

An Extension of the L^AT_EX-Theorem Environment*

Wolfgang May[†]
Institut für Informatik,
Universität Freiburg
Campus am Flughafen
D-79110 Freiburg
Federal Republic of Germany

Andreas Schlechte[‡]
Marienburger Weg 6
D-38678 Clausthal-Zellerfeld
Federal Republic of Germany

1999/12/26

Abstract

`ntheorem.sty` is a package for handling theorem-like environments. Additionally to several features for defining the layout of theorem-like environments which can be regarded to be standard requirements for a theorem-package, it provides solutions for two related problems: placement of endmarks and generation of lists of theorem-like environments.

In contrast to former approaches, it solves the problem of setting endmarks of theorem-like environments (theorems, definitions, examples, and proofs) *automatically* at the right positions, even if the environment ends with a `displaymath` or (even nested) list environments, it also copes with the `amsmath` package. This is done in the same manner as the handling of labels by using the `.aux` file.

It also introduces the generation of lists of theorem-like environments in the same manner as `listoffigures`. Additionally, more comfortable referencing is supported.

After running L^AT_EX several times (depending on the complexity of references, in general, three runs are sufficient), the endmarks are set correctly, and theoremlists are generated.

Since `ntheorem.sty` uses the standard L^AT_EX `\newtheorem` command, existing documents can be switched to `ntheorem.sty` without having to change the `.tex` file. Also, it is compatible with L^AT_EX files using `theorem.sty` written by Frank Mittelbach.

*This file has version number 1.18, last revised 1999/12/26.

[†]`may@informatik.uni-freiburg.de`

[‡]`Andreas.Schlechte@tu-clausthal.de`

Contents

1	Introduction	3
2	The User-Interface	3
2.1	How to include the package	3
2.2	Defining New Theorem Sets	4
2.3	Defining the Layout of Theorem Sets	4
2.3.1	Common Parameters for all Theorem Sets	5
2.3.2	Parameters for Individual Sets	5
2.3.3	Font Selection	6
2.3.4	Predefined theorem styles	6
2.3.5	Default Setting	6
2.3.6	A Standard Set of Theorems	7
2.3.7	Customization and Local Settings	7
2.4	Generating Theorem Lists	7
2.4.1	Defining the List Layout	8
2.4.2	Writing Extra Stuff to Theorem File	8
2.5	For Experts: Defining Layout Styles	9
2.5.1	Defining New Theorem Layouts	9
2.5.2	Defining New Theorem List Layouts	9
2.6	Setting End Marks	10
2.7	Extended Referencing Features	10
2.8	Miscellaneous	11
3	Possible Interferences	11
3.1	Interfering Document Options.	11
3.2	Combination with amslatex.	11
3.2.1	amsmath	11
3.2.2	amsthm	11
3.3	Babel	12
3.4	Hyperref	12
4	Examples	12
4.1	Extended Referencing Features	17
4.2	List of Theorems and Friends	18
5	The End Mark Algorithm	20
5.1	The Idea	20
5.2	The Realization	20
6	Problems and Questions	21
6.1	Known Limitations	21
6.2	Known “Bugs” and Problems	23
6.3	Open Questions	23

7	Code Documentation	23
7.1	Documentation of the Macros	23
7.1.1	Thmmarks-Related Stuff	23
7.1.2	Option leqno to Thmmarks	29
7.1.3	Option fleqn to Thmmarks	30
7.1.4	Option amsmath to Thmmarks	31
7.1.5	Theorem-Layout Stuff	35
7.1.6	Theorem-Environment Handling Stuff	40
7.1.7	Extended Referencing Facilities	47
7.1.8	Generation of Theorem Lists	48
7.1.9	Auxiliary macros	55
7.1.10	Other Things	55
7.2	The Standard Configuration	56
8	History and Acknowledgements	57
8.1	The endmark-Story (Wolfgang May)	57
8.2	Lists, Lists, Lists (Andreas Schlechte)	57
8.3	Let's come together	57
8.4	Acknowledgements	58

1 Introduction

For our purposes here, “theorems” are labelled enunciations, often set off from the main text by extra space and a font change. Theorems, corollaries, conjectures, definitions, examples, remarks, and proofs are all instances of “theorems”. The “header” of these structures is composed of the type of the structure (such as THEOREM or REMARK), a number which serializes the instances of the same type throughout the document, and an optional name (such as “Correctness Theorem”). The layout of theorems can be changed by parameters as the fonts of the header and the body, the way how to arrange the headers, the indentation, and the way of numbering it. Confronted with these requirements, `theorem.sty`, a style for dealing with theorem layout was developed by Frank Mittelbach which was the standard theorem-environment for long time.

But then the desire for additional features like “endmarks” and “theorem-lists” arose. Two extensions of `theorem.sty` were developed: One for handling endmarks, `thmmarks.sty` and one for generating lists, `newthm.sty`. Thus, Frank Mittelbach suggested to combine the new features into one “standard-to-be” package. And now, here it is.

2 The User-Interface

2.1 How to include the package

The package `ntheorem.sty` is included by

```
\usepackage[options]{ntheorem.sty},
```

where the optional parameter `<options>` selects predefined configurations and special requirements.

The following `<options>` are available by now, concerning three independent issues:

Predefining environments: (see Section 2.3.6) With [standard] and [noconfig], it can be chosen, if and what file is used for activating a (user-defined) standard set of theorem environments.

Compatibility with amsthm: option [amsthm] provides compatibility with the theorem-layout commands of the `amsthm`-package (see Section 3.2).

Activation of endmarks: [thmmarks] enables the automatical placement of endmarks (see 2.3); when using the `amsmath`-package, [thmmarks] must be complemented by [amsmath] (see Section 3.2).

Activation of extended reference features: [thref] enables the extended reference features (see Section 4.1).

Compatibility with hyperref: option [hyperref] provides compatibility with the `hyperref`-package (see section 3.4).

2.2 Defining New Theorem Sets

`\newtheorem` The syntax and semantics is exactly the same as in standard L^AT_EX: the command `\newtheorem` defines a new “theorem set” or “theorem-like structure”. Two required arguments name the new environment set and give the text to be typeset with each instance of the new “set”, while an optional argument determines how the “set” is enumerated:

`\newtheorem{foo}{bar}` The theorem set `foo` (whose name is `bar`) uses its own counter.

`\newtheorem{foo2}[foo]{bar2}` The theorem set `foo2` (printed name `bar2`) uses the same counter as the theorem set `foo`.

`\newtheorem{foo3}{bar}[section]` The theorem set `foo3` (printed name `bar`) is enumerated within the counter `section`, i.e. with every new `\section` the enumeration begins again with 1, and the enumeration is composed from the section-number and the theorem counter itself.

For every environment $\langle name \rangle$ defined by `\newtheorem`, two environments $\langle name \rangle$ and $\langle name* \rangle$ are defined. In the main document, they have exactly the same effect, but the latter causes no entry in the respective list of theorems (cf. `\section` and `\section*`), see also Section 2.4.

`\renewtheorem` Theorem sets can be redefined by `\renewtheorem`, with the same arguments as explained for `\newtheorem`. When redefining a theorem set, the counter is not re-initialized.

2.3 Defining the Layout of Theorem Sets

For theorem-like environments, the user can set parameters by setting several switches and then calling `\newtheorem`. The layout of a theorem set is defined with the values of the switches at the time `\newtheorem` is called.

2.3.1 Common Parameters for all Theorem Sets

`\theorempreskipamount` These additional parameters affect the vertical space around theorem environments:
`\theorempostskipamount` `\theorempreskipamount` and `\theorempostskipamount` define, respectively, the spacing before and after such an environment. These parameters apply for all theorem sets and can be manipulated with the ordinary length macros. They are rubber lengths, (‘skips’), and therefore can contain `plus` and `minus` parts.

2.3.2 Parameters for Individual Sets

The layout of individual theorem sets can be further determined by switches controlling the appearance of the headers and the header-body-layout:

- `\theoremstyle` • `\theoremstyle{<style>}`: The general structure of the theorem layout is defined via its `\theoremstyle`. `\ntheorem` provides several predefined styles including those of Frank Mittelbach’s `theorem.sty` (cf. Section 2.3.4. Additional styles can be defined by `\newtheoremstyle` (cf. Section 2.5.1).
- `\theoremheaderfont` • `\theoremheaderfont{<fontcmds>}`: The theorem header is set in the font specified by `<fontcmds>`.
 In contrast to `theorem.sty`, `\theoremheaderfont` can be set individually for each environment type.
- `\theorembodyfont` • `\theorembodyfont{<fontcmds>}`: The theorem body is set in the font specified by `<fontcmds>`.
- `\theoremseparator` • `\theoremseparator{<thing>}`: `<thing>` separates the header from the body of the theorem-environment. E.g., `<thing>` can be “:” or “.”.
- `\theoremindent` • `\theoremindent{<dimen>}` can be used to indent the theorem wrt. the surrounding text.
 ! It’s a ‘(dimen)’, so the user shouldn’t try to specify a `plus` or `minus` part, cause this leads to an error.
- `\theoremnumbering` • `\theoremnumbering{<style>}` specifies the appearance of the numbering of the theorem set. Possible `<styles>` are `arabic` (default), `alph`, `Alph`, `roman`, `Roman`, `greek`, `Greek`, and `fnsymbol`.
 Clearly, if a theorem-environment uses the counter of another environment type, also the numbering style of that environment is used.
- `\theoremsymbol` • `\theoremsymbol{<thing>}`: This is only active if `ntheorem.sty` is loaded with option `[thmmarks]`. `<thing>` is set as an endmark at the end of every instance of the environment. If no symbol should appear, say `\theoremsymbol{}`.

The flexibility provided by these command should relieve the users from the ugly hacking in `\newtheorem` to fit most of the requirements stated by publishers or supervisors.

`\theoremclass` With the command `\theoremclass{<theorem-type>}` (where `<theorem-type>` must be an already defined theorem type), these parameters can be set to the values which were used when `\newtheorem` was called for `<theorem-type>`.
 With `\theoremclass{LaTeX}`, the standard L^AT_EX layout can be chosen.

2.3.3 Font Selection

From the document structuring point of view, theorem environments are regarded as special parts inside a document. Furthermore, the theorem header is only a distinguished part of a theorem environment. Thus, `\theoremheaderfont` inherits characteristics of `\theorembodyfont` which also inherits in characteristics of the font of the surrounding environment. Thus, if for example `\theorembodyfont` is `\itshape` and `\theoremheaderfont` is `\bfseries` the font selected for the header will have the characteristics ‘bold extended italic’. If this is not desired, the corresponding property has to be explicitly overwritten in `\theoremheaderfont`, e.g. by `\theoremheaderfont{\normalfont\bfseries}`

2.3.4 Predefined theorem styles

The following theorem styles are predefined, covering those from `theorem.sty`:

<code>plain</code>	This theorem style emulates the original \LaTeX definition, except that additionally the parameters <code>\theorem...skipamount</code> are used.
<code>break</code>	In this style, the theorem header is followed by a line break.
<code>change</code>	Header number and text are interchanged, without a line break.
<code>changebreak</code>	Like <code>change</code> , but with a line break after the header.
<code>margin</code>	The number is set in the left margin, without a line break.
<code>marginbreak</code>	Like <code>margin</code> , but with a line break after the header.
<code>nonumberplain</code>	Like <code>plain</code> , without number (e.g. for proofs).
<code>nonumberbreak</code>	Like <code>break</code> , without number.
<code>empty</code>	No number, no name. Only the optional argument is typeset.

2.3.5 Default Setting

If no option is given, i.e. `ntheorem.sty` is loaded by `\usepackage{ntheorem.sty}`, the following default is set up:

```
\theoremstyle{plain},
\theoremheaderfont{\normalfont\bfseries} and
\theorembodyfont{\itshape},
\theoremseparator{,}
\theoremindent0cm,
\theoremnumbering{arabic},
\theoremsymbol{}
```

Thus, by only saying `\newtheorem{...}{...}`, the user gets the same layout as in standard \LaTeX .

2.3.6 A Standard Set of Theorems

A standard configuration of theorem sets is provided within the file `ntheorem.std`, which will be included by the option `[standard]`. It uses the `amssymb` and `latexsym` (automatically loaded) packages and defines the following sets:

Theorems: `Theorem`, `Lemma`, `Proposition`, `Corollary`, `Satz`, `Korollar`,

Definitions: `Definition`,

Examples: `Example`, `Beispiel`,

Remarks: `Anmerkung`, `Bemerkung`, `Remark`,

Proofs: `Proof` and `Beweis`.

These theorem sets seem to be the most frequently used environments in english and german documents.

The layout is defined to be `theoremstyle plain`, `bodyfont \itshape`, `Headerfont \bfseries`, and `endmark (theoremsymbol) \ensuremath{_{\Box}}` for all theorem-like environments¹. For the definition-, remark- and example-like sets, the above setting is used, except `bodyfont \upshape`. The proof-like sets are handled a bit differently. There, the layout is defined as `theoremstyle nonumberplain`, `bodyfont \upshape`, `headerfont \scshape` and `endmark \ensuremath{_{\blacksquare}}`. For a more detailed information look at `ntheorem.std` or at the code-section.

2.3.7 Customization and Local Settings

Since the user should not change `ntheorem.std`, we've added the possibility to use an own configuration-file. If one places the file `ntheorem.cfg` in the path searched by \TeX , this file is read automatically (if `[standard]` is not given). The usage of `ntheorem.cfg` can be prevented by the `[noconfig]` option. Thus, just a copy of `ntheorem.std` to `ntheorem.cfg` must be made which then can freely be modified by the user. Note, that if a configuration-file exists, this will always be used (I.e. with option `standard` and an existing configuration-file, the `.cfg` file will be used and the `.std` file won't).

2.4 Generating Theoremlists

`\listtheorems` Similar to the \LaTeX command `\listoffigures`, any theorem set defined with a `\newtheorem` statement may be listed at any place in your document by

```
\listtheorems{<list>}
```

The argument `<list>` is a comma-separated list of the theorem sets to be listed. For a theorem set `<name>`, only the instances are listed which are instantiated by `\begin{<name>}`. Those instantiated by `\begin{<name>*}` are omitted (cf. `\section` and `\section*`).

For example, `\listtheorems{Corollary, Lemma}` leads to a list of all instances of one of the theorem sets “Corollary” or “Lemma”. Note, that the set name given to the command is the first argument which is specified by `\newtheorem` which is also the one to be used in `\begin{theorem} ... \end{theorem}`.

¹Note, that `mathmode` is ensured for the symbol.

If `\listtheorems` is called for a set name which is not defined via `\newtheorem`, the user is informed that a list is generated, but there will be no typeset output at all.

2.4.1 Defining the List Layout

`\theoremlisttype` Theoremlists can be formatted in different ways. Analogous to theorem layout, there are several predefined types which can be selected by

`\theoremlisttype{<type>}`

The following four *<type>*s are available (for examples, the user is referred to section 4).

all List any theorem of the specified set by number, (optional) name and pagenumber. This one is also the default value.

allname Like **all**, additionally with leading theoremname.

opt Analogous to **all**, but only the theorems which have an optional name are listed.

optname Like **opt**, with leading theoremname.

2.4.2 Writing Extra Stuff to Theorem File

Similar to `\addcontentsline` and `\addtocontents`, additional entries to theoremlists are supported. Since entries to theoremlists are a bit more intricate than entries to the lists maintained by standard L^AT_EX `\addcontentsline` and `\addtocontents` cannot be used in a straightforward way².

`\addtheoremline` Analogous to `\addcontentsline`, an extra entry for a theorem list can be made by

`\addtheoremline{<name>}{<text>}`

where *<name>* is the name of a valid theorem set and *<text>* is the text, which should appear in the list. For example,

`\addtheoremline{Example}{Extra Entry with number}`

generates an entry with the following characteristics:

- The Label of the theorem “Example” is used.
- The current value of the counter for “Example” is used
- The current pagenumber is used.
- The specified text is the optional text for the theorem.

Thus, the above command has the same effect as it would be for

`\begin{Example}[Extra Entry with number] \end{Example}`

except, that there would be no output of the theorem, and the counter isn’t advanced.

`\addtheoremline*` Alternatively you can use

²for a theorem, its number has to be stored explicitly since different theorem sets can use the same counter. Also, it is optional to reset the counter for each section.

`\addtheoremline*{Example}{Extra Entry}`

which is the same as above, except that the entry appears without number.

`\addtotheoremfile` Sometimes, e.g. for long lists, special control sequences (e.g. a pagebreak) or additional text should be inserted into a list. This is done by

`\addtotheoremfile[⟨name⟩]{⟨text⟩}`

where $\langle name \rangle$ is the name of a theorem set and $\langle text \rangle$ is the text to be written into the theorem file. If the optional argument $\langle name \rangle$ is omitted, the given text is inserted in every list, otherwise it is only inserted for the given theorem set.

2.5 For Experts: Defining Layout Styles

2.5.1 Defining New Theorem Layouts

`\newtheoremstyle` Additional layout styles for theorems can be defined by

`\newtheoremstyle{⟨name⟩}{⟨head⟩}{⟨opt-head⟩}`.

After this, `\theoremstyle{⟨name⟩}` is a valid `\theoremstyle`. Here, $\langle head \rangle$ has to be a statement using two arguments, `##1`, containing the keyword, and `##2`, containing the number. $\langle opt-head \rangle$ has to be a statement using three arguments where the additional argument `##3` contains the optional parameter.

Since L^AT_EX implements theorem-like environments by `\trivlists`, both header declarations must be of the form `\item[... \theorem@headerfont ...]`..., where the dotted parts can be formulated by the user. If there are some statements producing output after the `\item[...]`, you have to care about implicit spaces.

Because of the `@`, if `\newtheoremstyle` is used in a `.tex` file, it has to be put between `\makeatletter` and `\makeatother`.

For details, look at the code documentation or the definitions of the predefined theoremstyles.

`\renewtheoremstyle` Theorem styles can be redefined by `\renewtheoremstyle`, with the same arguments as explained for `\newtheoremstyle`.

2.5.2 Defining New Theorem List Layouts

`\newtheoremlisttype` Analogous, additional layouts for theorem lists can be defined by

`\newtheoremlisttype{⟨name⟩}{⟨start⟩}{⟨line⟩}{⟨end⟩}`.

The first argument, $\langle name \rangle$, is the name of the listtype, which can be used as a valid `\theoremlisttype`. $\langle start \rangle$ is the sequence of commands to be executed at the very beginning of the list. Corresponding, $\langle end \rangle$ will be executed at the end of the list. These two are set to do nothing in the standard-types. $\langle line \rangle$ is the part to be called for every entry of the list. It has to be a statement using four arguments: `##1` will be replaced with the name of the theorem, `##2` with the number, `##3` with the theorem's optional text and `##4` with the pagenumber.

WARNING: Self-defined Layouts will break with the `hyperref`-package.

`\renewtheoremlisttype` Theorem list types can be redefined by `\renewtheoremlisttype`, with the same arguments as explained for `\newtheoremlisttype`.

2.6 Setting End Marks

The automatic placement of endmarks is activated by calling `ntheorem.sty` with the option `[thmmarks]`. Since then, the endmarks are set automatically, there are only a few commands for dealing with very special situations.

`\qed`
`\qedsymbol` If in a single environment, the user wants to replace the standard endmark by some other, this can be done by saying `\qed`, if `\qedsymbol` has been defined by `\qedsymbol{<something>}` (in option standard, `\qedsymbol` is defined to be the symbol used for proofs, since a potential use of this features is to close trivial corollaries without explicitly proving them).

Additionally, if in a single environment of a theorem set, that is defined without an endmark, the user wants to set an endmark, this is done with `\qedsymbol` and `\qed` as described above. `\qedsymbol` can be redefined everywhere in the document.

`\NoEndMark`
`\TheoremSymbol` On the other hand, if in some situation, the user decides to set the endmark manually (e.g. inside a figure or a minipage), the automatic handling can be turned off by `\NoEndMark` for the current environment. Then – assumed that the current environment is of type `<name>`, the endmark can manually be set by just saying `\<name>Symbol`.

Note that there must be no empty line in the input before the `\end{theorem}`, since then, the end mark is ignored (cf. Theorem 3 in Section 4).

2.7 Extended Referencing Features

The extended referencing features are activated by calling `ntheorem.sty` with the option `[thref]`.

Note that the option has to be explicitly given since version 1.13.

Often, when writing a paper, one changes propositions into theorems, theorems into corollaries, lemmata into remarks and so on. Then, it is necessary to adjust also the references, i.e., from “see Proposition~\ref{completeness}” to “see Theorem~\ref{completeness}”. For relieving the user from this burden, the type of the respective labeled entities can be associated with the label itself:

```
\label{<label>}[<type>]
```

associates the type `<type>` with `<label>`.

This task is automated for theorem-like environments:

```
\begin{Theorem}[<name>]\label{<label>}
```

is equivalent to

```
\begin{Theorem}[<name>]\label{<label>}[Theorem]
```

`\thref` The additional information is used by

```
\thref{<label>}
```

which outputs the respective environment-type *and* the number, e.g., “Theorem 42”. Note that L^AT_EX has to be run twice after changing labels (similar to getting references OK; in the intermediate run, warnings about undefined reference types can occur).

The `[thref]` option interferes with the `babel` package, thus in this case, `ntheorem` has to be loaded *after* `babel`.

2.8 Miscellaneous

Inside a theorem-like environment $\langle env \rangle$, the name given as optional argument is accessible by $\langle env \rangle name$.

3 Possible Interferences

Since `ntheorem` reimplements the handling of theorem-environments completely, it is incompatible with every package also concerning those macros.

Additionally, the `thmmarks` algorithm for placing endmarks requires modifications of several environments (cf. Section 7). Thus, environments which are reimplemented or additionally defined by document options or styles are not covered by the endmark algorithm of `ntheorem.sty`.

The `[thref]` option changes the `\label` command and the treatment of labels when reading the `.aux` file. Thus it is potentially incompatible with all packages also changing `\label` (or `\newlabel`). Compatibility with `babel`'s `\newlabel` is achieved if `babel` is loaded before `ntheorem`.

3.1 Interfering Document Options.

`ntheorem.sty` also copes with the usual document options `leqno` and `fleqn`³. If one of those options is used in the `\documentclass` declaration, it is automatically recognized by the `thmmarks` part of `ntheorem.sty`.

If one of those options is not used in `\documentclass`, but with `amsmath` (see next section), it must not be specified for `ntheorem`, since all `amsmath` environments detect this option by themselves.

3.2 Combination with `amslatex`.

`ntheorem.sty` interferes with `amsmath.sty` and `amsthm.sty`.

Note, that the LaTeX `amstex` package `amstex.sty` (L^AT_EX_{2.09}) is obsolete and you should use `amsmath` and `amstext` for L^AT_EX_{2 ϵ} instead.

We would be happy if someone knowing and using `amsmath` would join the development and maintenance of this style.

3.2.1 `amsmath`

The math environments of `amsmath` are adapted in the option `[amsmath]`, (i.e., if `\usepackage{amsmath}` is used then `\usepackage[thmmarks]{ntheorem.sty}` must be completed to `\usepackage[amsmath,thmmarks]{ntheorem}`). Note, that `amsmath` has to be loaded *before* `ntheorem` since the definitions have to be overwritten.

3.2.2 `amsthm`

`amsthm.sty` conflicts with the definition of theorem layouts in `theorem.sty`, some features of `amsthm.sty` have been incorporated into option `[amsthm]` which has to be used *instead of* `\usepackage{amsthm}`.

³although for `fleqn` and long formulas reaching to the right margin, equation numbers and endmarks can be smashed over the formula since `fleqn` does not use `\eqno` for controlling the setting of the equation number.

The Option provides theoremstyles `plain`, `definition`, and `remark`, and a `proof` environment as in `amsthm.sty`.

The `\newtheorem*` command is defined even without this option. Note that `\newtheorem*` always switches to the nonnumbered version of the current theoremstyle which thus must be defined.

The command `\newtheoremstyle` is not taken over from `amsthm.sty`. Also, `\swapnumbers` is not implemented. Here, the user has to express his definitions by the `\newtheoremstyle` command provided by `ntheorem.sty`, including the use of `\theoremheaderfont` and `\theorembodyfont`. The options `[amsthm]` and `[standard]` are in conflict since they both define an environment `proof`.

Thus, we recommend not to use `amsthm`, since the features for defining theorem-like environments in `ntheorem.sty`—following `theorem.sty`—seem to be more intuitive and user-friendly.

3.3 Babel

The `[thref]` option interferes with the `babel` package, thus in case that `babel` is used, `ntheorem` has to be loaded *after* `babel`.

3.4 Hyperref

Since `hyperref` redefines the L^AT_EX `\contentsline`-command, it breaks with `ntheorem` below version 1.17. Since version 1.17, the option `[hyperref]` makes `ntheorem` work with `hyperref`. Theorem lists will then get linked list.

WARNING: The definition and redefinition of Theorem List Layouts (see Section 2.5.2) isn't yet working with the `hyperref`-package.

4 Examples

The setting is as follows.

- For Theorems:

```
\theoremstyle{marginbreak}
\theoremheaderfont{\normalfont\bfseries}\theorembodyfont{\slshape}
\theoremsymbol{\ensuremath{\diamondsuit}}
\theoremseparator{:}
\newtheorem{Theorem}{Theorem}
```

- For Lemmas:

```
\theoremstyle{changebreak}
\theoremsymbol{\ensuremath{\heartsuit}}
\theoremindent0.5cm
\theoremnumbering{greek}
\newtheorem{Lemma}{Lemma}
```

- For Corollaries:

```
\theoremindent0cm
\theoremsymbol{\ensuremath{\spadesuit}}
\theoremnumbering{arabic}
\newtheorem{Corollary}[Theorem]{Corollary}
```

- For Examples:

```
\theoremstyle{change}
\theorembodyfont{\upshape}
\theoremsymbol{\ensuremath{\ast}}
\theoremseparator{}
\newtheorem{Example}{Example}
```

- For Definitions:

```
\theoremstyle{plain}
\theoremsymbol{\ensuremath{\clubsuit}}
\theoremseparator{.}
\newtheorem{Definition}{Definition}
```

- For Proofs:

```
\theoremheaderfont{\sc}\theorembodyfont{\upshape}
\theoremstyle{nonumberplain}
\theoremseparator{}
\theoremsymbol{\rule{1ex}{1ex}}
\newtheorem{Proof}{Proof}
```

Note, that parts of the setting are inherited. For instance, the fonts are not reset before defining “Lemma”, so the font setting of “Theorem” is used.

1 Example (Simple one) The first example is just a text.

In the next examples, it is shown how an endmark is put at a displaymath, a single equation and both types of eqnarrays. *

1 Theorem (Long Theorem):

The examples are put into this theorem environment.

The next example will not appear in the list of examples since it is written as

```
\begin{Example*} ... \end{Example*}
```

2 Example (Ending with a displayed formula) Look, the endmark is really at the bottom of the line:

$$f^{(n)}(z) = \frac{n!}{2\pi i} \int_{\partial D} \frac{f(\zeta)}{(\zeta - z)^{n+1}} d\zeta \quad *$$

At this point, we add an additional entry without number in the Example list:

```
\addtheoremline*{Example}{Extra Entry}
```

α Lemma (Display with array):

Lemmata are indented and numbered with greek symbols. Also for displayed arrays of this form, it looks good:

```
\[\begin{array}{l}
a = \begin{array}{t}{l}
first\ line \\
second\ line
\end{array}%
\mbox{try to put this text in the lowest line}\end{array}\]
```


Definition 1 (With a list).

$$\int_{\gamma} f(z) dz := \int_a^b f(\gamma(t))\gamma'(t) dt \quad (4)$$

- you've seen, how it works for text and
- math environments,
- and it works for lists. ♣

2 Corollary (Q.E.D.):

And here is a trivial corollary, which is ended by `\qedsymbol{\text{q.e.d}}` and `\qed`. *q.e.d*

3 Example

$$f^{(n)}(z) = \frac{n!}{2\pi i} \int_{\partial D} \frac{f(\zeta)}{(\zeta - z)^{n+1}} d\zeta$$

If there is some text after an environment, the endmark is put after the text. *

The next one is done by the following sequence. Note, that `~\hfill~` is inserted to prevent L^AT_EX from using its nested list management (a verbatim is also a trivlist), i.e. this causes L^AT_EX to start the `verbatim`-Part in a new line.

```
\begin{Example}
~\hfill~
\begin{verbatim}
And, it also works for verbatim
... when the \end{verbatim} is in the
same line as the text ends. \end{verbatim}
~ this space is important !!
\end{Example}
```

4 Example (Using verbatim)

```
And, it also works for verbatim
... when the \end{verbatim} is in the
same line as the text ends. *
```

There must be no empty line in the input before the `\end{theorem}` (since then, the end mark is ignored)

```
\begin{Theorem}
some text ... but no end mark

\end{Theorem}
```

3 Theorem:

some text ... but no end mark

Now, there is a corollary which should appear with a different name in the list of corollaries:

```

\begin{Corollary*}[title in text]\label{otherlabel}
...
\end{Corollary*} \addtheoremline{Corollary}{title in list}

```

4 Corollary (title in text):

*It also works in the
center
environment.*



5 Theorem (Quote):

In quote environments, the text is normally indented from left and right by the same space. The endmark is not indented from the right margin, i.e., it is typeset to the right margin of the surrounding text.



Here is an example for turning off the endmark automatics and manual handling:

```

\begin{Theorem}[Manual End Mark]\label{somelabel}
a line of text with a manually set endmark \hfill\TheoremSymbol \
some more text, but no automatic endmark set. \NoEndMark
\end{Theorem}

```

6 Theorem (Manual End Mark):

*a line of text with a manually set endmark
some more text, but no automatic endmark set.*



Also, one should note, that `\hfill` is inserted to set the endmark at the right margin.

5 Example (Quickie) It also works for short one's.



If you are tired of the greek numbers and the indentation for lemmata ... you can redefine it:

```

\theoremstyle{changebreak}
\theoremheaderfont{\normalfont\bfseries}\theorembodyfont{\slshape}
\theoremsymbol{\ensuremath{\heartsuit}}
\theoremsymbol{\ensuremath{\diamondsuit}}
\theoremseparator{:}
\theoremindent0.5cm
\theoremnumbering{arabic}
\renewtheorem{Lemma}{Lemma}

```

4 Lemma:

another lemma, with arabic numbering ... note that the numbering continues.



the optional argument (i.e. the 'theorem'-name) can be accessed by `\(env)name`.

```

\begin{Theorem}[somename]
Obviously, we are in Theorem~\Theoremname.
\end{Theorem}

```

7 Theorem (somename):

Obviously, we are in Theorem somename.



This feature can e.g. be used for automatically generating executable code and a commented solution sheet:

```

\begin{exercise}[quicksort]
  <the exercise text>
\begin{verbatimwrite}{solutions/\exercisename.c}
  <C-code>
\end{verbatimwrite}
\verbatiminput{solutions/\exercisename.c}
\end{exercise}

```

This will write the C-code to a file `solutions/quicksort.c` and type it also on the solution sheet.

Now, we define an environment `KappaTheorem` which uses the same style parameters as `Theorem`s and is numbered together with `Corollaries` (`Theorem`s are also numbered with `Corollaries`). Note that we define a complex header text and a complex end mark.

```

\theoremclass{Theorem}
\theoremsymbol{\ensuremath{a\atop b}}
\newtheorem{KappaTheorem}[Corollary]{\(\kappa\)-Theorem}

```

8 κ -Theorem (1st κ -Theorem):

That's the first Kappa-Theorem.

a
b

4.1 Extended Referencing Features

The standard `\label` command is extended by an optional argument which is intended to contain the “name” of the structure which is labeled, allowing more comfortable referencing; e.g., this section has been started with

```

\subsection*{Extended Referencing Features}%
\label{sec-ExtRef}[Section]

```

As already stated, for theorem-like environments the optional argument is filled in automatically, i.e.,

```

\begin{Theorem}[Manual End Mark]\label{somelabel}

```

(cf. page 16) is equivalent to

```

\begin{Theorem}[Manual End Mark]\label{somelabel}[Theorem]

```

`\thref{<label>}` additionally outputs the contents of the optional argument which has been associated with `<label>`:

```

This is \thref{sec-ExtRef}
A theorem end mark has been set manually in \thref{somelabel}.
A center environment has been shown in \thref{otherlabel}.
The first Kappa-Theorem has been given in \thref{kappatheorem1}.

```

generates

This is Section 4.1.

A theorem end mark has been set manually in Theorem 6. A center environment has been shown in Corollary 4. The first Kappa-Theorem has been given in κ -Theorem 8.

Here one must be careful that the handling of the optional argument is automated only for environments defined by `\newtheorem`, i.e., *not* for sectioning, equations, or enumerations.

Calling `\thref{<label>}` for a label which has been set without an optional argument can result in different unintended results: If *<label>* is not inside a theorem-like environment, an error message is obtained, otherwise the type of the surrounding theorem-like environment is output, e.g., calling `\thref{label}` then results in “Theorem *<number>*”! Additionally, currently there is no support for multiple references such as “see Theorems 5 and 7” (this would require plural-forms for different languages and handling of `\ref`-lists, probably splitting into different sublists for different environments)⁴.

4.2 List of Theorems and Friends

Note, that we put the following lists into the `quote`-environment to emphasise them from the surrounding text. So the lists are indented slightly at the margin.

With

```
\addtotheoremfile{Added into all theorem lists},
```

in every list, an additional line of text would be inserted. But it isn't actually done in this documentation since we want to use different list formats.

Only for the list of Examples, this one is added:

```
\addtotheoremfile[Example]{Only concerning Example lists}
```

With

```
\theoremlisttype{all}  
\listtheorems{Lemma},
```

all lemmas are listed:

α	Display with array	13
β	Equation	14
γ	Breakstyle	14
4	16
5	22
6	22

From the examples, only those are listed which have an optional name:

```
\theoremlisttype{opt}  
\listtheorems{Example}
```

⁴If someone is interested in programming this, please contact us; it seems to be algorithmically easy, but tedious.

leads to

0	Extra Entry with number	8
	Extra Entry	9
1	Simple one	13
	Extra Entry	13
4	Using <code>verbatim</code>	15
5	Quickie	16

Only concerning Example lists

One should note the line *Only concerning example lists*, which was added by the `\addtotheoremfile`-statement above.

For the next list, another layout, using the `tabular`-environment, is defined:

```
\newtheoremlisttype{tab}%
{\begin{tabular*}{\linewidth}{@{}lr1{\extracolsep{\fill}}r@{}}%
{##1&##2&##3&##4\}%
{\end{tabular*}}}
```

Thus, by saying

```
\theoremlisttype{tab}
\listtheorems{Theorem, Lemma},
```

theorems and lemmata are listed:

Theorem	1	Long Theorem	13
Lemma	α	Display with array	13
Lemma	β	Equation	14
Lemma	γ	Breakstyle	14
Theorem	3		15
Theorem	5	Quote	16
Theorem	6	Manual End Mark	16
Lemma	4		16
Theorem	7	somename	16
Theorem	9	Correctness	21
Theorem	10	Completeness	21
Lemma	5		22
Lemma	6		22
Theorem	11		23

L^AT_EX-lists can also be used to format the theoremlist. The input

```
\newtheoremlisttype{list}%
{\begin{trivlist}\item}
{\item[##2 ##1:] \ ##3\dotfill ##4}%
{\end{trivlist}}
\theoremlisttype{list}
\listtheorems{Corollary}
```

leads to

2	Corollary: Q.E.D.	15
4	Corollary: title in list	16

In this example, after the item, `_` is used instead of `_`, because in the latter case, `\dotfill` will produce an error if the optional argument (`##3`) is missing.

5 The End Mark Algorithm

5.1 The Idea

The handling of endmarks with `thmmarks.sty` is based on the same two-pass principle as the handling of labels: the necessary information about endmarks is contained in the `.aux` file.

With `thmmarks.sty`, \TeX is always aware whether it is in some theorem-like environment. There, potential positions for endmarks can be

1. at the end of simple text lines in open text,
2. at the end of `displaymaths`,
3. at the end of equations or `equationarrays`, or
4. at the end of text lines at the end of lists (or, more general, `trivlists`, such as `verbatim` or `center`).

The problem is, that in the cases (2)–(4), the endmark has to be placed in a box which is already shipped out, when `\end{...}` is processed. Thus, in those situations, \TeX needs to know from the `.aux` file, whether it has to put an endmark. When \TeX is in a theorem-like environment and comes to one of the points mentioned in (2)–(4), and the `.aux` file says that there is an endmark, then it is put there. Anyway, it maintains a counter of the potential positions of an end mark in the current theorem-like environment. When it comes to an `\end{theorem}`, it looks if it is in situation (1) (then the endmark is simply put at the end of the current line). Otherwise, the last horizontal box is already shipped out (thus it contains a situation (2)–(4)) and the endmark must be set in it. In this case, a note is written in the `.aux` file, where the endmark actually has to be set (ie, at the latest potential point for setting an endmark inside the theorem).

5.2 The Realization

Let $\langle env \rangle$ be a theorem-like environment. Then, additional to the counter $\langle env \rangle$, \TeX maintains two counters `curr $\langle env \rangle$ ctr` and `end $\langle env \rangle$ ctr`. In the i th environment of type $\langle env \rangle$, `curr $\langle env \rangle$ ctr` = i (the \LaTeX counter $\langle env \rangle$ cannot be used since a) environments can use the counter of other environments, and b) often counters are reinitialized inside a document). `end $\langle env \rangle$ ctr` counts the potential situations for putting an endmark inside an environment. It is set to 1 when starting an environment. Each time, when a situation (2)–(4) is reached, the command

$$\backslash\text{mark}\langle\text{roman}\{\text{currenvctr}\}\rangle\langle env \rangle\langle\text{roman}\{\text{end}\langle env \rangle\text{ctr}\}\rangle$$

is called (`\text{roman}\{\text{curr}\langle env \rangle\text{ctr}\}\rangle\langle env \rangle\langle\text{roman}\{\text{end}\langle env \rangle\text{ctr}\}\rangle` uniquely identifies all situations (2)–(4) in a document).

If at this position an endmark has to be set,

$$\backslash\text{mark}\langle\text{roman}\{\text{curr}\langle env \rangle\text{ctr}\}\rangle\langle env \rangle\langle\text{roman}\{\text{end}\langle env \rangle\text{ctr}\}\rangle$$

is defined in the `.aux` file to be `\end $\langle env \rangle$ Symbol`, otherwise it is undefined and simply ignored.

When \TeX comes to an `\end{\langle env \rangle}`, it looks if it is in situation (1). If so, the endmark is simply put at the end of the current line. Otherwise,

```
\def\mark<\roman{currentctr}><env>%
<\roman{end<env>ctr}>{\<env>Symbol}
```

is written to the `.aux` file for setting the endmark at the latest potential position inside the theorem in the next run.

9 Theorem (Correctness):

1. For a `.tex` file, which does not contain nested theorem-like environments of the same type, in the above situation, the following holds: When compiling, at the i th situation in the j th environment of type `<env>`, `\mark j <env>` i is handled.

For `.tex` files which contain nested theorem-like environments of the same type, `\mark k <env> l` is handled, where k is the number of the latest environment of type `<env>` which has been called at this moment, and l is the number of situations (2)–(4) which have occurred in environments of type `<env>` since the k th `\begin{<env>}`.

2. When finishing an environment, either an endmark is set directly (when in a text line) or an order to put the end symbol at the latest potential position is written to the `.aux` file. \diamond

10 Theorem (Completeness):

The handling of endmarks is complete wrt. `plain text`, `displaymath`, `equation`, `eqnarray`, `eqnarray*`, and all environments ended by `endtrivlist`, including `center` and `verbatim`. \diamond

So, where can be bugs ?

- in the plain `TEX` handling of endmarks,
- in some special situations which have not been tested yet,
- in some special environments which have not been tested yet.
- in the `amsmath` environments. We seldom use them, so we do not know their pitfalls, and we ran only general test cases.

6 Problems and Questions

6.1 Known Limitations

- Since `ntheorem.sty` uses the `.aux` file for storing information about the positions of endmarks, `LATEX` must be run twice for correctly setting the endmarks.
- Since `ntheorem.sty` uses the `.aux` file for storing information about lists in the `.thm` file, a minimum of two runs is needed. If theorems move in any of these runs up to five runs can be needed to generate correct lists.
- Since we need to expand the optional argument of theorems in various ways for the lists, we decided to copy the text verbatim into the `.thm` file. Thus, if you use things like `\thesection` etc., the list won't show the correct text. Therefore you shouldn't use any command that needs to be expanded.

- In nested environments ending at the same time, only the endmark for the inner environment is set, as the following example shows:

```
\begin{Lemma}
Some text.
\begin{Proof} The Proof \end{Proof}
\end{Lemma}
```

yields to

5 Lemma:

Some text.

PROOF The Proof ■

You can handle this by specifying something invisible after the end of the inner theorem. Then the endmark for the outer theorem is set in the next line:

```
\begin{Lemma}
Some text.
\begin{Proof} The Proof \end{Proof}~
\end{Lemma}
```

yields to

6 Lemma:

Some text.

PROOF The Proof ■

◇

- Document option `fleqn` is problematic: `fleqn` handles equations not by `$$` but by lists (check what happens for

```
\begin{theorem} \[ displaymath \] \end{theorem}
```

in standard L^AT_EX: The `displaymath` is *not* set in an own line). Also, for long formulas, the equation number and the endmark are smashed into the formula at the right text margin.

- Naturally, `ntheorem.sty` will not work correctly in combination with other styles which change the handling of
 1. theorem-like environments, or
 2. environments concerned with the handling of endmarks, e.g. `\[...\]`, `eqnarray`, etc.

- `ntheorem.sty` is compatible with Frank Mittelbach's `theorem.sty`, which is the most widespread style for setting theorems.

It cannot be used *with* `theorem.sty`, but it can be used instead of it.

6.2 Known “Bugs” and Problems

- Ending a theorem *directly* after the text, e.g.

```
\begin{Theorem} text\end{Theorem}
```

suppresses the endmark:

11 Theorem:

text

Therefore a space or a newline should be inserted before `\end{...}`.

- With `theoremstyle break`, if the linebreak would cause ugly linebreaking in the following text, it is suppressed.

6.3 Open Questions

- For `equations`, we decided to put the endmark after the equation number, which is vertically centered. Currently, we do not know, how to get the equation number centered and the endmark at the bottom (one has to know the internal height of the math material).
- The placement of endmarks is mainly based on a check whether \LaTeX is in an ordinary text line when encountering an end-of-environment. This question is *partially* answered by `\ifhmode`: In a text line, \LaTeX is always in `\hmode`. But, after an `displaymath`, \LaTeX is also in `\hmode`. Thus, additionally `\lastskip` is checked: after a `displaymath`, `\lastskip=0` holds. In most situations, when text has been written into a line, `\lastskip ≠ 0`. But, this does not hold, if the source code is of the following form: `...text\label{bla}`: then, `\lastskip=0`. In those situations, the endmark is suppressed.
?? How can it be detected whether \LaTeX has just ended a `displaymath`?
- The above problem with the label: The break style enforces a linebreak by `\hfill\penalty-8000` after the `\trivlist-item`. Thus, \TeX gets back into the horizontal mode. The label places a “whatsit” somewhere ... and, it seems that the “whatsit” makes \TeX think that there is a line of text.

If someone has a solution to one of those questions, please inform us. (You can be sure to be mentioned in the Acknowledgements.)

7 Code Documentation

7.1 Documentation of the Macros

```
1 \typeout{Style ‘\basename’, Version \fileversion\space <\filedate>}
2 \ProvidesPackage{ntheorem}[\filedate \space\fileversion]
3 \newif\if@thmmarks\@thmmarksfalse
4 \newif\ifthm@tempif
```

general setup.

7.1.1 Thmmarks-Related Stuff

`\endequation` For equations, end marks are placed behind the equation number:

```
25 \gdef\SetMark@endeqn{\quad}% as default, cf. option leqno
26 \gdef\endequation{\eqno \hbox{\@eqnnum \PotEndMark{\SetMark@endeqn}}%
27   $$\global\@ignoretrue}
```

Line 25: As default, work for equation numbers at the right: Then, a `\quad` is placed between equation number and endmark.

Line 26: In addition to the equation number (set by `\@eqnnum` at the right of the line) `\SetMark@endeqn` is carried out.

`\[` If an end mark is set, a `displaymath` is put into box such that the end marks appears at its bottom level at the right. Thus, also the definition of `\[` has to be changed:

```
28 \gdef\[{%
29   \relax\ifmmode
30     \@badmath
31   \else
32     \ifvmode
33       \nointerlineskip
34       \makebox[.6\linewidth]%
35     \fi
36     $$\stepcounter{end\InTheoType ctr}%
37     \@ifundefined{mark\roman{curr\InTheoType ctr}}{\relax}%
38       \InTheoType\roman{end\InTheoType ctr}}{\relax}%
39     {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
40       \boxmaxdepth=.5ex\begin{array}[b]{1}%
41       \boxmaxdepth=\maxdimen\displaystyle\fi}%
42     \addtocounter{end\InTheoType ctr}{-1}%
43     %$$$ BRACE MATCH HACK
44   \fi}
```

Lines 29–35, 43, 44: the old definition.

Lines 36–39: The end position of a `displaymath` inside a theorem-environment corresponds to `end\InTheoType ctr+1`. An endmark has to be set there, if `\mark<\roman{curr#1ctr}>#1 <\roman{end#1ctr}+1 >` is defined and not the empty symbol.

Lines 40–41: If so, the whole displayed stuff is put in an array with maximal depth `0.5ex` and vertically adjusted with its bottom line (then, the endmarks will appear adjusted to its bottom line).

Line 42: The counter has to be re-decremented.

`\]` At the end of a `displaymath`, the end marks is set at its bottom level:

```
45 \gdef\]%{%
46   \stepcounter{end\InTheoType ctr}%
47   \@ifundefined{mark\roman{curr\InTheoType ctr}}{\relax}%
48     \InTheoType\roman{end\InTheoType ctr}}{\relax}%
49   {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
50     \end{array}\fi}%
51   \addtocounter{end\InTheoType ctr}{-1}%
52   \relax\ifmmode
53     \ifinner
54     \@badmath
55   \else
56     \PotEndMark{\eqno}\global\@ignoretrue$$$$ BRACE MATCH HACK
```

```

57     \fi
58   \else
59     \@badmath
60   \fi
61   \ignorespaces}

```

Lines 46–51: Look, if an endmark has to be set in this displaymath (analogous to lines 36–42 of `\def\[]`) If so, there is an inner array which has to be closed (line 50).

Lines 52–61: the old definition.

Line 56: changed to set an endmark at the right of the line if necessary (this is done by `\eqno`).

`\endeqnarray` For `\eqnarrays`, the end marks is set below the number of the last equation:

```

62 \gdef\SetMark@endeqnarray#1{\llap{\raisebox{-1.3em}{#1}}}
63 \gdef\endeqnarray{%
64   \global\let\Oldeqnnum=\@eqnnum
65   \gdef\@eqnnum{\Oldeqnnum\PotEndMark{\SetMark@endeqnarray}}%
66   \@eqnrcr
67   \egroup
68   \global\advance\c@equation\m@ne
69   $$\global\@ignoretrue
70   \global\let\@eqnnum\Oldeqnnum}

```

Line 62: As default work for equation numbers at the right: Then, the endmark is placed below the last equation number at the right margin.

New: Lines 64, 65, 70:

Line 64: save `\@eqnnum`.

Line 65: define `\@eqnnum` to carry out `\Oldeqnnum`, then a potential endmark position is handled: if an endmark is set, between the equation number and the endmark, the command sequence `\SetMark@endeqnarray` is carried out – there, since `\SetMark@endeqnarray` is a function of one argument, the endmark will be this argument.

Lines 66–69: from `latex.ltx`. Line 66 sets the equation number.

Line 70: restore `\@eqnnum`.

`\endeqnarray*` In an `\eqnarray*`, the end mark is set at the right of the last equation:

```

71 \@namedef{endeqnarray*}{%
72   %   from \@eqnrcr:
73   \let\reserved@a\relax
74   \ifcase\@eqcnt \def\reserved@a{& & } \or \def\reserved@a{& &}%
75   \or \def\reserved@a{&} \else
76     \let\reserved@a\empty
77     \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
78   \reserved@a {\normalfont \normalcolor \PotEndMark{}}%
79   \global\@eqnswtrue\global\@eqcnt\z@\cr
80   %
81   \egroup
82   \global\advance\c@equation\m@ne
83   $$\global\@ignoretrue}

```

This is just L^AT_EX's `\endeqnarray` where lines 73–79 are inserted from `\@eqnrcr` and augmented (line 78) to set a potential endmark (with no additional commands)

at the end of the current line.

Changes to Tabbing Environment

Original: lttab.dtx

`\endtabbing` Here, the `\endtrivlist` modification is not sufficient: \LaTeX is not in hmode when it calls `\endtrivlist` from `\endtabbing`; additionally, `\@stopline` already outputs a linebreak. Thus, the end mark is inserted *before* `\@stopline` at the right margin (using `\'`).

```
84 \gdef\endtabbing{%
85   \PotEndMark{\'}\@stopline\ifnum\@tabpush >\z@ \@badpoptabs
86   \fi\endtrivlist}
```

Changes to Center Environment

Original: ltmiscen.dtx

`\endcenter` In \LaTeX , `\endcenter` just calls `\endtrivlist`. Here, the situation is more complex since the the endmark has to be put in the last line without affecting its centering: if in a text line (only then, here is a potential endmark position):

```
87 \gdef\endcenter{%
88   \@endtrivlist
89   {\PotEndMark{\rightskip0pt%
90     \settowidth{\leftskip}%
91     {\csname mark\roman{curr}\InTheoType ctr}\InTheoType
92       \roman{end}\InTheoType ctr}\endcsname}%
93   \advance\leftskip\@flushglue\hskip\@flushglue}}
```

The `\rightskip` of the line is set to 0, `\leftskip` is set to the width of one space (since on the right, one space is added after the text) plus the endmark and infinitely stretchable glue (`\@flushglue`), and also the line is continued with `\@flushglue` (the actual position is one space after the text), and then the endmark is placed (by `\PotEndMark`).

Handling of Endmarks

`\@endtheorem` `\@endtheorem` is called for every `\end{<env>}`, where `<env>` is a theorem-like environment. `\@endtheorem` is extended to organize the placement of the corresponding end mark (`\InTheoType` gives the innermost theorem-like environment, i.e. the one to be ended):

```
94 \gdef\@empty{}
95 \gdef\@endtheorem{%
96   \expandafter
97   \ifx\csname\InTheoType Symbol\endcsname\@empty\setendmarkfalse\fi
98   \@endtrivlist
99   {\ifsetendmark
100     \hbox{ }\nobreak\hfill\nobreak\csname\InTheoType Symbol\endcsname
101     \setendmarkfalse \fi}%
102   \ifsetendmark\OrganizeTheoremSymbol\else\global\setendmarktrue\fi}
```

Lines 96, 97: if the end symbol of the environment `<env>` to be closed is empty, simply no end symbol has to be set (it makes a difference, if no end symbol is set, or if an empty end symbol is set).

Lines 98, 102: (originally, it calls `\endtrivlist`):

Lines 98, 100, 101: `\@endtrivlist` is called to put $\langle env \rangle$ Symbol at the end of the line and set `setendmark` to false if \TeX is in a text line and `setendmark` is true. At this point, `setendmark` is false iff the user has disabled it locally or the end symbol is empty.

Line 99: the endmark is not set, if `setendmark` is false.

Line 102: if `setendmark` is true, the correct placement of the end symbol is organized, else (ie either `setendmarkfalse` is set by the user, or the endmark is already set by `\@endtrivlist`) reset `setendmark` to true.

For further comments see `\@endtrivlist` and `\OrganizeTheoremSymbol`.

The construction in line 100 guarantees that the endmark is put at the end of the line, even if it is the only letter in this line.

`\NoEndMark` By `\NoEndMark`, the automatical setting of an end mark is blocked for the *current* environment.

```
103 \gdef\NoEndMark{\global\setendmarkfalse}
```

set `setendmark` to false. It is automatically reset to `true` after the end of the current environment.

`\qed` With `\qed`, the user can locally change the end symbol to appear:

```
104 \gdef\qed{\expandafter\def\csname \InTheoType Symbol\endcsname
105           {\the\qedsymbol}}%
```

When calling `\qed`, the end symbol of the innermost theorem-like environment at that time is set to the value stored in `\qedsymbol` at that time.

`\PotEndMark` Handling a potential endmark position:

```
106 \gdef\PotEndMark#1{\SetEndMark{\InTheoType}{#1}}%
```

Argument: $\langle cmd_seq \rangle := \#1$ is a command sequence to be executed when setting the endmark.

It adds the current theorem type $\langle env \rangle$ to the parameters, and calls

```
\PotEndMark{\langle env \rangle}{\langle cmd\_seq \rangle}.
```

`\SetEndMark` `\SetEndMark` sets an endmark for an environment. It is called by `\PotEndMark`.

```
107 \gdef\SetEndMark#1#2{%
108   \stepcounter{end#1ctr}%
109   \@ifundefined{mark\roman{curr#1ctr}#1\roman{end#1ctr}}{%
110     {\relax}%
111     #2{\csname mark\roman{curr#1ctr}#1\roman{end#1ctr}\endcsname
112       \hskip-\rightmargin\hbox to 0cm{~}}}%
```

Arguments:

$\langle env \rangle := \#1$: current theorem-environment.

$\langle cmd_seq \rangle := \#2$: is a command sequence to be executed when setting the endmark.

Both arguments are transmitted from by `\PotEndMark`.

Line 108: increments `end\langle env \rangle ctr` for preparing the next situation for setting a potential endmark.

Line 109, 110: if

```
\mark<\roman{curr<env>ctr}><env><\roman{end<env>ctr}>
```

is undefined – which is the case iff at this position no endmark has to be set –, nothing is done,

Line 111: otherwise, `<cmd_seq>` and then

```
\mark<\roman{curr<env>ctr}>\env <\roman{end<env>ctr}>,
```

which is defined in the `.aux` file to be the end symbol are called.

The construction `<cmd_seq>{...}` in line 111 allows the handling of the end symbol as an argument of `<cmd_seq>` as needed for `\endeqnarray`.

Line 112: By `\hskip-\rightmargin\hbox to 0cm{~}`, a negative hspace of amount `\rightmargin` is added *after* the end symbol – thus, the symbol is set as there were no right margin (this concerns, e.g., `\quote` environments).

Writing to `.aux` file. (copied from `\def\label (ltxref.dtx)`)

```
113 \newskip\mysavskip
114 \gdef\@bbsphack{%
115     \mysavskip\lastskip
116     \unskip}
117 %
118 \gdef\@eesphack{%
119     \ifdim\mysavskip>\z@
120     \vskip\mysavskip \else\fi}
```

Lines 114–116 and 117–119 are similar to `\@bbsphack` and `\@eesphack` of `latex.ltx`. They undo resp. redo the last `skip`.

`\OrganizeTheoremSymbol` The information for setting the end marks is written to the `.aux` file:

```
121 \gdef\OrganizeTheoremSymbol{%
122     \@bbsphack
123     \edef\thm@tmp{\expandafter\expandafter\expandafter\thm@meaning
124         \expandafter\meaning\csname\InTheoType Symbol\endcsname\relax}%
125     \protected@write\@auxout{}%
126         {\string\global\string\def\string\mark%
127             \roman{curr\InTheoType ctr}\InTheoType \roman{end\InTheoType ctr}%
128             {\thm@tmp}}%
129     \@eesphack}
```

Lines 125–127: Write

```
\global\def\mark<\roman{curr<env>ctr}><env><\roman{end<env>ctr}>>
{<env>Symbol} to the .aux file.
<env>:=\InTheoType gives the innermost theorem-like environment, i.e. the one
the end symbol has to be set for.
```

```
130 } % end of option [thmmarks]
```

7.1.2 Option `leqno` to `Thmmarks`

```
131 \DeclareOption{leqno}{%
132     \if@thmmarks
133     \PackageInfo{basename}{Option ‘leqno’ loaded}%
134     \gdef\SetMark@endeqn#1{\hss\llap{#1}}
135     \gdef\SetMark@endeqnarray#1{\hss\llap{#1}}
```

136 `\fi}%`
`leqno` is only active if `thmmarks` is also active.

Line 134, 135: Since with `leqno`, the equation number is placed on the left, after infinitely stretchable glue, the endmark can be set straight at the right margin.

7.1.3 Option `fleqn` to `Thmmarks`

```
137 \DeclareOption{fleqn}{%
138 \if@thmmarks
139 \PackageInfo{\basename}{Option 'fleqn' loaded}%
```

`fleqn` is only active if `thmmarks` is also active.

\[Since `fleqn` treats displayed math as trivlists, it's quite another thing:

```
140 \renewcommand\[{ \relax
141 \ifmode\@badmath
142 \else
143 \begin{trivlist}%
144 \@beginparpenalty\predisplaypenalty
145 \@endparpenalty\postdisplaypenalty
146 \item[]\leavevmode
147 \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
148 \hskip\mathindent\bgroup
149 \stepcounter{end\InTheoType ctr}%
150 \@ifundefined{mark\roman{curr\InTheoType ctr}}%
151 \InTheoType\roman{end\InTheoType ctr}}{\relax}%
152 {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
153 \boxmaxdepth=.5ex\begin{array}[b]{1}%
154 \boxmaxdepth=\maxdimen\displaystyle\fi}%
155 \addtocounter{end\InTheoType ctr}{-1}%
156 \fi}
```

Lines 140–148, 156: the old definition.

Line 149–155: if an endmark has to be set in this displaymath, it is put into an array with depth $\leq 0.5ex$, and vertically adjusted to the bottom line.

\] Here, the end mark is placed after a `\hfil` at the end of the line containing the displaymath:

```
157 \renewcommand\]{%
158 \stepcounter{end\InTheoType ctr}%
159 \@ifundefined{mark\roman{curr\InTheoType ctr}}%
160 \InTheoType\roman{end\InTheoType ctr}}{\relax}%
161 {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
162 \end{array}\fi}%
163 \addtocounter{end\InTheoType ctr}{-1}%
164 \relax\ifmode
165 \egroup $\hfil\PotEndMark{ }$
166 \egroup
167 \end{trivlist}%
168 \else \@badmath
169 \fi}
```

Lines 158–162: Look, if an endmark has to be set in this displaymath. If so, close the inner array.

Lines 164–169: the old definition.

Line 165: Added `\PotEndMark`.

`\endequation` for equations, the end mark is also set with the equation number:

```
170 \gdef\endequation{%
171     $\hfil % $
172     \displaywidth\linewidth\hbox{\@eqnnum \PotEndMark{\SetMark@endeqn}}%
173     \egroup
174     \endtrivlist}
```

Line 172: When the equation number is set, also the endmark is set with the same trick as for `\endequation` without `fleqn`.

`\endeqnarray` When the equation number is set, also the endmark is set with the same trick as for `\endeqnarray` without `fleqn` (see Lines 176, 177, 182):

```
175 \gdef\endeqnarray{%
176     \global\let\Oldeqnnum=\@eqnnum
177     \gdef\@eqnnum{\Oldeqnnum\PotEndMark{\SetMark@endeqnarray}}%
178     \@eqncr
179     \egroup
180     \global\advance\c@equation\m@ne$$$ $$$
181     \global\@ignoretrue
182     \global\let\@eqnnum\Oldeqnnum}
```

```
183 \fi}% end of option fleqn
```

7.1.4 Option `amsmath` to `Thmmarks`

Taken from `amsmath.sty`.

```
184 \DeclareOption{amsmath}{%
185 \if@thmmarks
186 \PackageInfo{\basename}{Option ‘amsmath’ loaded}%
    amsmath is only active if thmmarks is also active.
187 \newdimen\thm@amstmpdepth
```

A temporarily used register.

`\TagsPlusEndmarks` Since `amsmath` uses “tags” for setting end marks, some macros are defined which prepare tags which include endmarks:

```
188 \gdef\TagsPlusEndmarks{%
189     \global\let\Old@maketag@@=\maketag@@@
190     \global\let\Old@df@tag=\df@tag
191     \if@eqnsw\SetTagPlusEndMark\else
192     \iftag@\SetTagPlusEndMark
193     \else\SetOnlyEndMark
194     \fi\fi}
```

Lines 189, 190: store the original macros.

Line 191: if equation numbers are set as default, call `\SetTagPlusEndMark` to set tag and end mark.

Line 192: if a tag is set manually, call `\SetTagPlusEndMark` to set tag and end mark.

Line 193: otherwise, call `\SetOnlyEndMark` to set only an end mark.

```

195 \gdef\SetOnlyEndMark{%
196     \global\tag@true
197     \iftagsleft@
198         \gdef\df@tag{\hbox
199             to \displaywidth{\hss\PotEndMark{\maketag@@}}}%
200     \else
201         \gdef\df@tag{\PotEndMark{\maketag@@}}%
202     \fi}

```

Set only an end mark:

Line 196: force setting the end mark as a tag:

Lines 198,199: if tags are set to the left, the tag consists of a `\hbox` over the whole displaywidth, with the (potential) endmark at its right.

Line 201: if tags are set to the right, the tag consists only of the (potential) endmark.

```

203 \gdef\SetTagPlusEndMark{%
204     \iftagsleft@
205         \gdef\maketag@@@##1{%
206             \hbox to \displaywidth{\m@th\normalfont##1%
207                 \PotEndMark{\hss}}}%
208     \else
209         \gdef\maketag@@@##1{\hbox{\m@th\normalfont##1%
210             \llap{\hss\PotEndMark{\raisebox{-1.3em}}}}}%
211     \fi}

```

Set a tag *and* an end mark:

Lines 204–211: redefine the `\maketag@@@` macro:

Lines 205–207: if tags are set to the left, build a box of the whole displaywidth and put the original tag on the left, and the (potential) endmark at the right.

Lines 209,210: if the tags are set to the right, the (potential) end mark is put below it.

```

212 \gdef\RestoreTags{%
213     \global\let\maketag@@@=\Old@maketag@@@
214     \global\let\df@tag=\Old@df@tag}

```

Lines 213,214: restore the original macros.

`\endgather` In the `gather` environment, just the augmented tag is used:

```

215 \gdef\endgather{%
216     \TagsPlusEndmarks % <<<<<<<<<
217     \math@cr
218     \black@totwidth@
219     \egroup
220     $$%
221     \RestoreTags % <<<<<<<<<
222     \global\@ignoretrue}
223 %
224 \xandafter\let\csname endgather*\endcsname\endgather

```

New:

Line 216: the last tag contains the potential endmark.

Line 221: restore the original macros.

Line 224: Since `let` always takes the expansion of a macro when the `let` is executed, all `let`'s have to be adjusted (this is the same for all subsequent `let`-statements).

`\math@cr@@@align`

`\endalign` `\endalign` also uses the augmented tags:

```
225 \def\endalign{%
226     \ifingather@ \else          % <<<<<<<<<
227     \TagsPlusEndmarks\fi % <<<<<<<<<
228     \math@cr
229     \black@ \totwidth@
230     \egroup
231     \ifingather@
232     \restorealignstate@
233     \egroup
234     \nonumber
235     \ifnum0='{ \fi}%
236     \else
237     $$%
238     \RestoreTags          % <<<<<<<<<
239     \fi
240     \global\@ignoretrue}
```

New:

Lines 226, 227: if the `align` is not inside another environment, its tags have to contain the endmarks.

Line 238: this case, the original macros have to be restored.

```
241 \expandafter\let\csname endalign*\endcsname\endalign
242 \let\endxalignat\endalign
243 \expandafter\let\csname endxalignat*\endcsname\endalign
244 \let\endxxalignat\endalign
245 \let\endalignat\endalign
246 \expandafter\let\csname endalignat*\endcsname\endalign
247 \let\endflalign\endalign
248 \expandafter\let\csname endflalign*\endcsname\endalign
```

Adjust let-statements.

`\lendmultline` The `multline` environment has two different `\end` commands, depending if the equation numbers are set on the left or on the right:

```
249 \def\lendmultline@{%
250     \@eqnswfalse\tag@true\tagleft@false
251     \rendmultline@}
```

End of `multline` environment if tags are set to the left: in this case, the last line of a `multline` does not contain a tag. Thus the situation of setting an endmark tag at the right is faked:

Lines 250, 251: display no equation number, set a tag, set it at the right, and call `\rendmultline`.

`\rendmultline` `\rendmultline` also uses the augmented tags:

```
252 \def\rendmultline@{%
253     \TagsPlusEndmarks          % <<<<<<<<<
254     \iftag@
255     \begingroup
256     \ifshifftag@
257     \hskip\multlinegap
```



```

330 \expandafter\ifundefined{th@#1}%
331   {\expandafter\gdef\csname th@#1\endcsname{%
332     \def\@begintheorem###1###2{#2}%
333     \def\@opargbegintheorem###1###2###3{#3}}}%
334   {\PackageError{\basename}{Theorem style #1 already defined}\@eha}}

```

Arguments:

$\langle style \rangle := \#1$: the name of the theoremstyle to be defined,
 $\langle cmd_seq1 \rangle := \#2$: command sequence for setting the header for environment instances with no optional text,

$\langle cmd_seq2 \rangle := \#3$: command sequence for setting the header for environment instances with optional text.

Line 330: if this style is not yet defined, define it.

Line 331: define $\backslash th@ \langle style \rangle$ to be a macro which defines

Line 332: a) the two-argument macro $\backslash @begintheorem \#1 \#2$ to be $\langle cmd_seq1 \rangle$,

Line 333: b) $\backslash @opargbegintheorem \#1 \#2 \#3$ to be $\langle cmd_seq2 \rangle$.

The predefined theorem styles use this command.

$\backslash renewtheoremstyle$

```

335 \gdef\renewtheoremstyle#1#2#3{%
336   \expandafter\ifundefined{th@#1}%
337     {\PackageError{\basename}{Theorem style #1 undefined}\@ehc}%
338     {}}%
339   \expandafter\let\csname th@#1\endcsname\relax
340   \newtheoremstyle{#1}{#2}{#3}}

```

Arguments:

$\langle style \rangle := \#1$: the name of the theoremstyle to be defined,
 $\#2, \#3$ as for $\backslash newtheoremstyle$.

Checks, if theoremstyle $\langle style \rangle$ is already defined. If so, $\backslash th@ \langle style \rangle$ is made undefined and $\backslash newtheoremstyle$ is called with the same arguments.

Predefined Theorem Styles

theoremstyles $th@plain, th@change,$ and $th@margin$ taken from theorem.sty by Frank Mittelbach; the break-styles have been changed.

```

341 \newtheoremstyle{plain}%
342   {\item[\hskip\labelsep \theorem@headerfont ##1 \ ##2\theorem@separator]}%
343   {\item[\hskip\labelsep \theorem@headerfont ##1 \ ##2 \ (##3)\theorem@separator]}
344 %
345 \newtheoremstyle{break}%
346   {\item\hbox to \textwidth{\theorem@headerfont ##1 \
347     ##2\theorem@separator\hfill}}%
348   {\item\hbox to \textwidth{\theorem@headerfont ##1 \ ##2 \
349     (##3)\theorem@separator\hfill}}
350 %
351 \newtheoremstyle{change}%
352   {\item[\hskip\labelsep \theorem@headerfont ##2 \ ##1\theorem@separator]}%
353   {\item[\hskip\labelsep \theorem@headerfont ##2 \ ##1 \ (##3)\theorem@separator]}
354 %
355 \newtheoremstyle{changebreak}%
356   {\item\hbox to \textwidth{\theorem@headerfont ##2 \

```

```

357     ##1\theorem@separator\hfill}}%
358   {\item\hbox to \textwidth{\theorem@headerfont ##2\ ##1\
359     (##3)\theorem@separator\hfill}}
360 %
361 \newtheoremstyle{margin}%
362   {\item[\theorem@headerfont \llap{##2}\hskip\labelsep ##1\theorem@separator]}%
363   {\item[\theorem@headerfont \llap{##2}\hskip\labelsep ##1\ (##3)\theorem@separator]}
364 %
365 \newtheoremstyle{marginbreak}%
366   {\item\hbox to \textwidth{\theorem@headerfont \llap{##2}\hskip\labelsep
367     ##1\theorem@separator\hfill}}%
368   {\item\hbox to \textwidth{\theorem@headerfont
369     \llap{##2}\hskip\labelsep
370     ##1\ (##3)\theorem@separator\hfill}}
371 %
372 \newtheoremstyle{nonumberplain}%
373   {\item[\theorem@headerfont\hskip\labelsep ##1\theorem@separator]}%
374   {\item[\theorem@headerfont\hskip\labelsep ##1\ (##3)\theorem@separator]}
375 %
376 \newtheoremstyle{nonumberbreak}%
377   {\item\hbox to \textwidth{\theorem@headerfont
378     ##1\theorem@separator\hfill}}%
379   {\item\hbox to \textwidth{\theorem@headerfont
380     ##1\ (##3)\theorem@separator\hfill}}
381 %
382 \newtheoremstyle{empty}%
383   {\item~\hfill}%
384   {\item\hbox to \textwidth{\theorem@headerfont##3\hfill}}
385 %
386 \@namedef{th@nonumbermargin}{\th@nonumberplain}
387 \@namedef{th@nonumberchange}{\th@nonumberplain}
388 \@namedef{th@nonumbermarginbreak}{\th@nonumberbreak}
389 \@namedef{th@nonumberchangebreak}{\th@nonumberbreak}
390 \@namedef{th@plainNo}{\th@nonumberplain}
391 \@namedef{th@breakNo}{\th@nonumberplain}
392 \@namedef{th@marginNo}{\th@nonumberplain}
393 \@namedef{th@changeNo}{\th@nonumberplain}
394 \@namedef{th@marginbreakNo}{\th@nonumberbreak}
395 \@namedef{th@changebreakNo}{\th@nonumberbreak}

```

For instance, `break` is commented:
`\newtheoremstyle{break}` results in

```

\gdef\th@break{%
  \def\@begintheorem##1##2{%
    \item[\hskip\labelsep \theorem@headerfont
      ##1\ ##2\theorem@separator]%
    \hfill\penalty-8000}%
  \def\@opargbegintheorem##1##2##3{%
    \item[\hskip\labelsep \theorem@headerfont
      ##1\ ##2\ (##3)\theorem@separator]%
    \hfill\penalty-8000}}

```

Then, calling `\th@break` sets `\@begintheorem` as follows:

Since each theorem environment is basically a trivlist, the header is set as the item contents: `\theorem@headerfont` holds the font commands for the header font, `##1` is the keyword to be displayed, and `##2` its environment number. The linebreak after the header is achieved by offering to fill the line with space and the distinct wish to put a linebreak after it. Thus, if plain text follows, the line break is executed, but if a list or a display follows, it is not executed.

Note: The `\hfill\penalty-8000` causes T_EX to leave vertical mode, setting the item contents (ie the header) and entering horizontal mode to perform the `\hfill`.

`\theoremstyle` The handling of `\theoremstyle`, `\theorembodyfont`, and `\theoremskipamounts` is taken from `theorem.sty` by Frank Mittelbach:

```
396 \gdef\theoremstyle#1{%
397   \@ifundefined{th@#1}{\@warning
398     {Unknown theoremstyle ‘#1’. Using ‘plain’}%
399     \theorem@style{plain}}%
400   {\theorem@style{#1}}
401 \newtoks\theorem@style
402 \newtoks\theorem@@style
403 \global\theorem@style{plain}
```

If `\theoremstyle` is called, it is checked if the argument is a valid `theoremstyle`, and if so, it is stored in the token `\theorem@style`. It is initialized to `plain`.

`\theorembodyfont`

```
404 \newtoks\theorembodyfont
405 \global\theorembodyfont{\itshape}
```

`\theoremnumbering`

```
406 \newtoks\theoremnumbering
407 \global\theoremnumbering{arabic}
```

`\theorempreskipamount`

`\theorempostskipamount`

```
408 \newskip\theorempreskipamount
409 \newskip\theorempostskipamount
410 \global\theorempreskipamount\topsep
411 \global\theorempostskipamount\topsep
```

`\theoremindent`

```
412 \newdimen\theoremindent
413 \global\theoremindent0cm
414 \newdimen\theorem@indent
```

`\theoremheaderfont`

```
415 \newtoks\theoremheaderfont
416 \global\theoremheaderfont{\normalfont\bfseries}
417 \def\theorem@headerfont{\normalfont\bfseries}
```

`\theoremseparator`

```
418 \newtoks\theoremseparator
419 \global\theoremseparator{}
420 \def\theorem@separator{}
```

```

\theoremsymbol
421 \newtoks\theoremsymbol
422 \global\theoremsymbol{}

\qedsymbol
423 \newtoks\qedsymbol
424 \global\qedsymbol{}

\theoremkeyword
425 \newtoks\theoremkeyword
426 \global\theoremkeyword{None}

```

```

\theoremclass
427 \gdef\theoremclass#1{%
428   \csname th@class@#1\endcsname}
429 \gdef\th@class@LaTeX{%
430   \theoremstyle{plain}
431   \theoremheaderfont{\normalfont\bfseries}
432   \theorembodyfont{\itshape}
433   \theoremseparator{}
434   \theoremindent0cm
435   \theoremnumbering{arabic}
436   \theoremsymbol{}}

```

Calling `\theoremclass{<env>}` calls `\th@class@<env>` (which is defined in `\@newtheorem` in Lines -45535). `\th@class@<env>` restores all style parameters to their values given for `<env>`. Especially, `\th@class@LaTeX` restores the standard LaTeX parameters.

```

\qedsymbol
437 \newtoks\qedsymbol
438 \global\qedsymbol{}

```

Compatibility with amsthm.

```

amsthm
439 \DeclareOption{amsthm}{%
440   \PackageInfo{\basename}{Option 'amsthm' loaded}%
441   \def\swapnumbers{\PackageError{\basename}{swapnumbers not implemented.
442     Use theoremstyle change instead.}\@eha}
443
444 \gdef\th@plain{%
445   \def\theorem@headerfont{\normalfont\bfseries}\itshape%
446   \def\@begintheorem##1##2{%
447     \item[\hskip\labelsep \theorem@headerfont ##1\ ##2.]}%
448   \def\@opargbegintheorem##1##2##3{%
449     \item[\hskip\labelsep \theorem@headerfont ##1\ ##2\ (##3).]}%
450 \gdef\th@nonumberplain{%
451   \def\theorem@headerfont{\normalfont\bfseries}\itshape%
452   \def\@begintheorem##1##2{%
453     \item[\hskip\labelsep \theorem@headerfont ##1.]}%
454   \def\@opargbegintheorem##1##2##3{%
455     \item[\hskip\labelsep \theorem@headerfont ##1\ (##3).]}%

```

```

456 \gdef\th@definition{%
457   \th@plain\def\theorem@headerfont{\normalfont\bfseries}\itshape}
458 \gdef\th@nonumberdefinition{%
459   \th@nonumberplain\def\theorem@headerfont{\normalfont\bfseries}\itshape}
460 \gdef\th@remark{%
461   \th@plain\def\theorem@headerfont{\itshape}\normalfont}
462 \gdef\th@nonumberremark{%
463   \th@nonumberplain\def\theorem@headerfont{\itshape}\normalfont}
464 \newcounter{proof}%
465 \if@thmmarks
466   \newcounter{currproofctr}%
467   \newcounter{endproofctr}%
468 \fi
469 \newcommand{\openbox}{\leavevmode
470   \hbox to.7778em{%
471     \hfil\vrule
472     \vbox to.675em{\hrule width.6em\vfil\hrule}%
473     \vrule\hfil}}
474 \gdef\proofSymbol{\openbox}
475 \newcommand{\proofname}{Proof}
476 \newenvironment{proof}[1][\proofname]{
477   \th@nonumberplain
478   \def\theorem@headerfont{\itshape}%
479   \normalfont
480   \theoremsymbol{\ensuremath{\_}\blacksquare}}
481   \@thm{proof}{proof}{#1}}%
482   {\@endtheorem}
483 }% end of option amsthm

```

Defines theorem styles plain, definition, and remark, and environment proof according to amsthm.sty.

7.1.6 Theorem-Environment Handling Stuff

Original: ltthm.dtx

```
484 \newskip\thm@topseppadd
```

An auxiliary variable.

Defining New Theorem-Environments.

`\newtheorem` The syntax of `\newtheorem` is retained. The macro is extended to deal with the additional requirements:

```

485 \gdef\newtheorem{%
486   \@ifstar
487   {\expandafter\@ifundefined{th@nonumber\the\theorem@style}%
488   {\PackageError{\basename}{Theorem style {\nonumber\the\theorem@style}
489     undefined (you need it here for newtheorem*) }\@ehc}%
490   }%
491   \edef\@tempa{{\nonumber\the\theorem@style}}%
492   \expandafter\theorem@@style\@tempa\@newtheorem}%
493   {\edef\@tempa{{\the\theorem@style}}%
494   \expandafter\theorem@@style\@tempa\@newtheorem}}

```

Defines `\theorem@style` to be the current `\theoremstyle` or – in case of `\newtheorem*` – to be its non-numbered equivalent (which has to be defined!), and then calls `\@newtheorem`.

`\renewtheorem`

```

495 \gdef\renewtheorem{%
496   \ifstar
497   {\expandafter\ifundefined{thnonumber\the\theorem@style}%
498   {\PackageError{\basename}{Theorem style {nonumber\the\theorem@style}
499     undefined (you need it here for newtheorem*) }\@ehc}%
500   }%
501   \edef\@tempa{{nonumber\the\theorem@style}}%
502   \expandafter\theorem@style\@tempa\@renewtheorem}%
503   {\edef\@tempa{{\the\theorem@style}}%
504   \expandafter\theorem@style\@tempa\@renewtheorem}}

```

Analogous to `\newtheorem`.

`\@newtheorem` `\@newtheorem` does the main job for initializing a new theorem environment type. It is called by `\newtheorem`.

```

505 \gdef\@newtheorem#1{%
506   \thm@tempiffalse
507   \expandafter\ifdefinable\csname #1\endcsname
508   {\expandafter\ifdefinable\csname #1*\endcsname
509   {\thm@tempiftrue
510     \thm@definethm{#1}% for lists
511     \if@thmmarks
512       \expandafter\ifundefined{c@curr#1ctr}%
513       {\newcounter{curr#1ctr}}}%
514       \expandafter\ifundefined{c@end#1ctr}%
515       {\newcounter{end#1ctr}}}%
516     \fi
517     \expandafter\protected@xdef\csname #1Symbol\endcsname{\the\theoremsymbol}%
518     \expandafter\gdef\csname#1\endcsname{%
519       \let\thm@starredenv\@undefined
520       \csname mkheader@#1\endcsname}%
521     \expandafter\gdef\csname#1*\endcsname{%
522       \let\thm@starredenv\relax
523       \csname mkheader@#1\endcsname}%
524     \def\@tempa{\expandafter\noexpand\csname end#1\endcsname}%
525     \expandafter\xdef\csname end#1*\endcsname{\@tempa}%
526     \expandafter\xdef\csname setparms@#1\endcsname
527     {\noexpand \def \noexpand \theorem@headerfont
528      {\the\theoremheaderfont\noexpand\theorem@checkbold}}%
529     \noexpand \def \noexpand \theorem@separator
530     {\the\theoremseparator}%
531     \noexpand \def \noexpand \theorem@indent
532     {\the\theoremindent}}%
533     \the \theorembodyfont
534     \noexpand\csname th@\the \theorem@style \endcsname}%
535   \expandafter\xdef\csname th@class#1\endcsname
536   {\noexpand\theoremstyle{\the\theorem@style}%
537   \noexpand\theoremheaderfont{\the\theoremheaderfont}}%
538   \noexpand\theorembodyfont{\the \theorembodyfont}}%

```

```

539     \noexpand\theoremseparator{\the\theoremseparator}%
540     \noexpand\theoremindent\the\theoremindent%
541     \noexpand\theoremnumbering{\the\theoremnumbering}%
542     \noexpand\theoremsymbol{\the\theoremsymbol}%
543   }}%
544   \@ifnextchar[{\@othm{#1}}{\@nthm{#1}}}% MUST NOT BE IN ANY IF !!!

```

Argument: $\langle env \rangle := \#1$ is the (internal) environment name to be defined, which is read from the L^AT_EX source.

Line 507: check if $\langle env \rangle$ is not yet defined (or is redefined).

Lines 509–534 are executed exactly if $\langle env \rangle$ and $\langle env \rangle^*$ are not yet defined.

Line 509: \thm@tempif=true iff $\langle env \rangle$ and $\langle env \rangle^*$ are not yet defined.

Line 510: Initialize theorem list handling for $\langle env \rangle$.

Lines 512–515: if thmmarks is active and the counters are not yet defined, for every theorem-like, define

$\text{curr}\langle env \rangle\text{ctr}$: in the i th environment of type $\langle env \rangle$, $\text{curr}\langle env \rangle\text{ctr} = i$, and
 $\text{end}\langle env \rangle\text{ctr}$: when the innermost environment is of type $\langle env \rangle$, in the j th potential position for an end mark in this environment, $\text{end}\langle env \rangle\text{ctr} = j$. (if the counters are already defined, $\langle env \rangle$ is redefined, and these internal counters have to be continued).

Lines 517–534: define several commands: (\xdef expands the definition at the time it is called and makes it global):

Line 517: store the current value of \theoremsymbol (\edef : expand $\text{\the\theoremsymbol}$ now) as $\langle env \rangle\text{Symbol}$.

Lines 518–520, 521–523: Define the commands \env and \env^* to set the header of \env . (using a switch \thm@starredenv : \relax iff starred).

Lines 524, 525: Set $\text{\end}\langle env \rangle^*$ to $\text{\end}\langle env \rangle$.

Lines 526–534: define $\text{\setparms}\langle env \rangle$ to set the style parameters of the header for every $\langle env \rangle$ environment (in the sequel, *current* means, at the moment \@newtheorem is called):

Lines 527, 528: setting $\text{\theorem@headerfont}$ to the *current* value of $\text{\theoremheaderfont}$, followed by a check if it is a bold style,

Lines 529, 530: setting $\text{\theorem@separator}$ to the *current* value of \theoremseparator ,

Lines 531, 532: setting \theorem@indent to the *current* value of \theoremindent ,

Line 533: executing the command sequence currently stored in \theorembodyfont , and

Line 534: calling $\text{th}\langle the \rangle\text{\theorem@@style}$ (which initializes \@begintheorem and $\text{\@opargbegintheorem}$ according to the *current* value of \theoremstyle by calling $\text{th}\langle the \rangle\text{\theorem@@style}$).

Line 535–549: define $\text{\th@class}\langle env \rangle$ to initialize all style parameters as they are set for the $\langle env \rangle$ environment.

Note, that the \@ifdefinable from line 507 ends after line 549.

Line 550: According to the next character, call $\text{\@othm}\langle env \rangle$ (if another counter is used) or $\text{\@nthm}\langle env \rangle$.

Thus, when calling \@newthm with $\#1 = \langle env \rangle$, for current values $\text{\theoremstyle=plain}$, $\text{\theorembodyfont}=\text{\upshape}$, $\text{\theoremheaderfont}=\text{\bf}$, $\text{\theoremseparator}=:$, $\text{\theoremindent}=1\text{cm}$, $\text{\theoremnumbering}=\text{arabic}$, and $\text{\theoremsymbol}=\text{\Box}$,

the macro `\setparms@⟨env⟩` is defined as

```
\setparms@⟨env⟩ == \def\theorem@headerfont{\bf\theorem@checkbold}
                  \def\theorem@separator{:}
                  \def\theorem@indent{0cm}
                  \upshape
                  \th@plain
```

and the macro `\th@class@⟨env⟩` is defined as

```
\th@class@⟨env⟩ == \def\theoremstyle{plain}
                   \def\theoremheaderfont{\bf}
                   \def\theorembodyfont{\upshape}
                   \def\theoremseparator{:}
                   \def\theoremindent{0cm}
                   \def\theoremnumbering{arabic}
                   \def\theoremsymbol{\Box}
```

Note, that line 533 must not be inside *any* `\if... \fi` construct.

`\@renewtheorem`

```
545 \gdef\@renewtheorem#1{%
546   \expandafter\@ifundefined{#1}%
547   {\PackageError{\basename}{Theorem style #1 undefined}\@ehc}%
548   }%
549   \expandafter\let\csname #1\endcsname\relax
550   \expandafter\let\csname #1*\endcsname\relax
551   \@newtheorem{#1}}
```

Argument: `⟨env⟩:=#1` is the (internal) environment name to be redefined, which is read from the L^AT_EX source.

If `⟨env⟩` is already defined, make it (and `⟨env⟩*`, too) undefined and call `\@newtheorem{⟨env⟩}`.

`\@nthm` `\@nthm` is called by `\@newtheorem` if the environment to be defined has a counter of its own.

```
552 \gdef\@nthm#1#2{%
553   \expandafter\protected@xdef\csname num@addtheorem#1\endcsname{%
554     \noexpand\@num@addtheoremline{#1}{#2}}%
555   \expandafter\protected@xdef\csname nonum@addtheorem#1\endcsname{%
556     \noexpand\@nonum@addtheoremline{#1}{#2}}%
557   \theoremkeyword{#2}%
558   \expandafter\protected@xdef\csname #1keyword\endcsname
559   {\the\theoremkeyword}%
560   \@ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

Arguments:

`⟨env⟩:=#1` is the (internal) environment name to be defined (transmitted from `\@newtheorem`).

`⟨output_name⟩:=#2` is its keyword to be used in the output (read from the L^AT_EX source).

Lines 553–556: Define `\(no)num@addtheoremline⟨env⟩` to call

`\@(no)num@addtheoremline{⟨env⟩}{⟨output_name⟩}`.

For comments on `\@num@addtheoremline` and `\@nonum@addtheoremline` see Section 7.1.8.

Lines 557–559: Define $\langle env \rangle$ Keyword $\langle env \rangle$ to typeset/output $\langle output_name \rangle$. (note the similarity with the handling of \backslash theoremsymbol for handling complex keywords)

Line 560: According to the next character, call \backslash @xnthm $\langle env \rangle$ { $\langle output_name \rangle$ } (if $\langle env \rangle$ -environments should be numbered relative to some structuring level) or \backslash @ynthm $\langle env \rangle$ { $\langle output_name \rangle$ }.

\backslash @othm \backslash @othm is called by \backslash @newtheorem if the environment to be defined uses another counter.

```

561 \gdef\@othm#1[#2]#3{%
562   \ifundefined{c@#2}{\nocounterr{#2}}%
563   {\ifthm@tempif
564     \global\@namedef{the#1}{\@nameuse{the#2}}%
565     \expandafter\protected@xdef\csname num@addtheorem#1\endcsname{%
566       \noexpand\@num@addtheorem#1}{#3}}%
567     \expandafter\protected@xdef\csname nonum@addtheorem#1\endcsname{%
568       \noexpand\@nonum@addtheorem#1}{#3}}%
569     \theoremkeyword{#3}%
570     \expandafter\protected@xdef\csname #1Keyword\endcsname
571     {\the\theoremkeyword}%
572     \expandafter\gdef\csname mkheader@#1\endcsname
573     {\csname setparms@#1\endcsname
574       \@thm{#1}{#2}{#3}}%
575     \global\@namedef{end#1}{\@endtheorem}\fi}
```

Arguments:

$\langle env \rangle$:=#1 is the (internal) environment name to be defined (transmitted from \backslash @newtheorem).

$\langle use_ctr \rangle$:=#2 is the internal name of the theorem which counter is used, and

$\langle output_name \rangle$:=#3 is its “name” to be used in the output (both read from the L^AT_EX source).

Line 562: if the counter to be used is undefined, goto error, else set \backslash the $\langle env \rangle$ to use \backslash the $\langle use_ctr \rangle$ and do the following:

Lines 564–572 happen only if $\langle env \rangle$ is not yet defined or gets redefined:

Line 564: (from latex.ltx) make $\langle env \rangle$ use the counter $\langle use_ctr \rangle$.

Lines 565–571 similar to lines 553–559 of \backslash @nthm.

Lines 572–574 define \backslash mkheader $\langle env \rangle$ to set the style parameters of the header and set the header (by \backslash @thm):

$$\backslash$$
mkheader $\langle env \rangle$ == \backslash setparms $\langle env \rangle$ \backslash @thm $\langle env \rangle$ { $\langle use_ctr \rangle$ }{ $\langle output_name \rangle$ }.

(\backslash setparms $\langle env \rangle$ is defined when \backslash @newtheorem $\langle env \rangle$ is carried out).

Line 575: (from latex.ltx): \backslash end $\langle env \rangle$ calls \backslash @endtheorem.

\backslash @xnthm \backslash @xnthm is called by \backslash @nthm if the numbering is relative to some structuring level.

```

576 \gdef\@xnthm#1#2[#3]{%
577   \ifthm@tempif
578     \expandafter\ifundefined{c@#1}%
579     {\@definecounter{#1}}{}%
580     \@newctr{#1}[#3}%
581     \expandafter\xdef\csname the#1\endcsname{%
582     \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
```

```

583         {\noexpand\csname\the\theoremnumbering\endcsname{#1}}}%
584     \expandafter\gdef\csname mkheader@#1\endcsname
585         {\csname setparms@#1\endcsname
586         \@thm{#1}{#1}{#2}}%
587     \global\@namedef{end#1}{\@endtheorem}\fi}

```

Arguments:

$\langle env \rangle := \#1$ is the (internal) environment name to be defined (transmitted from $\@newtheorem$).

$\langle output_name \rangle := \#2$ is its keyword to be used in the output,

$\langle level \rangle := \#3$ is the structuring level relative to which $\langle env \rangle$ has to be numbered (both read from the L^AT_EX source).

Lines 578–587 happen only if $\langle env \rangle$ is not yet defined or gets redefined:

Lines 578,579: in not yet defined, define $\langle env \rangle$ - counter (otherwise, $\langle env \rangle$ is redefined).

Line 581: (from latex.ltx): define the counter for $\langle env \rangle$ and add $\langle level \rangle$ to its reset-triggers.

Lines 582, 583: define $\the\langle env \rangle$ to be the command sequence

```
\the\langle level \rangle \@thmcountersep\langle numbering \rangle{\langle env \rangle} ,
```

where $\langle numbering \rangle$ is the value of \theoremnumbering when $\@xnthm$ (and thus, $\newtheorem{\langle env \rangle}$) is called.

Lines 584–586: define $\mkheader@{\langle env \rangle}$ to set the style parameters of the header and set the header (by $\@thm$):

```
\mkheader@{\langle env \rangle} == \setparms@{\langle env \rangle}\@thm{\langle env \rangle}{\langle output\_name \rangle}.
```

($\setparms@{\langle env \rangle}$ is defined when $\@newtheorem{\langle env \rangle}$ is carried out).

Line 587: (from latex.ltx): $\end{\langle env \rangle}$ calls $\@endtheorem$.

$\@ynthm$ $\@ynthm$ is called by $\@nthm$ if the counter is not relative to any structuring level.

```

588 \gdef\@ynthm#1#2{%
589     \ifthm@tempif
590         \expandafter\@ifundefined{c@#1}%
591         {\@definecounter{#1}}{\}%
592     \expandafter\xdef\csname the#1\endcsname
593         {\noexpand\csname\the\theoremnumbering\endcsname{#1}}%
594     \expandafter\gdef\csname mkheader@#1\endcsname
595         {\csname setparms@#1\endcsname
596         \@thm{#1}{#1}{#2}}%
597     \global\@namedef{end#1}{\@endtheorem}\fi}

```

Arguments:

$\langle env \rangle := \#1$ is the (internal) environment name to be defined (transmitted from $\@newtheorem$).

$\langle output_name \rangle := \#2$ is its keyword to be used in the output.

$\@ynthm$ works analogous to $\@xnthm$.

Handling Instances of Theorem-Environments.

$\@thm$ $\@thm$ is called by $\@{\langle env \rangle}$ (which is defined by $\@othm/\@xnthm/\@ynthm$).

```

598 \gdef\@thm#1#2#3{%
599   \if@thmmarks
600     \stepcounter{end\InTheoType ctr}%
601   \fi
602   \renewcommand{\InTheoType}{#1}%
603   \if@thmmarks
604     \stepcounter{curr#1ctr}%
605     \setcounter{end#1ctr}{0}%
606   \fi
607   \refstepcounter{#2}%
608   \thm@topsepadd \theorempostskipamount % cf. latex.ltx: \@trivlist
609   \ifvmode \advance\thm@topsepadd\partopsep\fi
610   \@trivlist
611   \@topsep \theorempreskipamount
612   \@topsepadd \thm@topsepadd % used by \@endparenv
613   \advance\linewidth -\theorem@indent
614   \advance\@totalleftmargin \theorem@indent
615   \parshape \@ne \@totalleftmargin \linewidth
616   \@ifnextchar[{\@ythm{#1}{#2}{#3}}{\@xthm{#1}{#2}{#3}}

```

Changed to three instead of two parameters (the first one is new):

$\langle env \rangle := \#1$: (added) internal name of the theorem environment,
 $\langle use_ctr \rangle := \#2$: internal name of the theorem which counter is used,
 $\langle output_name \rangle := \#3$: keyword to be displayed in the output; all arguments are transmitted from $\@othm/\@xnthm/\@ynthm$.

Lines 599–601: if `thmmarks` is active, the counter for the current environment $\langle env \rangle$ is incremented, since the last endmark in environment $\langle env \rangle$ is definitely not the position for its endmark (necessary for nested environments ending at the same time).

Line 602: set \InTheoType to $\langle env \rangle$.

Lines 603–606: if `thmmarks` is active, increment $\curr\langle env \rangle\ctr$ and set $\end\langle env \rangle\ctr$ to 0.

Line 607: adapted from `latex.ltx`: increment the corresponding counter.

Lines 608–612: handle \theorempreskipamount and \theorempostskipamount (if in `vmode`, there is additional space, cf. $\@trivlist$ and $\@trivlist$ in `latex.ltx`).

Lines 613–615: handle \theoremindent .

Line 616: if there is an optional argument, call $\@ythm\langle env \rangle\{\langle use_ctr \rangle\}\{\langle output_name \rangle\}$, otherwise call $\@xthm\langle env \rangle\{\langle use_ctr \rangle\}\{\langle output_name \rangle\}$.

$\@xthm$ $\@xthm$ is called by $\@thm$ if there is no optional text in the theorem header.

```

617 \def\@xthm#1#2#3{%
618   \@begintheorem{#3}{\csname the#2\endcsname}%
619   \ifx\thm@starredenv\@undefined
620     \thm@thmcaption{#1}{#3}{\csname the#2\endcsname}{}}\fi
621 \ignorespaces}

```

Changed to three instead of two parameters (the first one is new):

$\langle env \rangle := \#1$: (added) internal name of the theorem environment,
 $\langle use_ctr \rangle := \#2$: internal name of the theorem which counter is used,
 $\langle output_name \rangle := \#3$: keyword to be displayed in the output.

All arguments are transmitted from $\@thm$.

For comments, see $\@ythm$.

`\@ythm` `\@ythm` is called by `\@thm` if there is an optional text in the theorem header.

```

622 \def\@ythm#1#2#3[#4]{%
623   \expandafter\global\expandafter\def\csname#1name\endcsname{#4}%
624   \@opargbegintheorem{#3}{\csname the#2\endcsname}{#4}%
625   \ifx\thm@starredenv\undefined
626     \thm@thmcaption{#1}{#3}{\csname the#2\endcsname}{#4}\fi%
627   \ignorespaces}

```

Changed to four instead of three parameters (the first one is new):

`<env>:=#1`: (added) internal name of the theorem environment,
`<use_ctr>:=#2`: internal name of the theorem which counter is used,
`<output_name>:=#3`: keyword to be displayed in the output.
`<opt_text>:=#4`: optional text to appear in the header.

`#1–#3` are transmitted from `\@thm`, `#4` is read from the L^AT_EX source.

Line 623: define `\<env>name` to be the optional argument.

Line 624: call

```
\@opargbegintheorem{<output_name>}{\the<use_ctr>}{<opt_text>}
```

which outputs the header.

Line 625, 626: if `<env>` is not the starred version, call

```
\thm@thmcaption{<env>}{<output_name>}{\the<use_ctr>}{<opt_text>}}
```

which makes an entry into the theorem list.

`\@endtheorem` `\@endtheorem` it is only changed by option `[thmmarks]`. Otherwise refer to the definition in `latex.ltx`

7.1.7 Extended Referencing Facilities

`\label` The original `\label` macro is extended (cf. `ltxref.dtx`) with an optional argument, containing the type of the labeled construct:

```

628 \DeclareOption{thref}{%*****
629   \PackageInfo{basename}{Option ‘thref’ loaded}%
630 \def\label#1{%
631   \ifnextchar[%]
632     {\label@optarg{#1}}%
633     {\thm@makelabel{#1}}
634 %
635 \def\thm@makelabel#1{%
636   \@bbsphack
637   \edef\thm@tmp{\expandafter\expandafter\expandafter\thm@meaning
638     \expandafter\meaning\csname\InTheoType Keyword\endcsname\relax}%
639   \protected@write\@auxout{}%
640     {\string\newlabel{#1}{\@currentlabel}{\thepage}}[\thm@tmp]}%
641   \@eesphack}
642 %
643 \def\label@optarg#1[#2]{%
644   \@bsphack
645   \protected@write\@auxout{}%
646     {\string\newlabel{#1}{\@currentlabel}{\thepage}}[#2]}%
647   \@esphack}

```

`thm@makelabel`: If no optional argument is given, the keyword of the current environment type is used instead.

`label@optarg`: The original definition, extended with the optional argument which is appended to the `\newlabel`-command to be written to the .aux-file.

`\newlabel` The original behavior of `\newlabel` (called when evaluating the .aux-file) is also adapted.

Original syntax: `\newlabel{<label>}{<{section}>}{<page>}`

Modified syntax: `\newlabel{<label>}{<{section}>}{<page>}[<type>]`

Definition of `\newlabel`: `\def\newlabel{\@newl@bel r}`.

Therefore, the modification is encoded into the `\@newl@bel` macro:

```
648 \def\@newl@bel#1#2#3{%
649   \ifpackageloaded{babel}{\@safe@activestruer}\relax%
650   \@ifundefined{#1@#2}%
651     \relax
652     {\gdef \@multiplelabels {%
653       \@latex@warning@no@line{There were multiply-defined labels}}%
654       \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
655   \global\@namedef{#1@#2}{#3}%
656   \@ifnextchar[{\set@label@type{#1}{#2}}%
657     \relax}%
658 \def\set@label@type#1#2[#3]{%
659   \global\@namedef{#1@#2@type}{#3}}
```

the macro is called with three arguments (same as originally):

`#1=r`,

`<labelname> := #2` is the label name,

`#3` is a pair (section, page-number) consisting of the values needed for `\ref` and `\pageref`, respectively.

Line 649: adaptation to `babel`

Lines 650–655: The original definition (both standard L^AT_EX and `babel`).

Line 656: if an optional argument follows (containing the environment-type), continue with `\set@label@type`, otherwise return (the original behavior).

Lines 658,659: set `\r@<labelname>@type` to the type of the respective environment.

`\thref` `\thref` is an adaptation of `\ref`:

```
660 \def\thref#1{%
661   \expandafter\ifx\csname r@#1@type\endcsname\None
662     \PackageWarning{\basename}{thref: Reference Type of ‘#1’ on page
663       \thepage \space undefined}\G@refundefinedtrue
664     \else\csname r@#1@type\endcsname~\fi%
665   \expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
666 }% end of option thref
```

Lines 660,665: similar to `\ref`.

Line 649: if a legal theorem type is given, then output `\r@<labelname>@type` and avoid linebreaking between the type and the number.

7.1.8 Generation of Theorem Lists

The following macros are needed for the generation of theorem-lists. We will document it for the theorem `\begin{definition}[optional]`, which we assume to be the first definition at all and which is placed on page 5.

`\thm@thmcaption` This macro, used internally, strips of the outer brackets from the second argument and calls `\thm@@thmcaption`. It's typically called like this

```
\thm@thmcaption{definition}{\Definition}{1}{optional}}
```

(internal name of the environment, output keyword, running number, optional text)
667 `\def\thm@thmcaption#1#2{\thm@@thmcaption{#1}#2}`

`\thm@@thmcaption` `\thm@caption` is called from `\thm@caption`; it writes an appropriate entry to the `.thm`-file.

```
668 \def\thm@@thmcaption#1#2#3#4{%
669   \thm@parseforwriting{#2}%
670   \let\thm@tmpii\thm@tmp%
671   \thm@parseforwriting{#4}%
672   \edef\thm@t{\thm@tmpii}{#3}{\thm@tmp}}%
673   \addcontentsline{thm}{#1}{\thm@t}}
```

Arguments: `\langle env \rangle := #1` is the internal environment name, `\langle output_name \rangle := #2` is its keyword to be used in the output, `#3` is the running number, and `#4` is the optional text argument in the header.

Lines 668,669: the command sequence for the output keyword is prepared by `\thm@parseforwriting` (which returns `\thm@tmpii`) and then stored in `\thm@tmpii`.

Line 670: the optional text is also prepared by `\thm@parseforwriting`

Lines 671,672: The output is collected and written into the `.aux` file, which will forward it to the theorem-file.

The following two macros are just shortcuts, often needed for the output of one single line in the theorem-lists. The first one is used in unnamed lists, the second one in named. Warning: Don't remove the leading `\let`, since you will get wrong `\if- \fi -nesting` without it, if you don't use `hyperref`.

`\thm@@thmline@noname`

```
674 \def\thm@@thmline@noname#1#2#3#4{%
675   \@dottedtocline{-2}{0em}{2.3em}%
676   {\protect\numberline{#2}#3}%
677   {#4}}
```

`\thm@@thmline@name`

```
678 \def\thm@@thmline@name#1#2#3#4{%
679   \@dottedtocline{-2}{0em}{2.3em}%
680   {#1 \protect\numberline{#2}#3}%
681   {#4}}
```

`\thm@thmline` This is another short one, which only discards the outer brackets from the first argument and calls `\thm@@thmline`. It's normally called like this:

```
\thm@@thmline{\Definition}{1}{optional}}{5}
```

```
682 \def\thm@thmline#1#2{\thm@@thmline#1{#2}}
```

`\thm@lgobble` The next macro is used to ignore all theorem-sets, which should not be listed.

```
683 \long\def\thm@lgobble#1#2{\ignorespaces}
```

The following four macros set up the predefined list-types. To do so, they define the internal macros `\thm@@thmlstart` (containing the code to be executed at the beginning of the list), `\thm@@thmlend` (code to be executed at the end of the list) and `\thm@@thmline` (code to be executed for every line). In order to gain compatibility with `newthm.sty`, we decided not to make this commands inaccessible to the user. But we recommend not using these commands, because they may disappear in later distributions.

`\theoremlistall` This one implements the type `all`.

```
684 \def\theoremlistall{%
685   \let\thm@@thmlstart=\relax
686   \let\thm@@thmlend=\relax
687   \let\thm@@thmline=\thm@@thmline@noname}
```

`\theoremlistallname` And here's the type `allname`.

```
688 \def\theoremlistallname{%
689   \let\thm@@thmlstart=\relax
690   \let\thm@@thmlend=\relax
691   \let\thm@@thmline=\thm@@thmline@name}
```

`\theoremlistoptional` This one is the list-type `opt`. In case of `[hyperref]`, the fifth argument, which is provided by `hyperref.sty` is automatically given to `\thm@@thmline@noname`.

```
692 \def\theoremlistoptional{%
693   \let\thm@@thmlstart=\relax
694   \let\thm@@thmlend=\relax
695   \def\thm@@thmline##1##2##3##4{%
696     \ifx\empty ##3%
697       \else%
698         \thm@@thmline@noname{##1}{##2}{##3}{##4}
699       \fi}}
```

`\theoremlistoptname` And the last type, `optname`. In case of `[hyperref]`, the fifth argument, which is provided by `hyperref.sty` is automatically given to `\thm@@thmline@name`.

```
700 \def\theoremlistoptname{%
701   \let\thm@@thmlstart=\relax
702   \let\thm@@thmlend=\relax
703   \def\thm@@thmline##1##2##3##4{%
704     \ifx\empty ##3%
705       \else%
706         \thm@@thmline@name{##1}{##2}{##3}{##4}
707       \fi}}
```

Theoremlists and Hyperref Since the `hyperref`-package redefines `\contentsline`, we must redefine some commands to handle this:

1. Let the different versions of `\thm@@thmline@..` take a 5th argument, the one provided by `hyperref`.
2. Don't use `hyperref`'s `contentsline`: restore the normal definition at the beginning of `\thm@processlist` (see there).
3. Let `\thm@lgobble` take a 3rd argument, the one provided by `hyperref`.

4. Do the hyperlinks manually in the different versions of `\thm@thmline` as defined by the theoremtypes.

```

708 \DeclareOption{hyperref}{%
709   \def\thm@thmline@noname#1#2#3#4#5{%
710     \ifx\#5\%
711       \@dottedtocline{-2}{0em}{2.3em}%
712         {\protect\numberline{#2}#3}%
713         {#4}
714     \else
715       \ifhy@linktocpage\relax\relax
716         \@dottedtocline{-2}{0em}{2.3em}%
717         {\protect\numberline{#2}#3}%
718         {\hyper@linkstart{link}{#5}{#4}\hyper@linkend}
719       \else
720         \@dottedtocline{-2}{0em}{2.3em}%
721         {\hyper@linkstart{link}{#5}{\protect\numberline{#2}#3}%
722           \hyper@linkend}%
723         {#4}
724     \fi
725   \fi}
726 \def\thm@thmline@name#1#2#3#4#5{%
727   \ifx\#5\%
728     \@dottedtocline{-2}{0em}{2.3em}%
729     {#1 \protect\numberline{#2}#3}%
730     {#4}
731   \else
732     \ifhy@linktocpage\relax\relax
733       \@dottedtocline{-2}{0em}{2.3em}%
734       {#1 \protect\numberline{#2}#3}%
735       {\hyper@linkstart{link}{#5}{#4}\hyper@linkend}
736     \else
737       \@dottedtocline{-2}{0em}{2.3em}%
738       {\hyper@linkstart{link}{#5}%
739         {#1 \protect\numberline{#2}#3}\hyper@linkend}%
740       {#4}
741     \fi
742   \fi}
743 \def\thm@thmline#1#2#3{\thm@thmline#1{#2}{#3}}
744 \long\def\thm@lgobble#1#2#3{\ignorespaces}
745 \def\theoremlistoptional{%
746   \let\thm@thmlstart=\relax
747   \let\thm@thmlend=\relax
748   \def\thm@thmline##1##2##3##4##5{%
749     \ifx\empty ##3%
750     \else%
751       \thm@thmline@noname{##1}{##2}{##3}{##4}{##5}
752     \fi}}
753 \def\theoremlistoptname{%
754   \let\thm@thmlstart=\relax
755   \let\thm@thmlend=\relax
756   \def\thm@thmline##1##2##3##4##5{%
757     \ifx\empty ##3%
758     \else%

```

```

759             \thm@thml@name{##1}{##2}{##3}{##4}{##5}
760             \fi}}
761 }% end of option hyperref

```

`\theoremlisttype` The next one is the user-interface for selecting the list-type. It simply calls `\thm@thml@<type>`, if the given `<type>` is defined.

```

762 \def\theoremlisttype#1{%
763   \@ifundefined{thm@thml@#1}%
764     {\PackageError{\basename}{Listtype #1 not defined}\@eha}
765     {\csname thm@thml@#1\endcsname}}

```

Now, here is a the code, which maps the types – selected by `\theoremlisttype` – to the defined macros.

```

766 \def\thm@thml@all{\theoremlistall}
767 \def\thm@thml@opt{\theoremlistoptional}
768 \def\thm@thml@optname{\theoremlistoptname}
769 \def\thm@thml@allname{\theoremlistallname}

```

`\newtheoremlisttype` According to the given documentation, this one can be used to define new list-types. It's done by defining the macro `\thm@thml@<type>`, which *locally* redefines the commands `\thm@thmlstart`, `\thm@thml@line` and `\thm@thmlend`.

```

770 \def\newtheoremlisttype#1#2#3#4{%
771   \@ifundefined{thm@thml@#1}%
772     {\expandafter\gdef\csname thm@thml@#1\endcsname{%
773       \def\thm@thmlstart{#2}%
774       \def\thm@thml@line###1###2###3###4{#3}%
775       \def\thm@thmlend{#4}}%
776     }{\PackageError{\basename}{list type #1 already defined}\@eha}}

```

`\renewtheoremlisttype`

```

777 \def\renewtheoremlisttype#1#2#3#4{%
778   \@ifundefined{thm@thml@#1}%
779     {\PackageError{\basename}{List type #1 not defined}\@ehc}{}%
780   \expandafter\let\csname thm@thml@#1\endcsname\relax
781   \newtheoremlisttype{#1}{#2}{#3}{#4}}

```

if the list type to be redefined is already defined, make it undefined and define it.

`\thm@definethm` For each theorem-set, we need to setup two commands. Thus, the next macro is called by `\newtheorem`. It defines the command `\l@<theorem-set>` and `\thm@listdo<theorem-set>`, which initially discard the input by calling `\thm@lgobble`. The first one is called for each theorem when you are generating lists. The second one is called, if you've added something with `\addtotheoremfile`.

```

782 \def\thm@definethm#1{%
783   \expandafter\gdef\csname l@#1\endcsname{\thm@lgobble}%
784   \expandafter\gdef\csname thm@listdo#1\endcsname{\thm@lgobble}}

```

`\thm@inlistdo` If in some case, you've written additional text into the theorem-file by `\addtotheoremfile`, this one is called internally. It simply discards the first argument and strips of the outer brackets from the second one.

```

785 \long\def\thm@inlistdo#1#2{#2}%

```

`\listtheorems` Now we need the user-interface for generating lists. This is done by the next macro. We set the `tocdepth` to `-2` to assure that the predefined list-types work. After storing the names of the theorem-sets, we call `\thm@processlist`, which actually generates the list.

```
786 \def\listtheorems#1{\begingroup%
787   \c@tocdepth=-2%
788   \def\thm@list{#1}\thm@processlist%
789   \endgroup}
```

`\thm@processlist` To generate the specified list, we redefine globally the commands `\l@{theorem-set}` and `\thm@listdo{theorem-set}`. Remember that these initially were set to ignore everything. After reading in the theorem-file, we redefine the mentioned commands to discard everything again. We need to define the macros globally for dealing with complex, user-defined, list-types.

```
790 \def\thm@processlist{%
791   \begingroup%
792   \typeout{** Generating table of \thm@list}%
793   \def\contentsline##1{\csname l@##1\endcsname}%
794   \thm@thmlstart
795   \@for\thm@currentlist:=\thm@list\do{%
796     \ifx\thm@currentlist\@empty\else%
797       \expandafter\gdef\csname l@\thm@currentlist\endcsname{\thm@thmline}%
798       \expandafter\gdef\csname thm@listdo\thm@currentlist\endcsname{\thm@inlistdo}%
799     \fi%
800   }%
801   \@input{\jobname .thm}%
802   \thm@thmlend%
803   \@for\thm@currentlist:=\thm@list\do{%
804     \ifx\thm@currentlist\@empty\else%
805       \expandafter\gdef\csname l@\thm@currentlist\endcsname{\thm@lgobble}%
806       \expandafter\gdef\csname thm@listdo\thm@currentlist\endcsname{\thm@lgobble}%
807     \fi%
808   }%
809   \endgroup}
```

`\thm@enablelistoftheorems` Up to now, we've set up various macros for writing and reading the theorem-file. Thus, it's time to set up the file itself. This is done by the next macro. We simply took the lines for `\starttoc` from the L^AT_EX-base and changed some things. The main intention to copy `\starttoc` is that we don't want the file to be input when it is set up – like it's done by `\starttoc`.

```
810 \def\thm@enablelistoftheorems{%
811   \begingroup%
812   \makeatletter%
813   \if@files%
814     \expandafter\newwrite\csname tf@thm\endcsname%
815     \immediate\openout \csname tf@thm\endcsname \jobname.thm\relax%
816   \fi%
817   \@nobeckfalse%
818   \endgroup}
```

`\addtheoremline` By `\addtheoremline{theorem-set}{entry}`, the user can insert an extra entry into the theorem-file. `\addtheoremline*` calls the internal macro `\nonum@addtheoremline`,

otherwise `\num@addtheorem` is called. `\num/nonum@addtheorem{<theorem-set>}{<entry>}` calls `\num/nonum@addtheorem{<theorem-set>}{<entry>}` which are defined when `<theorem-set>` is declared (cf. `\nthm` and `\@nthm`). `\turn` calls `\@num/nonum@addtheorem{<theorem-set>}{<keyword>}{<entry>}` which write information to the theorem file.

```
819 \def\addtheorem{\@ifstar{\nonum@addtheorem}{\num@addtheorem}}
820 \def\nonum@addtheorem#1{\csname nonum@addtheorem#1\endcsname}%
821 \def\num@addtheorem#1{\csname num@addtheorem#1\endcsname}%
```

`\@nonum@addtheorem` `\@num@addtheorem` and `\@nonum@addtheorem` write the actual entries to the `.thm` file.

Syntax: `\@num/nonum@addtheorem{<theorem-set>}{<keyword>}{<entry>}`

```
822 \def\@nonum@addtheorem#1#2#3{%
823   \thm@parseforwriting{#3}%
824   \edef\thm@t{#{#2}-}{\thm@tmp}}%
825   \addcontentsline{thm}{#1}{\thm@t}}
```

`\@num@addtheorem`

```
826 \def\@num@addtheorem#1#2#3{%
827   \thm@parseforwriting{#3}%
828   \edef\thm@t{#{#2}-\csname the#1\endcsname}{\thm@tmp}}%
829   \addcontentsline{thm}{#1}{\thm@t}}
```

`\addtotheoremfile` To write any additional stuff into the theorem-file, the next macro is used. It first checks, if the optional name of a theorem-set is given. In that case, the macro `\@addtotheoremfile`, otherwise `\@addtotheoremfile` is used to write the stuff into the file.

```
830 \long\def\addtotheoremfile{%
831   \@ifnextchar[{\@addtotheoremfile}{\@addtotheoremfile}}
```

`\@addtotheoremfile` Write additional stuff for all theorems.

```
832 \long\def\@addtotheoremfile#1{%
833   \thm@parseforwriting{#1}%
834   \protected@write\@auxout%
835     {}{\string\@writefile{thm}{\thm@tmp}}}
```

`\@addtotheoremfile` Write additional stuff for a given theorem-set.

```
836 \long\def\@addtotheoremfile[#1]#2{%
837   \thm@parseforwriting{#2}%
838   \protected@write\@auxout%
839     {}{\string\@writefile{thm}{\string\theoremlistdo{#1}{\thm@tmp}}}}
```

`\theoremlistdo` This one is called from the theorem-file to insert the additional stuff for a theorem-set.

```
840 \long\def\theoremlistdo#1#2{\csname thm@listdo#1\endcsname{#1}{#2}}
```

Now we assure, that the theorem-file is activated. This is done by inserting a hook at the end of the document.

```
841 \AtEndDocument{\thm@enablelistoftheorems}
```

7.1.9 Auxiliary macros

For generating theorem-lists, we need to write informations into a separate file. Beause we don't want to expand this information, we parse it specially for writing.

```
842 \def\thm@meaning#1->#2\relax{#2}% remove "macro: ->"
843 \long\def\thm@parseforwriting#1{%
844     \def\thm@tmp{#1}%
845     \edef\thm@tmp{\expandafter\thm@meaning\meaning\thm@tmp\relax}}
```

In some countries it's usual to number theorems with greek letters:

`\theorem@checkbold` For correctness, we need to check if a bold font is active. This is done by the following macro:

```
846 \def\theorem@checkbold{\if b\expandafter\@car\@series\@nil\boldmath\fi}
```

`\@greek` According to L^AT_EX-base, this is the internal command for generating lowercase greek numberings.

```
847 \def\@greek#1{\theorem@checkbold%
848 \ifcase#1\or$\alpha$\or$\beta$\or$\gamma$\or$\delta$\or$\varepsilon$%
849 \or$\zeta$\or$\eta$\or$\vartheta$\or$\iota$\or$\kappa$\or$\lambda$\or$%
850 \mu$\or$\nu$\or$\xi$\or$ o$\or$\varpi$\or$\varrho$\or$\varsigma$\or$\tau$%
851 \or$\upsilon$\or$\varphi$\or$\chi$\or$\psi$\or$\omega$\else\@ctrerr\fi}
```

`\@Greek` According to L^AT_EX-base, this is the internal command for generating uppercase greek numberings.

```
852 \def\@Greek#1{\theorem@checkbold%
853 \ifcase#1\or A\or B\or$\Gamma$\or$\Delta$\or E%
854 \or Z\or H\or$\Theta$\or I\or K\or$\Lambda$\or M%
855 \or N\or$\Xi$\or O\or$\Pi$\or P\or$\Sigma$\or T%
856 \or$\Upsilon$\or$\Phi$\or X\or$\Psi$\or$\Omega$\else\@ctrerr\fi}
```

`\greek` According to L^AT_EX-base, this is the user interface for lowercase greek numberings.

```
857 \def\greek#1{\@greek{\csname c@#1\endcsname}}
```

`\Greek` According to L^AT_EX-base, this is the user interface for uppercase greek numberings.

```
858 \def\Greek#1{\@Greek{\csname c@#1\endcsname}}
```

7.1.10 Other Things

After declaring several package-options, we need to process the specified ones. The additional `\relax` was mentioned by Rainer Schöpf at DANTE'97.

```
859 \ProcessOptions\relax
```

Now we set up the default theorem listtype. Make sure this is called after processing the options. Otherwise, `ntheorem` will break with `hyperref`.

```
860 \theoremlistall
```

If automatical configuration is not disabled by `[noconfig]`, it is checked if the file `ntheorem.cfg` exists and in this case the definitions in this file are read. If it does not exist and the option `standard` was specified, the file `ntheorem.std` is used.

```
861 \ifx\thm@noconfig\undefined
862 \InputIfFileExists{ntheorem.cfg}%
863 {\PackageInfo{\basename}{Local config file ntheorem.cfg used}}%
864 {\ifx\thm@usestd\undefined}
```

```

865 \else%
866 \InputIfFileExists{ntheorem.std}%
867 {\PackageInfo{\basename}{Standard config file ntheorem.std used}}{}
868 \fi}
869 \fi

```

7.2 The Standard Configuration

```

1 \theoremnumbering{arabic}
2 \theoremstyle{plain}
3 \RequirePackage{latexsym}
4 \theoremsymbol{\ensuremath{\_ \Box}}
5 \theorembodyfont{\itshape}
6 \theoremheaderfont{\normalfont\bfseries}
7 \theoremseparator{}
8 \newtheorem{Theorem}{Theorem}
9 \newtheorem{theorem}{Theorem}
10 \newtheorem{Satz}{Satz}
11 \newtheorem{satz}{Satz}
12 \newtheorem{Proposition}{Proposition}
13 \newtheorem{proposition}{Proposition}
14 \newtheorem{Lemma}{Lemma}
15 \newtheorem{lemma}{Lemma}
16 \newtheorem{Korollar}{Korollar}
17 \newtheorem{korollar}{Korollar}
18 \newtheorem{Corollary}{Corollary}
19 \newtheorem{corollary}{Corollary}
20
21 \theorembodyfont{\upshape}
22 \newtheorem{Example}{Example}
23 \newtheorem{example}{Example}
24 \newtheorem{Beispiel}{Beispiel}
25 \newtheorem{beispiel}{Beispiel}
26 \newtheorem{Bemerkung}{Bemerkung}
27 \newtheorem{bemerkung}{Bemerkung}
28 \newtheorem{Anmerkung}{Anmerkung}
29 \newtheorem{anmerkung}{Anmerkung}
30 \newtheorem{Remark}{Remark}
31 \newtheorem{remark}{Remark}
32 \newtheorem{Definition}{Definition}
33 \newtheorem{definition}{Definition}
34
35 \theoremstyle{nonumberplain}
36 \theoremheaderfont{\scshape}
37 \theorembodyfont{\normalfont}
38 \theoremsymbol{\ensuremath{\_ \blacksquare}}
39 \RequirePackage{amssymb}
40 \newtheorem{Proof}{Proof}
41 \newtheorem{proof}{Proof}
42 \newtheorem{Beweis}{Beweis}
43 \newtheorem{beweis}{Beweis}
44 \qedsymbol{\ensuremath{\_ \blacksquare}}
45 \theoremclass{LaTeX}

```

8 History and Acknowledgements

8.1 The endmark-Story (Wolfgang May)

In 1995, I started a hack for setting endmarks semiautomatically at the end of displayed formulas. The work on `thmmarks.sty` was initiated in October 1996 by a thread asking for a routine for setting endmarks in *de.comp.tex* founded by Boris Piwinger. Version 0.1 incorporated the main features for setting endmarks automatically by using the `.aux` file. Version 0.2 included some bugfixes and was the first one accessible on the internet. Boris suggested to include `fleqn` and `leqno` which has been done in version 0.3 (which was never made public). Since at this point, `thmmarks.sty` was incompatible to the widely used `theorem.sty` written by Frank Mittelbach, in Version 0.4, the features of `theorem.sty` have been integrated. To version 0.5, the case of “empty” end symbols has been handled, `\qed` has been added (also suggested by Boris), and the handling of theoremstyles by `\newtheoremstyle` has been included.

For version 0.6, the handling of endmarks in `displaymaths` has been changed in order to adjust them with the bottom of the displayed math.

Version 0.6 was the first one announced in *comp.text.tex*. For version 0.7, I added the handling of `amsmath` features, suggested by my colleague Peter Neuhaus.

Versions 0.71 and 0.72 incorporated minor bugfixes.

8.2 Lists, Lists, Lists (Andreas Schlechte)

I often saw questions on `theoremlists` in the german newsgroup *de.comp.text.tex*, but I never spent any attention on those postings. This changed in summer 1996, when I needed those lists for myself. Thus, I asked the holy question. But none of the given answers satisfied my wish for a simple, easy to use and short solution.

I decided to take a look at Frank Mittelbachs `theorem.sty`. First I didn't understand much of the code, but Bernd Raichle helped me a lot by answering my boring questions and I finally understood it.

I started the coding and within a few days, a first experimental version was born. Not only that I had implemented the lists, I also inserted a separator and a flexible numbering of the theorems.

After a long period of testing, I wanted to share the new features with other `TeX`-Freaks and wrote an article for the “Die `TeX`nische Komödie” (Journal of german tug, DANTE e.V.). As soon as I had sent the article to DANTE, I got first reactions on the style. Gerd Neugebauer gave me many hints. I hid several cryptical notations in easy definitions and improved the user interface.

In January 1997, I released “`newthm`” to the world and it was uploaded to the CTAN-Archives. Few days later I sent my files to Frank Mittelbach in order to show him my extensions. He told me, that already other extensions were made, and that it would be good to combine altogether.

8.3 Let's come together

With version 0.8, in February 1997, the combination of `thmmarks.sty` with `newthm.sty` to `ntheorem.sty` has been started. On April 21, 1997, version 0.94 beta has been made public as version 1.0.

In course of the development, the following changes were made:

You should create the list of changes by

```
makeindex -s gglo.ist -o ntheorem.gls ntheorem.glo  
and running latex ntheorem.drv again.
```

8.4 Acknowledgements

This place is dedicated to all those, who helped us developing our separate styles and this combined package. Thanks to (listed in alphabetical order):

Frank Mittelbach, Gerd Neugebauer, Boris Piwinger, Bernd Raichle,
Rainer Schöpf, Didier Verna.