
    \else
      \ifnum\Etoc@tmp=\Etoc@maxlevel
        \in@{.#1,}{.figure,.table,}%
        \ifin@
           \PackageWarning{etoc}
              {You can not use \string\etocsetstyle\space with `#1'.\MessageBreak
               Check the package documentation (in particular about\MessageBreak
               \string\etoclocallistoffigureshook/\string\etoclocallistoftableshook)%
               \MessageBreak on how to customize
               figure and table entries in local\MessageBreak lists. Reported}%
        \else
           \PackageInfo{etoc}
              {Attempt to set the style of `#1',\MessageBreak
               whose level is currently the maximal one \etocthemaxlevel,\MessageBreak
               which is never displayed. \space This will be ignored\MessageBreak
               but note that we do quit compatibility mode.\MessageBreak
               Reported}%
           \Etoc@standardlinesfalse
        \fi
      \else
```

This branch was actually in the end only for the case of a level equal to -2, with a document class not memoir, but 1.2a prevents \etocsetlevel to use numerical level -2 in such case.

```
        \PackageWarning{etoc}{This should not happen.  Reported}%
      \fi
      \expandafter\@gobblefour
    \fi
}
```

Prior to 1.2 the #1 here was some alphabetical string such as minusone. We now use digit tokens (and minus sign) in the macro names. The "begin" macros formerly incorporated \Etoc@global\Etoc@isfirsttrue. They are now located in \Etoc@traversestackofends.

1.2 adds \etocstorelinestylesinto. And \etocstorethislinestyleinto. No error check on the level.

```
\long\def\Etoc@setstyle@b#1#2#3#4#5{%
     \expandafter\def\csname Etoc@begin@#1\endcsname    {#2}%
     \expandafter\def\csname Etoc@prefix@#1\endcsname   {#3}%
     \expandafter\def\csname Etoc@contents@#1\endcsname {#4}%
     \expandafter\def\csname Etoc@end@#1\endcsname      {#5}%
}
\def\Etoc@setstyle@e#1{%
     \expandafter\let\csname Etoc@begin@#1\endcsname    \@empty
     \expandafter\let\csname Etoc@prefix@#1\endcsname   \@empty
     \expandafter\let\csname Etoc@contents@#1\endcsname \@empty
     \expandafter\let\csname Etoc@end@#1\endcsname      \@empty
}
\def\Etoc@storelines@a#1{%
   \noexpand\Etoc@setstyle@b{#1}%
     {\expandafter\Etoc@expandonce\csname Etoc@begin@#1\endcsname}%
     {\expandafter\Etoc@expandonce\csname Etoc@prefix@#1\endcsname}%
     {\expandafter\Etoc@expandonce\csname Etoc@contents@#1\endcsname}%
     {\expandafter\Etoc@expandonce\csname Etoc@end@#1\endcsname}%
}
\def\Etoc@expandonce#1{\unexpanded\expandafter{#1}}
\def\etocstorelinestylesinto#1{%
    \edef#1{\Etoc@storelines@a{-2}\Etoc@storelines@a{-1}\Etoc@storelines@a{0}%
            \Etoc@storelines@a {1}\Etoc@storelines@a {2}\Etoc@storelines@a{3}%
            \Etoc@storelines@a {4}\Etoc@storelines@a {5}%
            \ifEtoc@deeplevels
              \Etoc@storelines@a{6}\Etoc@storelines@a{7}\Etoc@storelines@a{8}%
              \Etoc@storelines@a{9}\Etoc@storelines@a{10}\Etoc@storelines@a{11}%
            \fi
    }%
}
\def\etocstorethislinestyleinto#1#2{%
    \edef#2{\expandafter\Etoc@storelines@a\expandafter{\number\etoclevel{#1}}}%
}%
```

Customization macros of the package default line styles.

```
\def\etocfontminustwo {\normalfont \LARGE \bfseries}
\def\etocfontminusone {\normalfont \large \bfseries}
\def\etocfontzero     {\normalfont \large \bfseries}
\def\etocfontone      {\normalfont \normalsize \bfseries}
\def\etocfonttwo      {\normalfont \normalsize}
\def\etocfontthree    {\normalfont \footnotesize}
```

placeholder for comments

```
\def\etocsepminustwo  {4ex \@plus .5ex \@minus .5ex}
\def\etocsepminusone  {4ex \@plus .5ex \@minus .5ex}
\def\etocsepzero      {2.5ex \@plus .4ex \@minus .4ex}
\def\etocsepone       {1.5ex \@plus .3ex \@minus .3ex}
\def\etocseptwo       {.5ex \@plus .1ex \@minus .1ex}
\def\etocsepthree     {.25ex \@plus .05ex \@minus .05ex}
```

placeholder for comments

```
\def\etocbaselinespreadminustwo {1}
\def\etocbaselinespreadminusone {1}
\def\etocbaselinespreadzero     {1}
\def\etocbaselinespreadone      {1}
\def\etocbaselinespreadtwo      {1}
\def\etocbaselinespreadthree    {.9}
```

placeholder for comments

```
\def\etocminustwoleftmargin  {1.5em plus 0.5fil}
\def\etocminustworightmargin {1.5em plus -0.5fil}
\def\etocminusoneleftmargin  {1em}
\def\etocminusonerightmargin {1em}
\def\etoctoclineleaders
        {\hbox{\normalfont\normalsize\hb@xt@2ex {\hss.\hss}}}
\def\etocabbrevpagename {p.~}
```

Versions earlier than 1.08b (and since 1.05 2012/12/01) defined \etocpartname (for use by etoc's own line styles) to expand to \partname. But this didn't make sense in the context for example of babel and babel-french, because \frenchpartname does things depending on the current value of the counter part. The code in recent babel-french (but not yet v2.5a when \etocpartname was introduced) constructs control sequences \ordinali, etc... If the part counter is zero, this gives \ordinal. Usually this is not defined, hence no error happens (as it is constructed via \csname), but under class memoir the bug showed up. All this to explain that I found out about this long lasting problem only on 2015/03/14. Probably a sign that etoc's own line styles are rarely used...

```
\def\etocpartname       {Part}
\def\etocbookname       {Book}
```

The macro \etocdefaultlines was initially called \etoctoclines. Now \etoctoclines just does \Etoc@standardlinesfalse.

Version 1.09f wraps \etocbookname, respectively \etocpartname, in the book, resp. part, line styles inside a (potential) hyperlink together with the number.

1.2a does not define a 'book' style if not with memoir class as the numerical level -2 is made available only with that class.

Unfortunately \@ifclassloaded was preamble-only until the 2021-11-15 LaTeX release (and also \@ifpackageloaded). Anyway let's refactor this \etocdefaultlines at 1.2a to use encapsulating macros rather than be a single one with gigantic contents.

```
\def\etocdefaultlines{%
    \Etoc@standardlinesfalse
    \etocdefaultlines@setbook
    \etocdefaultlines@setpart
    \etocdefaultlines@setchapter
    \etocdefaultlines@setsection
    \etocdefaultlines@setsubsection
    \etocdefaultlines@setsubsubsection
    \etocdefaultlines@setdeeperones
}
```

1.2b adds \etocnoprotrusion as an alias to the LaTeX kernel 2018/12/01 command \noprotrusion which is used since that release in \@dottedtocline after the page number. And it uses it in its own default line styles for section and subsection.

```
\def\etocnoprotrusion{\leavevmode\kern-\p@\kern\p@}
\@ifclassloaded{memoir}{%
 \def\etocdefaultlines@setbook{%
 \Etoc@setstyle@b
  {-2}%
  {\addpenalty\@M\etocskipfirstprefix}
  {\addpenalty\@secpenalty}
  {\begingroup
   \etocfontminustwo
   \addvspace{\etocsepminustwo}%
   \parindent \z@
   \leftskip  \etocminustwoleftmargin
   \rightskip \etocminustworightmargin
   \parfillskip \@flushglue
   \vbox{\etocifnumbered{\etoclink{\etocbookname\enspace\etocthenumber:\quad}}{}%
         \etocname
```

```
          \baselineskip\etocbaselinespreadminustwo\baselineskip
          \par}%
    \addpenalty\@M\addvspace{\etocsepminusone}%
    \endgroup}
   {}%
  }
 }{\let\etocdefaultlines@setbook\@empty}
\def\etocdefaultlines@setpart{%
\Etoc@setstyle@b
   {-1}%
   {\addpenalty\@M\etocskipfirstprefix}
   {\addpenalty\@secpenalty}
   {\begingroup
    \etocfontminusone
    \addvspace{\etocsepminusone}%
    \parindent \z@
    \leftskip  \etocminusoneleftmargin
    \rightskip \etocminusonerightmargin
    \parfillskip \@flushglue
    \vbox{\etocifnumbered{\etoclink{\etocpartname\enspace\etocthenumber.\quad}}{}%
          \etocname
          \baselineskip\etocbaselinespreadminusone\baselineskip
          \par}%
    \addpenalty\@M\addvspace{\etocsepzero}%
    \endgroup}
   {}%
}
```

No need to worry about left protrusion activation and the delicacies of `\leftprotrusion` from microtype, as the paragraph indentation here is nil. So, no changes at 1.2b.

```
\def\etocdefaultlines@setchapter{%
\Etoc@setstyle@b
   {0}%
   {\addpenalty\@M\etocskipfirstprefix}
   {\addpenalty\@itempenalty}
   {\begingroup
    \etocfontzero
    \addvspace{\etocsepzero}%
    \parindent \z@ \parfillskip \@flushglue
    \vbox{\etocifnumbered{\etocnumber.\enspace}{}\etocname
          \baselineskip\etocbaselinespreadzero\baselineskip
          \par}%
    \endgroup}
   {\addpenalty{-\@highpenalty}\addvspace{\etocsepminusone}}%
}
```

This is now very old code, dating back to earliest releases, and looks a bit too convoluted. There was probably a better way with suitable `\nobreak` and `\null` trick. But I am too old to revisit these things now. And, by the way, as paragraph indentation is zeroed, no need to worry about perhaps activating explictly protrusion on the left.

```
\def\etocdefaultlines@setsection{%
\Etoc@setstyle@b
   {1}%
   {\addpenalty\@M\etocskipfirstprefix}
   {\addpenalty\@itempenalty}
   {\begingroup
    \etocfontone
    \addvspace{\etocsepone}%
```

```
    \parindent \z@ \parfillskip \z@
    \setbox\z@\vbox{\parfillskip\@flushglue
                    \etocname\par
                    \setbox\tw@\lastbox
                    \global\setbox\@ne\hbox{\unhbox\tw@\ }}%
    \dimen\z@=\wd\@ne
    \setbox\z@=\etoctoclineleaders
    \advance\dimen\z@\wd\z@
    \etocifnumbered
      {\setbox\tw@\hbox{\etocnumber, \etocabbrevpagename\etocpage\etocnoprotrusion}}
      {\setbox\tw@\hbox{\etocabbrevpagename\etocpage\etocnoprotrusion}}%
    \advance\dimen\z@\wd\tw@
    \ifdim\dimen\z@ < \linewidth
        \vbox{\etocname~%
              \leaders\box\z@\hfil\box\tw@
              \baselineskip\etocbaselinespreadone\baselineskip
              \par}%
    \else
        \vbox{\etocname~%
              \leaders\copy\z@\hfil\break
              \hbox{}\leaders\box\z@\hfil\box\tw@
              \baselineskip\etocbaselinespreadone\baselineskip
              \par}%
    \fi
    \endgroup}
  {\addpenalty\@secpenalty\addvspace{\etocsepzero}}%
}
\def\etocdefaultlines@setsubsection{%
\Etoc@setstyle@b
  {2}%
  {\addpenalty\@medpenalty\etocskipfirstprefix}
  {\addpenalty\@itempenalty}
  {\begingroup
   \etocfonttwo
   \addvspace{\etocseptwo}%
   \parindent \z@ \parfillskip \z@
   \setbox\z@\vbox{\parfillskip\@flushglue
                   \etocname\par\setbox\tw@\lastbox
                   \global\setbox\@ne\hbox{\unhbox\tw@}}%
   \dimen\z@=\wd\@ne
   \setbox\z@=\etoctoclineleaders
   \advance\dimen\z@\wd\z@
   \etocifnumbered
     {\setbox\tw@\hbox{\etocnumber, \etocabbrevpagename\etocpage\etocnoprotrusion}}
     {\setbox\tw@\hbox{\etocabbrevpagename\etocpage\etocnoprotrusion}}%
   \advance\dimen\z@\wd\tw@
   \ifdim\dimen\z@ < \linewidth
       \vbox{\etocname~%
             \leaders\box\z@\hfil\box\tw@
             \baselineskip\etocbaselinespreadtwo\baselineskip
             \par}%
   \else
       \vbox{\etocname~%
             \leaders\copy\z@\hfil\break
             \hbox{}\leaders\box\z@\hfil\box\tw@
             \baselineskip\etocbaselinespreadtwo\baselineskip
             \par}%
```

```
    \fi
    \endgroup}
   {\addpenalty\@secpenalty\addvspace{\etocsepone}}%
}
\def\etocdefaultlines@setsubsubsection{%
\Etoc@setstyle@b
  {3}%
  {\addpenalty\@M
   \etocfontthree
   \vspace{\etocsepthree}%
   \noindent
   \etocskipfirstprefix}
  {\allowbreak\,--\,}
  {\etocname}
  {.\hfil
    \begingroup
     \baselineskip\etocbaselinespreadthree\baselineskip
     \par
    \endgroup
   \addpenalty{-\@highpenalty}}
}
\def\etocdefaultlines@setdeeperones{%
\Etoc@setstyle@e{4}%
\Etoc@setstyle@e{5}%
\ifEtoc@deeplevels
  \Etoc@setstyle@e{6}%
  \Etoc@setstyle@e{7}%
  \Etoc@setstyle@e{8}%
  \Etoc@setstyle@e{9}%
  \Etoc@setstyle@e{10}%
  \Etoc@setstyle@e{11}%
\fi
}
```

The \etocinnertopsep default value is too big as well as \etocbelowtocskip and \etocabovetocskip, I guess, but if I am remember correctly I chose them to mimic the standard TOC spacings in `article` class.

```
\def\etocabovetocskip{3.5ex \@plus 1ex \@minus .2ex}
\def\etocbelowtocskip{3.5ex \@plus 1ex \@minus .2ex}
\def\etoccolumnsep{2em}
\def\etocmulticolsep{0ex}
\def\etocmulticolpretolerance{-1}
\def\etocmulticoltolerance{200}
\def\etocdefaultnbcol{2}
\def\etocinnertopsep{2ex}
\newcommand\etocmulticolstyle[2][\etocdefaultnbcol]{%
\etocsettocstyle
   {\let\etocoldpar\par
    \addvspace{\etocabovetocskip}%
    \ifnum #1>\@ne
          \expandafter\@firstoftwo
    \else \expandafter\@secondoftwo
    \fi
    {\multicolpretolerance\etocmulticolpretolerance
    \multicoltolerance\etocmulticoltolerance
    \setlength{\columnsep}{\etoccolumnsep}%
    \setlength{\multicolsep}{\etocmulticolsep}%
    \begin{multicols}{#1}[#2\etocoldpar\addvspace{\etocinnertopsep}]}
    {#2\ifvmode\else\begingroup\interlinepenalty\@M\parskip\z@skip
```

```
                    \@@par\endgroup
      \fi
      \nobreak\addvspace{\etocinnertopsep}%
      \pretolerance\etocmulticolpretolerance
      \tolerance\etocmulticoltolerance}%
  }%
  {\ifnum #1>\@ne
        \expandafter\@firstofone
   \else \expandafter\@gobble
   \fi
   {\end{multicols}}%
   \addvspace{\etocbelowtocskip}}%
}
```

placeholder for comments

```
\def\etocinnerbottomsep{3.5ex}
\def\etocinnerleftsep{2em}
\def\etocinnerrightsep{2em}
\def\etoctoprule{\hrule}
\def\etocleftrule{\vrule}
\def\etocrightrule{\vrule}
\def\etocbottomrule{\hrule}
\def\etoctoprulecolorcmd{\relax}
\def\etocbottomrulecolorcmd{\relax}
\def\etocleftrulecolorcmd{\relax}
\def\etocrightrulecolorcmd{\relax}
```

placeholder

```
\def\etoc@ruledheading #1{%
  \hb@xt@\linewidth{\color@begingroup
        \hss #1\hss\hskip-\linewidth
        \etoctoprulecolorcmd\leaders\etoctoprule\hss
        \phantom{#1}%
        \leaders\etoctoprule\hss\color@endgroup}%
        \nointerlineskip\nobreak\vskip\etocinnertopsep}
\newcommand*\etocruledstyle[2][\etocdefaultnbcol]{%
\etocsettocstyle
  {\addvspace{\etocabovetocskip}%
   \ifnum #1>\@ne
        \expandafter\@firstoftwo
   \else \expandafter\@secondoftwo
   \fi
      {\multicolpretolerance\etocmulticolpretolerance
       \multicoltolerance\etocmulticoltolerance
       \setlength{\columnsep}{\etoccolumnsep}%
       \setlength{\multicolsep}{\etocmulticolsep}%
       \begin{multicols}{#1}[\etoc@ruledheading{#2}]}
      {\etoc@ruledheading{#2}%
        \pretolerance\etocmulticolpretolerance
        \tolerance\etocmulticoltolerance}}
  {\ifnum #1>\@ne\expandafter\@firstofone
        \else \expandafter\@gobble
   \fi
   {\end{multicols}}%
   \addvspace{\etocbelowtocskip}}}
```

placeholder

```
\def\etocframedmphook{\relax}
\long\def\etocbkgcolorcmd{\relax}
```

```
\long\def\Etoc@relax{\relax}
```

placeholder for comments

```
\newbox\etoc@framed@titlebox
\newbox\etoc@framed@contentsbox
\newcommand*\etocframedstyle[2][\etocdefaultnbcol]{%
\etocsettocstyle{%
    \addvspace{\etocabovetocskip}%
    \sbox\z@{#2}%
    \dimen\z@\dp\z@
    \ifdim\wd\z@<\linewidth \dp\z@\z@ \else \dimen\z@\z@ \fi
    \setbox\etoc@framed@titlebox=\hb@xt@\linewidth{\color@begingroup
        \hss
        \ifx\etocbkgcolorcmd\Etoc@relax
        \else
            \sbox\tw@{\color{white}%
            \vrule\@width\wd\z@\@height\ht\z@\@depth\dimen\z@}%
            \ifdim\wd\z@<\linewidth \dp\tw@\z@\fi
            \box\tw@
            \hskip-\wd\z@
        \fi
        \copy\z@
        \hss
        \hskip-\linewidth
        \etoctoprulecolorcmd\leaders\etoctoprule\hss
        \hskip\wd\z@
        \etoctoprulecolorcmd\leaders\etoctoprule\hss\color@endgroup}%
    \setbox\z@\hbox{\etocleftrule\etocrightrule}%
    \dimen\tw@\linewidth\advance\dimen\tw@-\wd\z@
        \advance\dimen\tw@-\etocinnerleftsep
        \advance\dimen\tw@-\etocinnerrightsep
    \setbox\etoc@framed@contentsbox=\vbox\bgroup
        \hsize\dimen\tw@
        \kern\dimen\z@
        \vskip\etocinnertopsep
        \hbox\bgroup
        \begin{minipage}{\hsize}%
        \etocframedmphook
    \ifnum #1>\@ne
          \expandafter\@firstoftwo
    \else \expandafter\@secondoftwo
    \fi
        {\multicolpretolerance\etocmulticolpretolerance
        \multicoltolerance\etocmulticoltolerance
        \setlength{\columnsep}{\etoccolumnsep}%
        \setlength{\multicolsep}{\etocmulticolsep}%
        \begin{multicols}{#1}}
        {\pretolerance\etocmulticolpretolerance
         \tolerance\etocmulticoltolerance}}
    {\ifnum #1>\@ne\expandafter\@firstofone
         \else \expandafter\@gobble
     \fi
      {\end{multicols}\unskip }%
    \end{minipage}%
    \egroup
    \vskip\etocinnerbottomsep
    \egroup
    \vbox{\hsize\linewidth
```

```
        \ifx\etocbkgcolorcmd\Etoc@relax
        \else
            \kern\ht\etoc@framed@titlebox
            \kern\dp\etoc@framed@titlebox
            \hb@xt@\linewidth{\color@begingroup
            \etocleftrulecolorcmd\etocleftrule
            \etocbkgcolorcmd
            \leaders\vrule
                   \@height\ht\etoc@framed@contentsbox
                   \@depth\dp\etoc@framed@contentsbox
            \hss
            \etocrightrulecolorcmd\etocrightrule
            \color@endgroup}\nointerlineskip
            \vskip-\dp\etoc@framed@contentsbox
            \vskip-\ht\etoc@framed@contentsbox
            \vskip-\dp\etoc@framed@titlebox
            \vskip-\ht\etoc@framed@titlebox
        \fi
    \box\etoc@framed@titlebox\nointerlineskip
    \hb@xt@\linewidth{\color@begingroup
    {\etocleftrulecolorcmd\etocleftrule}%
    \hss\box\etoc@framed@contentsbox\hss
    \etocrightrulecolorcmd\etocrightrule\color@endgroup}
    \nointerlineskip
    \vskip\ht\etoc@framed@contentsbox
    \vskip\dp\etoc@framed@contentsbox
    \hb@xt@\linewidth{\color@begingroup\etocbottomrulecolorcmd
          \leaders\etocbottomrule\hss\color@endgroup}}
    \addvspace{\etocbelowtocskip}}}
```

For time being I will not however add the versions for "lists of", as anyhow probably nobody apart myself ever uses these things.

It is impossible to know what kind of division heading #2 uses, so there is not much I can do here at 1.2 apart from providing a user interface for adding the suitable thing to the .toc file. And I discover at time of writing (finishing the 1.2 documentation) that I already have \etoclocalheadtotoc. I only need to add \etocglobalheadtotoc.

```
\newcommand\etoc@multicoltoc[2][\etocdefaultnbcol]{%
    \etocmulticolstyle[#1]{#2}%
    \tableofcontents}
\newcommand\etoc@multicoltoci[2][\etocdefaultnbcol]{%
    \etocmulticolstyle[#1]{#2}%
    \tableofcontents*}
\newcommand\etoc@local@multicoltoc[2][\etocdefaultnbcol]{%
    \etocmulticolstyle[#1]{#2}%
    \localtableofcontents}
\newcommand\etoc@local@multicoltoci[2][\etocdefaultnbcol]{%
    \etocmulticolstyle[#1]{#2}%
    \localtableofcontents*}
```

placeholder for comments

```
\newcommand*\etoc@ruledtoc[2][\etocdefaultnbcol]{%
    \etocruledstyle[#1]{#2}%
    \tableofcontents}
\newcommand*\etoc@ruledtoci[2][\etocdefaultnbcol]{%
    \etocruledstyle[#1]{#2}%
    \tableofcontents*}
\newcommand*\etoc@local@ruledtoc[2][\etocdefaultnbcol]{%
    \etocruledstyle[#1]{#2}%
```

```
    \localtableofcontents}
\newcommand*\etoc@local@ruledtoci[2][\etocdefaultnbcol]{%
    \etocruledstyle[#1]{#2}%
    \localtableofcontents*}
```

placeholder for comments

```
\newcommand*\etoc@framedtoc[2][\etocdefaultnbcol]{%
    \etocframedstyle[#1]{#2}%
    \tableofcontents}
\newcommand*\etoc@framedtoci[2][\etocdefaultnbcol]{%
    \etocframedstyle[#1]{#2}%
    \tableofcontents*}
\newcommand*\etoc@local@framedtoc[2][\etocdefaultnbcol]{%
    \etocframedstyle[#1]{#2}%
    \localtableofcontents}
\newcommand*\etoc@local@framedtoci[2][\etocdefaultnbcol]{%
    \etocframedstyle[#1]{#2}%
    \localtableofcontents*}
```

placeholder for comments

```
\def\etocmulticol{\begingroup
    \Etoc@mustclosegrouptrue
    \@ifstar
    {\etoc@multicoltoci}
    {\etoc@multicoltoc}}
\def\etocruled{\begingroup
    \Etoc@mustclosegrouptrue
    \@ifstar
    {\etoc@ruledtoci}
    {\etoc@ruledtoc}}
\def\etocframed{\begingroup
    \Etoc@mustclosegrouptrue
    \@ifstar
    {\etoc@framedtoci}
    {\etoc@framedtoc}}
\def\etoclocalmulticol{\begingroup
    \Etoc@mustclosegrouptrue
    \@ifstar
    {\etoc@local@multicoltoci}
    {\etoc@local@multicoltoc}}
\def\etoclocalruled{\begingroup
    \Etoc@mustclosegrouptrue
    \@ifstar
    {\etoc@local@ruledtoci}
    {\etoc@local@ruledtoc}}
\def\etoclocalframed{\begingroup
    \Etoc@mustclosegrouptrue
    \@ifstar
    {\etoc@local@framedtoci}
    {\etoc@local@framedtoc}}
```

1.2 makes local TOCs in compatibility display style use `\subsection*` rather than `\section*`. This is not good for local TOCs to a `\part` but anyhow the default is via `\etocetoclocaltocstyle` which will do the right thing and the change here is irrelevant. More comments below about handling memoir and KOMA-script.

The macros next will be modified if under book or memoir class. The KOMA-script case uses rather the `leveldown` mechanism. These local TOCs things are used only under `\etocclasstocstyle`. With `\etocetoclocaltocstyle`, they are not used.

```
\def\etocmemoirtoctotocfmt #1#2{%
```

```
    \PackageWarning{etoc}
        {\string\etocmemoirtoctotocfmt\space is deprecated.\MessageBreak
         Use in its place \string\etocsettoclineforclasstoc,\MessageBreak
         and \string\etocsettoclineforclasslistof{toc} (or {lof}, {lot}).
         I will do this now.\MessageBreak
         Reported}%
    \etocsettoclineforclasstoc{#1}{#2}%
    \etocsettoclineforclasslistof{toc}{#1}{#2}%
}
\def\etocsettoclineforclasstoc #1#2{%
    \def\etocclassmaintocaddtotoc{\etocglobalheadtotoc{#1}{#2}}%
}
\def\etocsettoclineforclasslistof #1#2#3{%
    \@namedef{etocclasslocal#1addtotoc}{\etoclocalheadtotoc{#2}{#3}}%
}
\let\etocclasslocaltocaddtotoc\@empty
\let\etocclasslocallofaddtotoc\@empty
\let\etocclasslocallotaddtotoc\@empty
\ifdefined\c@chapter
  \def\etocclasslocaltocmaketitle{\section*{\localcontentsname}}
  \def\etocclasslocallofmaketitle{\section*{\locallistfigurename}}
  \def\etocclasslocallotmaketitle{\section*{\locallisttablename}}
  \etocsettoclineforclasstoc        {chapter}{\contentsname}
  \etocsettoclineforclasslistof{toc}{section}{\localcontentsname}
  \etocsettoclineforclasslistof{lof}{section}{\locallistfigurename}
  \etocsettoclineforclasslistof{lot}{section}{\locallisttablename}
\else
  \def\etocclasslocaltocmaketitle{\subsection*{\localcontentsname}}%
  \def\etocclasslocallofmaketitle{\subsection*{\locallistfigurename}}%
  \def\etocclasslocallotmaketitle{\subsection*{\locallisttablename}}%
  \etocsettoclineforclasstoc        {section}{\contentsname}
  \etocsettoclineforclasslistof{toc}{subsection}{\localcontentsname}
  \etocsettoclineforclasslistof{lof}{subsection}{\locallistfigurename}
  \etocsettoclineforclasslistof{lot}{subsection}{\locallisttablename}
\fi
```

This is moved to a macro to localize complications with conditionals.

```
\def\etocclasslocalperhapsaddtotoc #1{%
    \etocifisstarred
      {}
      {\csname ifEtoc@local#1totoc\endcsname
         \csname etocclasslocal#1addtotoc\endcsname
       \fi
      }%
}
```

No need for a \phantomsection if the \addcontentsline is after \section*. For the standard classes, I make no effort to adjust level used for the local heading if it is local to a \part, not a \section. Anyhow this code will not be used by default for local TOCs due to \etocetoclocaltocstyle.

```
\def\etocarticlestyle{%
    \etocsettocstyle
    {\ifEtoc@localtoc
       \@nameuse{etocclasslocal\Etoc@currext maketitle}%
       \etocclasslocalperhapsaddtotoc\Etoc@currext
     \else
       \section *{\contentsname
                  \@mkboth {\MakeUppercase \contentsname}
                           {\MakeUppercase \contentsname}}%
```

```
      \etocifisstarred{}{\etocifmaintoctotoc{\etocclassmaintocaddtotoc}{}}%
    \fi
   }
   {}%
}
\def\etocarticlestylenomarks{%
   \etocsettocstyle
   {\ifEtoc@localtoc
      \@nameuse{etocclasslocal\Etoc@currext maketitle}%
      \etocclasslocalperhapsaddtotoc\Etoc@currext
    \else
      \section *{\contentsname}%
      \etocifisstarred{}{\etocifmaintoctotoc{\etocclassmaintocaddtotoc}{}}%
    \fi
   }
   {}%
}
```

Make definitions with book in the macro names or redefine them for book? Chose the latter.

```
\def\etocbookstyle{%
   \etocsettocstyle
   {\if@twocolumn \@restonecoltrue \onecolumn \else \@restonecolfalse \fi
    \ifEtoc@localtoc
      \@nameuse{etocclasslocal\Etoc@currext maketitle}%
      \etocclasslocalperhapsaddtotoc\Etoc@currext
    \else
      \chapter *{\contentsname
                 \@mkboth {\MakeUppercase \contentsname}
                          {\MakeUppercase \contentsname}}%
      \etocifisstarred{}{\etocifmaintoctotoc{\etocclassmaintocaddtotoc}{}}%
    \fi
   }%
   {\if@restonecol \twocolumn \fi}%
}
\def\etocbookstylenomarks{%
   \etocsettocstyle
   {\if@twocolumn \@restonecoltrue \onecolumn \else \@restonecolfalse \fi
    \ifEtoc@localtoc
      \@nameuse{etocclasslocal\Etoc@currext maketitle}%
      \etocclasslocalperhapsaddtotoc\Etoc@currext
    \else
      \chapter *{\contentsname}%
      \etocifisstarred{}{\etocifmaintoctotoc{\etocclassmaintocaddtotoc}{}}%
    \fi
   }%
   {\if@restonecol \twocolumn \fi}%
}
\let\etocreportstyle\etocbookstyle
\let\etocreportstylenomarks\etocbookstylenomarks
```

v3.7i of memoir has moved the `\phantomsection` to a better location, before typesetting the title and we follow suit at 1.09a, and at 1.09b. Formerly **etoc** used `\etocaftertitlehook` to mimic the memoir code but as its name indicate, it is supposedly executed after the title... and this also had the defect of making `\etocaftertitlehook` not anymore a user command. Thus we here use some refactoring of the `\Etoc@aftertitlehook` internal mechanism to help recognize if we are in the starred case or not.

`\phantomsection` is always defined by memoir, empty if hyperref absent.

Updates at 1.2 for the standard display style inherited from the class to actually be usable with local table of contents. Unfortunately the printing of the title rigidly hard-codes a `\thispagestyle{chapter}`:

```
\@lofmaketitle ->\@nameuse {lofheadstart} {\parindent \z@ \parskip \z@ \interli
nepenalty \@M \@nameuse {printlofnonum}\@nameuse {printloftitle}{\listfigurenam
e }\@nameuse {lofmark}\thispagestyle {chapter}\@nameuse {afterloftitle} } \@aft
erheading
```

Oh well I discover there is a \chapterheadstart but not \sectionheadstart...

Actually now etoc will by default not use the class inherited style for the local TOC titles, so maybe I should have not wasted time on this. From the \etocsettocstyle arguments one can know if this is for a local TOC, but one can not know the level, which is inferred from the actual execution of the TOC data. There is a problem though that the memoir default would be usable for a local TOC in a \part. Although one can argue if it makes sense to display a TOC as prominently as a chapter anyway. Again, the default with etoc will anyhow not use this as \etocetoclocaltocstyle deactivates the \etocsettocstyle for local TOCs.

MEMO: would it make sense to have an \etocsetlocaltocstyle? As explained above this would however induced dramatic internal etoc changes as one would have to wait for title insertion until the time one knows the local level. Which is exactly what \etocetoclocaltocstyle via the \ifEtoc@etocstyle does.

Update at 1.2 to account for \locallistoffigures and \locallistoftables.

```
\def\etocmemoirstyle{%
    \etocsettocstyle
        {\ensureonecol \par \begingroup \phantomsection
         \ifx\Etoc@aftertitlehook\@empty
         \else
```

This branch executed for the non-starred variant

```
           \ifmem@em@starred@listof
           \else
```

Global case is only for \tableofcontents as etoc does not hack into \listofpictures and \listoftables.

Local case will not be executed in default configurations (cf \etocetoclocaltocstyle).

\etocclasslocalperhapsaddtotoc is to avoid worries with conditionals.

```
             \ifEtoc@localtoc
                 \etocclasslocalperhapsaddtotoc\Etoc@currext
             \else
               \ifEtoc@maintoctotoc
                  \etocclassmaintocaddtotoc
               \fi
             \fi
           \fi
         \fi
         \ifEtoc@localtoc
```

Trying to mimic a section title but there is no \sectionheadstart there is no \printsectiontitle etc... Oh well I will be more radical then.

```
            \@namedef{@\Etoc@currext maketitle}{%
               \@nameuse{etocclasslocal\Etoc@currext maketitle}%
            }%
         \fi
         \@nameuse {@\Etoc@currext maketitle} %<< space token here from memoir code
         \ifx\Etoc@aftertitlehook\@empty
          \else
```

Execute etoc hook before the \cfttocbeforelisthook and keep distinction between starred and non-starred contexts for other hooks (by testing if \Etoc@aftertitlehook is empty or not). Notice that the memoir class way of implementing \tableofcontents leaves no way for code executed by the TOC code to know if it is executed in starred or non-starred context.

```
          \Etoc@aftertitlehook \let \Etoc@aftertitlehook \relax
         \fi
         \parskip \cftparskip \@nameuse {cft\Etoc@currext beforelisthook}%
```

```
        }%
        {\@nameuse {cft\Etoc@currext afterlisthook}%
         \endgroup\restorefromonecol
        }%
}
```

Compatibility layer for the KOMA-script classes:

1.09c (2020/05/15) did an update as KOMA-script deprecated `\iftocfeature`. Thanks to Bilel Omrani for report. I did not check if cloning of KOMA code required some further updates. 1.09f added more updates.

At 1.2 this is further updated to be usable also for list of figures and tables. However, this update is somewhat theoretical because **etoc** does not interfere at all with `\listoffigures` and `\listoftables`: the update is useful only to make **etoc**'s `\locallistoffigures` and `\locallistoftables` usable in display style compatibility mode.

But during development on this I became aware that KOMA-script since its `3.30` release has the feature that an unnumbered section resets the counters of subsections. This creates a problem (whose description I have moved to the user manual) which can be alleviated for local TOCs at the highest level below global one by using KOMA's `\setuptoc{toc}{leveldown}`. So I decided to do this systematically, as in the code next via a new private hook, but the problem will remain for local TOCs at lower levels, and there does not seem to be any way to tell KOMA to use say `\subsubsection*`, barring, from what I understand from the manual, usage of its `\deftocheading`. So in a second stage I decided that per default **etoc** would rather use for local TOCs, and not only for KOMA classes but all classes, an adaptive heading fitted to the "local top". As this "local top" can only be determined from inside the expansion of `\Etoc@toctoc` which contains the `.toc` file data, the boolean `\ifEtoc@etocstyle` was added which will make `\etocsettocstyle` configuration ignored. In this way, with `\etocetoclocaltocstyle`, the document class emulation will apply to the global TOC whereas local TOCs will use the adaptive scheme. To avoid duplication other relevant info is moved to the user manual.

We put the trigger of `leveldown` for KOMA classes in `\Etoc@beforetitlehook`. So there will be no test with `\ifEtoc@localtoc` here contrarily to the case of standard classes and memoir. Again, all of this normally is not relevant as by default **etoc** 1.2 will use its `\etocetoclocaltocstyle` which for local TOCs ignores the emulation code of the main TOC.

```
\let\Etoc@beforetitlehook\@empty
\if1\@ifclassloaded{scrartcl}0{\@ifclassloaded{scrbook}0{\@ifclassloaded{scrreprt}01}}%
\expandafter\@gobble
\else
```

Surely paranoid here but I don't have time to go through KOMA documentation (I am not really familiar with these classes).

MEMO: I do not know if KOMA-script's `\setuptoc` sets options locally or globally. If globally the code below must be modifed to unset the `totoc` option depending on at time of use status of the **etoc** own `totoc` options.

```
   \ifdefined\setuptoc
     \def\Etoc@beforetitlehook{%
         \ifEtoc@localtoc
            \etocclasslocalperhapsaddtotoc\Etoc@currext
            \setuptoc{\Etoc@currext}{leveldown}%
         \else
            \etocifisstarred{}{\etocifmaintoctotoc{\setuptoc{toc}{totoc}}}%
         \fi
     }%
   \fi
\expandafter\@firstofone
\fi
{\def\etocclasslocalperhapsaddtotoc #1{%
    \etocifisstarred
    {}%
    {\csname ifEtoc@local#1totoc\endcsname
      \setuptoc{\Etoc@currext}{totoc}%
```

```
      \fi
    }%
  }%
}
\ifdefined\Iftocfeature
  \def\etoc@Iftocfeature{\Iftocfeature}%
\else
  \def\etoc@Iftocfeature{\iftocfeature}%
\fi
```

Peut-être en fait je devrais toujours faire `\let\if@dynlist\if@tocleft`? Mais je ne l'ai pas vu dans le code de KOMA pour les LOF et LOT (globales, évidemment). Mais cela m'obligerait à lire vraiment le code source de KOMA. Pas le temps.

```
\def\etocscrartclstyle{%
    \etocsettocstyle
        {\ifx\Etoc@currext\Etoc@tocext
             \expandafter\@firstofone
         \else
             \expandafter\@gobble
         \fi
         {\let\if@dynlist\if@tocleft}%
         \edef\@currext{\Etoc@currext}%
         \@ifundefined{listof\@currext name}%
            {\def\list@fname{\listofname~\@currext}}%
            {\expandafter\let\expandafter\list@fname
                        \csname listof\@currext name\endcsname}%
         \etoc@Iftocfeature {\@currext}{onecolumn}
             {\etoc@Iftocfeature {\@currext}{leveldown}
              {}
              {\if@twocolumn \aftergroup \twocolumn \onecolumn \fi }}
             {}%
         \etoc@Iftocfeature {\@currext}{numberline}%
                           {\def \nonumberline {\numberline {}}}{}%
         \expandafter\tocbasic@listhead\expandafter {\list@fname}%
         \begingroup \expandafter \expandafter \expandafter
         \endgroup \expandafter
         \ifx
             \csname microtypesetup\endcsname \relax
         \else
             \etoc@Iftocfeature {\@currext}{noprotrusion}{}
                 {\microtypesetup {protrusion=false}%
                  \PackageInfo {tocbasic}%
                  {character protrusion at \@currext\space deactivated}}%
         \fi
         \etoc@Iftocfeature{\@currext}{noparskipfake}{}{%
               \ifvmode \@tempskipa\lastskip \vskip-\lastskip
               \addtolength{\@tempskipa}{\parskip}\vskip\@tempskipa\fi
               }%
         \setlength {\parskip }{\z@ }%
         \setlength {\parindent }{\z@ }%
         \setlength {\parfillskip }{\z@ \@plus 1fil}%
         \csname tocbasic@@before@hook\endcsname
         \csname tb@\@currext @before@hook\endcsname
       }% end of before_toc
       {% start of after_toc
```

(This next discussion has not been revised at 1.2 so let's hope it is fine).

At 1.09f I considered adding this `\BeforeClosingMainAux` hunk to the second argument of `\etocsettocstyle`-emulation of KOMA-script. But:

- there seems to be no interface to \tocbasic@end@toc@file,
- it defaults to issuing a \par, but we want etoc to still be able to produce other TOCs, possibly inline, and they should not be influenced by it and I don't want at this stage to add an interface to enable/disable and have to document it,
- the whole thing appears to me to be ill-conceived in so far as it sort of implies the \tableofcontents is used only once, as each instance will again add this \tocbasic@end@toc@file to end of toc file, which may thus end up being executed multiple times.

So rather than putting the thing in the .toc file, we will execute it here. This way it will not impact other TOCs typeset via etoc design facilities in the document.

```
%           \BeforeClosingMainAux
%           {\addtocontents
%             {toc}{\string\providecommand\string\tocbasic@end@toc@file{}%
%                   \string\tocbasic@end@toc@file}%
%           }%
          \providecommand\tocbasic@end@toc@file{}\tocbasic@end@toc@file
          \edef\@currext{\Etoc@currext}%
          \csname tb@\@currext @after@hook\endcsname
          \csname tocbasic@@after@hook\endcsname
      }% end of after_toc
}
\let\etocscrbookstyle\etocscrartclstyle
\let\etocscrreprtstyle\etocscrartclstyle
```

The \etocclasstocstyle will be redefined according to document class. Then, later, it will be extended with an \Etoc@classstyletrue.

```
\def\etocclasstocstyle{\etocarticlestyle}
\newcommand*\etocmarkboth[1]{%
    \@mkboth{\MakeUppercase{#1}}{\MakeUppercase{#1}}}
\newcommand*\etocmarkbothnouc[1]{\@mkboth{#1}{#1}}
\newcommand\etoctocstyle[3][section]{\etocmulticolstyle[#2]%
    {\csname #1\endcsname *{#3}}}
\newcommand\etoctocstylewithmarks[4][section]{\etocmulticolstyle[#2]%
    {\csname #1\endcsname *{#3\etocmarkboth{#4}}}}
\newcommand\etoctocstylewithmarksnouc[4][section]{\etocmulticolstyle[#2]%
    {\csname #1\endcsname *{#3\etocmarkbothnouc{#4}}}}
\def\Etoc@redefetocstylesforchapters{%
    \renewcommand\etoctocstylewithmarks[4][chapter]{%
          \etocmulticolstyle[##2]{\csname ##1\endcsname *{##3\etocmarkboth{##4}}}%
    }
    \renewcommand\etoctocstylewithmarksnouc[4][chapter]{%
          \etocmulticolstyle[##2]{\csname ##1\endcsname *{##3\etocmarkbothnouc{##4}}}%
    }
    \renewcommand\etoctocstyle[3][chapter]{%
          \etocmulticolstyle[##2]{\csname ##1\endcsname *{##3}}
    }
}
\@ifclassloaded{scrartcl}
    {\renewcommand*\etocclasstocstyle{\etocscrartclstyle}}{}
\@ifclassloaded{book}
    {\renewcommand*\etocfontone{\normalfont\normalsize}
     \renewcommand*\etocclasstocstyle{\etocbookstyle}
     \Etoc@redefetocstylesforchapters}{}
\@ifclassloaded{report}
    {\renewcommand*\etocfontone{\normalfont\normalsize}
     \renewcommand*\etocclasstocstyle{\etocreportstyle}
     \Etoc@redefetocstylesforchapters}{}
\@ifclassloaded{scrbook}
```

```
    {\renewcommand*\etocfontone{\normalfont\normalsize}
     \renewcommand*\etocclasstocstyle{\etocscrbookstyle}
     \Etoc@redefetocstylesforchapters}{}
\@ifclassloaded{scrreprt}
    {\renewcommand*\etocfontone{\normalfont\normalsize}
     \renewcommand*\etocclasstocstyle{\etocscrreprtstyle}
     \Etoc@redefetocstylesforchapters}{}
\@ifclassloaded{memoir}
    {\renewcommand*\etocfontone{\normalfont\normalsize}
     \renewcommand*\etocclasstocstyle{\etocmemoirstyle}
     \Etoc@redefetocstylesforchapters}{}
\def\etoctocloftstyle {%
    \etocsettocstyle{%
      \@cfttocstart
      \par
      \begingroup
        \parindent\z@ \parskip\cftparskip
```

I don't feel like redefining `\@cftmaketoctitle` etc to apply "level down". Up to the user to use the tocloft interface after the main TOC to do appropriate actions. I consider emitting a warning, but then if the user has customize `\@cftmaketoctitle` or variant, how to know? (apart from numerous ifx tests).

```
        \@nameuse{@cftmake\Etoc@currext title}%
```

But for adding an appropriate entry to the toc file I can intervene silently. I can remove the test `\if@cfttocbibind` here as anyhow I have to test status of `etoc` 'to toc' options.

```
        \ifEtoc@localtoc
           \etoctocloftlocalperhapsaddtotoc\Etoc@currext
        \else
           \etocifisstarred {}{\ifEtoc@maintoctotoc\@cftdobibtoc\fi}%
        \fi
    }%
    {%
      \endgroup
      \@cfttocfinish
    }%
}
\def\etoctocloftlocalperhapsaddtotoc#1{%
    \etocifisstarred
      {}%
      {\csname ifEtoc@local#1totoc\endcsname
        \ifdefined\c@chapter\def\@tocextra{@section}\else\def\@tocextra{@subsection}\fi
        \csname @cftdobib#1\endcsname
       \fi
      }%
}
```

This is active only if tocbibind boolean `\if@dotoctoc` is found true at begin document and there was no use of `\etocsettocstyle` in the preamble. The part on local TOC also applies to local LOF and LOT but is executed only if `\etocclasstocstyle` was present in the preamble. Under the default `\etocetoclocaltocstyle`, only the global TOC is under influence of this (assuming thus that tocbibind was loaded without its `nottoc` or none option).

```
\def\etoctocbibindstyle {%
    \etocsettocstyle {%
      \toc@start
      \ifEtoc@localtoc
          \@nameuse{etocclasslocal\Etoc@currext maketitle}%
          \etocclasslocalperhapsaddtotoc\Etoc@currext
      \else
          \etoc@tocbibind@dotoctitle
```

```
      \fi
      }%
    {\toc@finish}%
}
\def\etoc@tocbibind@dotoctitle {%
```

If tocbibind is loaded but later between \begin{document} and \tableofcontents the user does \etocsetup{maintoctotoc=afalse} we have to catch this.

```
       \if@bibchapter
        \etocifisstarred
          {\chapter*{\contentsname}\prw@mkboth{\contentsname} % id.
          }%
          {\ifEtoc@maintoctotoc
             \toc@chapter{\contentsname} %<-space from original
           \else
             \chapter*{\contentsname}\prw@mkboth{\contentsname} % id.
           \fi
           }%
       \else
        \etocifisstarred
          {\@nameuse{\@tocextra}*{\contentsname\prw@mkboth{\contentsname}} %<-space
          }
          {\ifEtoc@maintoctotoc
             \toc@section{\@tocextra}{\contentsname} %<-space from original
           \else
             \@nameuse{\@tocextra}*{\contentsname\prw@mkboth{\contentsname}} % id.
          \fi
          }%
       \fi
}%
```

The 1.2 added a bug here in the non-memoir tocloft branch, the needed \AtBeginDocument was removed by accident and as a result the overwriting by tocloft of \tableofcontents was not undone.

```
\@ifclassloaded{memoir}
{}
{% memoir not loaded
 \@ifpackageloaded{tocloft}
    {\if@cftnctoc\else
                \ifEtoc@keeporiginaltoc
                  \else
                  \AtBeginDocument{\let\tableofcontents\etoctableofcontents}%
                \fi
     \fi }
    {\AtBeginDocument
      {\@ifpackageloaded{tocloft}
       {\if@cftnctoc\else
         \PackageWarningNoLine {etoc}
         {Package `tocloft' was loaded after `etoc'.\MessageBreak
          To prevent it from overwriting \protect\tableofcontents, it will\MessageBreak
          be tricked into believing to have been loaded with its\MessageBreak
          option `titles'. \space But this will cause the `tocloft'\MessageBreak
          customization of the titles of the main list of figures\MessageBreak
          and list of tables to not apply either.\MessageBreak
          You should load `tocloft' before `etoc'.}%
        \AtEndDocument{\PackageWarning{etoc}
                       {Please load `tocloft' before `etoc'!\@gobbletwo}}%
        \fi
        \@cftnctoctrue }%
```

```
        {}%
       }%
      }%
}
\@ifclassloaded{memoir}
{}
{% memoir not loaded
  \AtBeginDocument{%
  \@ifpackageloaded{tocloft}
   {%
     \def\etocclasstocstyle{%
       \etoctocloftstyle
       \Etoc@classstyletrue
     }%
     \ifEtoc@etocstyle
        \ifEtoc@classstyle
          \etocclasstocstyle
          \Etoc@etocstyletrue
        \fi
     \else
        \ifEtoc@classstyle
          \etocclasstocstyle
        \fi
     \fi
   }%
   {% no tocloft
    \@ifpackageloaded {tocbibind}
     {\if@dotoctoc
        \def\etocclasstocstyle{%
          \etoctocbibindstyle
          \Etoc@classstyletrue
         }%
        \ifEtoc@etocstyle
           \ifEtoc@classstyle
             \etocclasstocstyle
             \Etoc@etocstyletrue
           \fi
        \else
           \ifEtoc@classstyle
             \etocclasstocstyle
           \fi
        \fi
```

Not clear if I should interpret `\Etoc@keeporiginaltoctrue` in this sense to not modify tocbibind overwrite of `\tableofcontents`. But let's obey this interpretation.

```
        \ifEtoc@keeporiginaltoc
        \else
           \let\tableofcontents\etoctableofcontents
        \fi
     }%
     {}%
   }%
```

Maybe I should check if the options were already set. I will simply make the message more generic.

```
   \@ifpackageloaded{tocbibind}
   {% tocbibind, perhaps with tocloft
      \if@dotoctoc
                 \ifEtoc@keeporiginaltoc
```

```
                    \else
                    \let\tableofcontents\etoctableofcontents
                  \fi
        \etocsetup{maintoctotoc,localtoctotoc}%
         \PackageInfo{etoc}{%
            Setting (or re-setting) the options `maintoctotoc' and\MessageBreak
            `localtoctotoc' to true as tocbibind was detected and\MessageBreak
            found to be configured for `TOC to toc'.\MessageBreak
            Reported at begin document}%
      \fi
      \if@dotoclof
       \ifEtoc@lof
        \etocsetup{localloftotoc}%
        \PackageInfo{etoc}{%
            Setting (or re-setting) `localloftotoc=true' as the\MessageBreak
            package tocbibind was detected and is configured for\MessageBreak
            `LOF to toc'. Reported at begin document}%
       \fi
      \fi
      \if@dotoclot
       \ifEtoc@lot
        \etocsetup{locallottotoc}%
        \PackageInfo{etoc}{%
            Setting (or re-setting) `locallottotoc=true' as the\MessageBreak
            package tocbibind was detected and is configured for\MessageBreak
            `LOT to toc'. Reported at begin document}%
       \fi
      \fi
   }% end of tocbibind branch
   {}%
  }% end of at begin document
}% end of not with memoir branch
```

LaTeX 2021 fall release lets \contentsline always grab four arguments, so with 1.09e 2021/09/23 we make sure our \addtocontents will always provide \contentsline with four arguments. This extra {} is done without checking LaTeX's version by laziness, as an impact on documents compiled with former LaTeX could be visible only with very special contexts that only the author himself would ever consider.

Let's also add \protected@file@percent at 1.09e although this is a priori of no relevance as **etoc** reads the toc file with \endlinechar=-1 regime.

When using \addcontentsline nothing needs to be done as both things are handled by LaTeX upstream. 1.2 executes the \ifEtoc@hyperref test inside the fourth argument, rather than using one \addtocontents in each branch.

```
\def\Etoc@addtocontents #1#2{%
    \addtocontents {toc}{%
      \protect\contentsline{#1}{#2}{\thepage}{\ifEtoc@hyperref\@currentHref\fi}%
      \ifdefined\protected@file@percent\protected@file@percent\fi
    }%
}
\def\Etoc@addcontentsline@ #1#2#3{%
    \@namedef{toclevel@#1}{#3}\addcontentsline {toc}{#1}{#2}%
}
\DeclareRobustCommand*{\etoctoccontentsline}
    {\@ifstar{\Etoc@addcontentsline@}{\Etoc@addtocontents}}
\def\Etoc@addtocontents@immediately#1#2{%
    \begingroup
       \let\Etoc@originalwrite\write
       \def\write{\immediate\Etoc@originalwrite}%
       \Etoc@addtocontents{#1}{#2}%
```

```
    \endgroup
}
\def\Etoc@addcontentsline@@immediately#1#2#3{%
    \begingroup
       \let\Etoc@originalwrite\write
       \def\write{\immediate\Etoc@originalwrite}%
       \Etoc@addcontentsline@{#1}{#2}{#3}%
    \endgoroup
}
\DeclareRobustCommand*{\etocimmediatetoccontentsline}
    {\@ifstar{\Etoc@addcontentsline@@immediately}{\Etoc@addtocontents@immediately}}
```

Formerly a LaTeX counter etoc@tocdepth was used but at 1.2 it has been replaced by macro-storage.

```
\def\Etoc@storetocdepth    {\xdef\Etoc@savedtocdepth{\number\c@tocdepth}}
\def\Etoc@restoretocdepth {\global\c@tocdepth\Etoc@savedtocdepth\relax}
\def\etocobeytoctocdepth {\def\etoc@settocdepth
                          {\afterassignment\Etoc@@nottoodeep \global\c@tocdepth}}
\def\Etoc@@nottoodeep {\ifnum\Etoc@savedtocdepth<\c@tocdepth
                          \global\c@tocdepth\Etoc@savedtocdepth\relax\fi }
\def\etocignoretoctocdepth {\let\etoc@settocdepth\@gobble }
\def\etocsettocdepth   {\futurelet\Etoc@nexttoken\Etoc@set@tocdepth }
\def\Etoc@set@tocdepth {\ifx\Etoc@nexttoken\bgroup
                           \expandafter\Etoc@set@tocdepth@
                        \else\expandafter\Etoc@set@toctocdepth
                        \fi }
\def\Etoc@set@tocdepth@ #1{\@ifundefined {Etoc@#1@@}
      {\PackageWarning{etoc}
          {Unknown sectioning unit #1, \protect\etocsettocdepth\space ignored}}
      {\global\c@tocdepth\csname Etoc@#1@@\endcsname}%
}
\def\Etoc@set@toctocdepth  #1#{\Etoc@set@toctocdepth@ }
\def\Etoc@set@toctocdepth@ #1{%
  \@ifundefined{Etoc@#1@@}%
   {\PackageWarning{etoc}
        {Unknown sectioning depth #1, \protect\etocsettocdepth.toc ignored}}%
   {\addtocontents {toc}
   {\protect\etoc@settocdepth\expandafter\protect\csname Etoc@#1@@\endcsname}}%
}
```

placeholder

```
\def\etocimmediatesettocdepth #1#{\Etoc@set@toctocdepth@immediately}
\def\Etoc@set@toctocdepth@immediately #1{%
  \@ifundefined{Etoc@#1@@}%
   {\PackageWarning{etoc}
        {Unknown sectioning depth #1, \protect\etocimmediatesettocdepth.toc ignored}}%
   {\begingroup
      \let\Etoc@originalwrite\write
      \def\write{\immediate\Etoc@originalwrite}%
      \addtocontents {toc}
      {\protect\etoc@settocdepth\expandafter\protect
       \csname Etoc@#1@@\endcsname}%
    \endgroup
   }%
}
```

placeholder

```
\def\etocdepthtag   #1#{\Etoc@depthtag }
\def\Etoc@depthtag  #1{\addtocontents {toc}{\protect\etoc@depthtag {#1}}}
```

1.09f adds \etocimmediatedepthtag.toc. This can serve in some circumstances, see user documentation. Apologies for long delay to Norman Ramsey who reported problem and his fix in July... 2016!

```
\def\etocimmediatedepthtag   #1#{\Etoc@depthtag@immediately }
\def\Etoc@depthtag@immediately #1{%
  \begingroup
    \let\Etoc@originalwrite\write
    \def\write{\immediate\Etoc@originalwrite}%
    \addtocontents {toc}{\protect\etoc@depthtag {#1}}%
  \endgroup
}
\def\etocignoredepthtags {\let\etoc@depthtag \@gobble }
\def\etocobeydepthtags   {\let\etoc@depthtag \Etoc@depthtag@ }
\def\Etoc@depthtag@ #1{\@ifundefined{Etoc@depthof@#1}%
       {}% ignore in silence if tag has no associated depth
       {\afterassignment\Etoc@@nottoodeep
                  \global\c@tocdepth\csname Etoc@depthof@#1\endcsname}%
}
\def\etocsettagdepth #1#2{\@ifundefined{Etoc@#2@@}%
       {\PackageWarning{etoc}
         {Unknown sectioning depth #2, \protect\etocsettagdepth\space ignored}}%
       {\@namedef{Etoc@depthof@#1}{\@nameuse{Etoc@#2@@}}}%
}
```

We must cancel all tocvsec2 toc-related actions. But a check must be done for the memoir class, as its tocvsec2 emulation does not have the incompatible things etoc needs to revert.

```
\def\Etoc@tocvsec@err #1{\PackageError {etoc}
        {The command \protect#1\space is incompatible with `etoc'}
        {Use \protect\etocsettocdepth.toc as replacement}%
}%
\AtBeginDocument {%
  \@ifclassloaded{memoir}
    {\PackageInfo {etoc}
     {Regarding `memoir' class command \protect\settocdepth, consider\MessageBreak
     \protect\etocsettocdepth.toc as a drop-in replacement with more\MessageBreak
     capabilities (see `etoc' manual). \space
     Also, \protect\etocsettocdepth\MessageBreak
     and \protect\etocsetnexttocdepth\space should be used in place of\MessageBreak
     `memoir' command \protect\maxtocdepth\@gobble}%
    }%
    {\@ifpackageloaded {tocvsec2}{%
      \def\maxtocdepth  #1{\Etoc@tocvsec@err \maxtocdepth   }%
      \def\settocdepth  #1{\Etoc@tocvsec@err \settocdepth   }%
      \def\resettocdepth  {\@ifstar {\Etoc@tocvsec@err \resettocdepth }%
                                    {\Etoc@tocvsec@err \resettocdepth }%
                          }%
```

If etoc is added to a LaTeX document using already tocvsec2.

```
      \def\save@tocdepth #1#2#3{}%
      \let\reset@tocdepth\relax
      \let\remax@tocdepth\relax
      \let\tableofcontents\etoctableofcontents
      \PackageWarningNoLine {etoc}
      {Package `tocvsec2' detected and its modification of\MessageBreak
       \protect\tableofcontents\space reverted. \space Use
       \protect\etocsettocdepth.toc\MessageBreak as a replacement
       for `tocvsec2' toc-related commands}%
     }% tocvsec2 loaded
     {}% tocvsec2 not loaded
```

```
    }%
}%
```

placeholder

```
\def\invisibletableofcontents {\etocsetnexttocdepth {-3}\tableofcontents }%
\def\invisiblelocaltableofcontents
                          {\etocsetnexttocdepth {-3}\localtableofcontents }%
\def\etocsetnexttocdepth #1{%
    \@ifundefined{Etoc@#1@@}
     {\PackageWarning{etoc}
       {Unknown sectioning unit #1, \protect\etocsetnextocdepth\space ignored}}
     {\Etoc@setnexttocdepth{\csname Etoc@#1@@\endcsname}}%
}%
\def\Etoc@setnexttocdepth#1{%
    \def\Etoc@tocdepthset{%
        \Etoc@tocdepthreset
        \edef\Etoc@tocdepthreset {%
               \global\c@tocdepth\the\c@tocdepth\space
               \global\let\noexpand\Etoc@tocdepthreset\noexpand\@empty
         }%
        \global\c@tocdepth#1%
        \global\let\Etoc@tocdepthset\@empty
    }%
}%
\let\Etoc@tocdepthreset\@empty
\let\Etoc@tocdepthset  \@empty
\def\etocsetlocaltop #1#{\Etoc@set@localtop}%
\def\Etoc@set@localtop #1{%
  \@ifundefined{Etoc@#1@@}%
   {\PackageWarning{etoc}
        {Unknown sectioning depth #1, \protect\etocsetlocaltop.toc ignored}}%
   {\addtocontents {toc}
   {\protect\etoc@setlocaltop\expandafter\protect\csname Etoc@#1@@\endcsname}}%
}%
```

placeholder

```
\def\etocimmediatesetlocaltop #1#{\Etoc@set@localtop@immediately}%
\def\Etoc@set@localtop@immediately #1{%
  \@ifundefined{Etoc@#1@@}%
   {\PackageWarning{etoc}
        {Unknown sectioning depth #1, \protect\etocimmediatesetlocaltop.toc ignored}}%
   {\begingroup
       \let\Etoc@originalwrite\write
       \def\write{\immediate\Etoc@originalwrite}%
       \addtocontents {toc}
           {\protect\etoc@setlocaltop\expandafter\protect
                                      \csname Etoc@#1@@\endcsname}%
    \endgroup
   }%
}%
```

1.09i would like to rename this to `\Etoc@setlocaltop`, for consistency with internal macros, but too late it is already in user `.toc` files.

```
\def\etoc@setlocaltop #1{%
  \ifnum#1=\Etoc@maxlevel
   \Etoc@skipthisonetrue
  \else
   \Etoc@skipthisonefalse
```

MEMO: the \Etoc@level thus continues receiving updates from various \etoc@setlocaltop present in the .toc file even once the local toc is already done, but this has no importance, as the code in \Etoc@toctoc after executing the code contents is not influenced by this but only by the final status of \Etoc@stackofends remains the same. And this gets updated via \Etoc@setlocaltop@doendsandbegin (see below).

I should check if not worthwile to move the \ifEtoc@stoptoc test earlier.

There should be two notions of local top. One for a potential barrier, stopping a toc, the other for setting a local top. These two should be distinct. Or at least additional to this one which does both. But then I would have to document. And test. And implement first.

```
    \global\let\Etoc@level #1%
    \global\let\Etoc@virtualtop #1%
    \ifEtoc@localtoc
     \ifEtoc@stoptoc
      \Etoc@skipthisonetrue
     \else
      \ifEtoc@notactive
       \Etoc@skipthisonetrue
      \else
       \unless\ifnum\Etoc@level>\etoclocaltop
        \Etoc@skipthisonetrue
        \global\Etoc@stoptoctrue
       \fi
      \fi
     \fi
    \fi
   \fi
   \let\Etoc@next\@empty
   \ifEtoc@skipthisone
   \else
    \ifnum\Etoc@level>\c@tocdepth
    \else
     \ifEtoc@standardlines
     \else
```

So here the \etocsetlocaltop.toc will cause various starts and finish parts to get executed, even for the somewhat fictitious levels. But this may cause collaterals, in particular we have to be careful about the \ifEtoc@skipprefix Boolean, which may be set to \iftrue in the {⟨*start*⟩} part of the level style and thus needs to be reset to \iffalse. This is done automatically in \Etoc@etoccontentsline@ but here we are not executing it so we need to add somewhere a \global\Etoc@skipprefixfalse (else we may impact rendering of subsequent level). So we put it together with \Etoc@doendsandbegin in a wrapper. This wrapper is also there to avoid a problem when the TOC is checked for emptiness, as we need then to be able to tell \etoc@setlocaltop to not execute anything.

This also stresses that the name of the macro is a bit of a misnomer, yes it serves to delimit local table of contents, but really it is implemented as a ghost of a sectioning unit which does have an impact (on the global TOC or local TOCs from encompassing levels), as it triggers when encountered the {⟨*finish*⟩} portions of previous finer levels (and the {⟨*finish*⟩} code of its own level will be executed sooner or later), and the {⟨*start*⟩} code of subsequent finer levels (as well as its own {⟨*start*⟩} code at least once, depending on how levels are nested).

```
     \let\Etoc@next\Etoc@setlocaltop@doendsandbegin
    \fi
   \fi
  \fi
  \Etoc@next
}%
```

It is important to reset the \ifEtoc@skipprefix boolean, as is done in \Etoc@etoccontentsline@.

```
\def\Etoc@setlocaltop@doendsandbegin{%
    \Etoc@doendsandbegin
    \global\Etoc@skipprefixfalse
```

```
}
\addtocontents {toc}{\protect\@ifundefined{etoctocstyle}%
            {\let\protect\etoc@startlocaltoc\protect\@gobble
             \let\protect\etoc@settocdepth\protect\@gobble
             \let\protect\etoc@depthtag\protect\@gobble
             \let\protect\etoc@setlocaltop\protect\@gobble}{}}%
```

Initializations.

```
\def\etocstandardlines {\Etoc@standardlinestrue}
\def\etoctoclines      {\Etoc@standardlinesfalse}
\etocdefaultlines
\etocstandardlines
\def\etocstandarddisplaystyle{%
    \PackageWarningNoLine{etoc}{%
      \string\etocstandarddisplaystyle \on@line\MessageBreak
      is deprecated. \space Please use \string\etocclasstocstyle}%
}
\expandafter\def\expandafter\etocclasstocstyle\expandafter{%
    \etocclasstocstyle
    \Etoc@classstyletrue
}
\def\etocetoclocaltocstyle{\Etoc@etocstyletrue}
\def\etocusertocstyle{\Etoc@etocstylefalse}
\etocclasstocstyle
\etocetoclocaltocstyle
\etocobeytoctocdepth
\etocobeydepthtags
\let\etocbeforetitlehook   \@empty
\let\etocaftertitlehook    \@empty
\let\etocaftercontentshook \@empty
\let\etocaftertochook      \@empty
```

`listings` abuses `\tableofcontents` for its `\lstlistoflistings`. It doesn't seem worth to let my version of `\tableofcontents` have to check for this special circumstance. So at `1.09d`, simply add this (the boolean was added at `1.2`):

```
\def\etockeeporiginaltableofcontents
   {\Etoc@keeporiginaltoctrue\let\tableofcontents\etocoriginaltableofcontents}%
\endinput
```