An Extension of the LaTeX-Theorem Evironment*

Wolfgang May‡
Institut für Informatik,
Universität Göttingen
Germany

Andreas Schedler§

2011/08/15

Abstract

`ntheorem.sty` is a package for handling theorem-like environments. Aditionally 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 LaTeX 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 LaTeX `\newtheorem` command, existing documents can be switched to `ntheorem.sty` without having to change the `.tex` file. Also, it is compatible with LaTeX files using `theorem.sty` written by Frank Mittelbach.

*This file has version number 1.33, last revised 2011/08/15.

‡`may@informatik.uni-goettingen.de`

§`ntheorem@andreas-schedler.net`

Contents

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 developped: 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}`,

where the optional parameter ⟨*options*⟩ selects predefined configurations and special requirements.
The following ⟨*options*⟩ are available by now, concerning partially independent issues:

Predefined 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.

Fancy boxes around theorems: The [framed] option allows to use `framed.sty` that provides boxes even across pagebreaks.

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); when using the `amsmath`-package, [thref] must be complemented by [amsmath] (see Section 3.2).

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

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

The package itself loads `ifthen.sty`.

2.2 Defining New Theorem Sets

`\newtheorem` The syntax and semantics is exactly the same as in standard LaTeX: 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 ⟨*name*⟩ defined by `\newtheorem`, *two* enviroments ⟨*name*⟩ and ⟨*name**⟩ 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 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 2.3.3. Additional styles can be defined by `\newtheoremstyle` (cf. Section2.5.1).

`\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*⟩.

`\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.

`\theoremseparator`
- `\theoremseparator{`⟨*thing*⟩`}`: ⟨*thing*⟩ separates the header from the body of the theorem-environment. E.g., ⟨*thing*⟩ can be ":" or ".".

\theorempreskip
\theorempostskip

- `\theorempreskip{`⟨*skip*⟩`}` and `\theorempostskip{`⟨*skip*⟩`}` can be used to specify the vertical space before/after the theorem environment (note that Section 4.2 that allows framed and shaded theorems also defines additional skip parameters). The arguments are rubber lengths, ('skips'), and therefore can contain `plus` and `minus` parts. (Note that these parameters changed with version 1.32; see Section 2.3.5.)

\theoremindent

- `\theoremindent`⟨*dimen*⟩ can be used to indent the theorem wrt. the surrounding text (note that `\theoremindent` is specified without `{...}`).

  ! It's a '(dimen)', so the user shouldn't try to specify a `plus` or `minus` part, because this leads to an error.

\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 above properties carry over to all subsequent `\newtheorem` statements until they are set differently. Initially, they have default values.
The following two properties apply only to the very next `\newtheorem` and are then automatically reset:

\theoremprework

- `\theoremprework{`⟨*thing*⟩`}`: ⟨*thing*⟩ is performed before starting the theorem structure. E.g., ⟨*thing*⟩ can be `\bigskip\hrule\leavevmode`. If the vertical space after your theoremprework does not look as intended, try to put `\leavevmode` at its end (as in the above example).

\theorempostwork

- `\theorempostwork{`⟨*thing*⟩`}`: ⟨*thing*⟩ is performed after finishing the theorem structure. E.g., ⟨*thing*⟩ can be `\hrule`.

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 LaTeX layout can be chosen.

2.3.2 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}`

2.3.3 Predefined theorem styles

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

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

2.3.4 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{}`,
> `\theorempreskip{\topsep}`,
> `\theorempostskip{\topsep}`, where `\topsep` refers to the space that LaTeX inserts above and below lists,
> `\theoremindent0cm`,
> `\theoremnumbering{arabic}`,
> `\theoremsymbol{}`.

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

2.3.5 Deprecated: Skips until Version 1.32

`\theorempreskipamount`
`\theorempostskipamount`

Until version 1.31, there was only a simplified handling of vertical space before/after theorems that did not consider framed and shaded theorems (that have been introduced with v1.21). `\theorempreskipamount`⟨*skip*⟩ and `\theorempostskipamount`⟨*skip*⟩ defined, respectively, the spacing before and after such an environment (note that both are specified without `{...}`). These parameters applied 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.
Unchanged, older LaTeX sources that used these commands yield the same output as before since the new skip scheme described in Section 2.3.1 is only activated if one of its commands is used. Otherwise, the old scheme is applied.

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 environments[1]. 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 Framed and Boxed Theorems

With the advent of the `framed` package (by Donald Arseneau) in 2001, a feature that has often been asked for for ntheorem could be implemented: theorems that are framed, or that are put into a colored box. It requires to load the `framed` package; shaded theorems also require the `pstricks` package. Frames and colored boxes are orthogonal to the existing theoremstyles – thus, they can be combined in arbitrary ways.

`\newframedtheorem` A theorem type can be framed by defining it by

```
\newframedtheorem{...}{...}
```

with the same parameters as usually for `\newtheorem`. Note that the use of the `framed` package also allows to have longer theorems across a page break framed (in this case, by default, there are horizontal lines before and after the page break; this can even be circumvented by combining with `mdframed` package (since 2010)).

`\newshadedtheorem`

The same ideas hold for theorems in shaded boxes. The declaration

```
\newshadedtheorem{...}{...}
```

declares a theorem environment that is shaded. By default, the background color is `gray`. This can be changed by defining

`\shadecolor{`⟨*color*⟩`}`

before declaring the theorem type. Note that later declarations of other shaded theorem types can use another shadecolor.
By default, the box is given as a `\psframebox` (see `pstricks` package) with shadecolor as `linecolor` and `fillcolor`. All these parameters can be changed by setting

`\def\theoremframecommand{`⟨*any box command*⟩`}`

before declaring the theorem type (for examples, the user is referred to section 4). For using pdflatex (where pstricks is not available), e.g. `\usepackage{color}` and `\theoremframecommand{\colorbox[rgb]{1,.9,.9}}` can be used.

Vertical Spacing of Framed Theorems The New Skip Scheme introduced with version 1.32 allows a detailed specification of vertical space also for framed theorems (specified individually for each theorem class):

- `\theorempreskip{`⟨*skip*⟩`}` and `\theorempostskip{`⟨*skip*⟩`}` have no effect for framed theorems.

`\theoremframepreskip`
`\theoremframepostskip`

- `\theoremframepreskip{`⟨*skip*⟩`}` and `\theoremframepostskip{`⟨*skip*⟩`}` can be used to specify the vertical space before/after the frame/box.

`\theoreminframepreskip`
`\theoreminframepostskip`

- `\theoreminframepreskip{`⟨*skip*⟩`}` and `\theoreminframepostskip{`⟨*skip*⟩`}` can be used to specify the vertical space around the theorem text *inside* the frame/box.
- The arguments of the above commands are rubber lengths, ('skips'), and therefore can contain `plus` and `minus` parts.
- the default values of all above skips is `\topsep`, i.e., the space LaTeX normally inserts before/after lists.

Old Skip Scheme (until v 1.31): `\theorempreskipamount` and `\theorempostskipamount` are applied *inside* the frame/box. To obtain vertical space *before* and *after* the frame/box in versions 1.30–v.1.31, `\theoremframepreskipamount` and `\theoremframepostskipamount` could be used (both defined by default to 0pt) analogously (i.e., they are also common to all theorem types.)

[1]Note, that mathmode is ensured for the symbol.

2.3.8 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}`.
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.
Note that in contrast to similar LATEX commands like `\listoffigures` etc. there is no automatically created heading. Users have to write it themselves – but are free to choose what they want to have.

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 the 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 LaTeX `\addcontentsline` and `\addtocontents` cannot be used in a straightforward way[2].

`\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

```
\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 ⟨*name*⟩ is the name of a theorem set and ⟨*text*⟩ is the text to be written into the theorem file. If the optional argument ⟨*name*⟩ 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, ⟨*head*⟩ has to be a statement using two arguments, `##1`, containing the keyword, and `##2`, containing the number. ⟨*opt-head*⟩ has to be a statement using three arguments where the additional argument `##3` contains the optional parameter.
Since LaTeX 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, ⟨*name*⟩, is the name of the listtype, which can the be used as a valid `\theoremlisttype`. ⟨*start*⟩ is the sequence of commands to be executed at the very beginning of the list. Corresponding, ⟨*end*⟩ will be executed at the end of the list. These two are set to do nothing in the standard-types. ⟨*line*⟩ 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

[2]for 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.

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 he 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]`.

Often, when writing a paper, one changes propositions into theorems, theorems into corollaries, lemmata into remarks an 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 LaTeX 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`. It also interferes with `amsmath`; see Section 3.2.

2.8 Miscellaneous

Inside a theorem-like environment ⟨*env*⟩, the name given as optional argument is accessible by \⟨*env*⟩`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` isa 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` (LaTeX2.09) is obsolete and you should use `amsmath` and `amstext` for LaTeX 2_ε instead. Up to `ntheorem-1.18`, it is compatible with `amsmath-1.x`. Since `ntheorem-1.19`, it is (hopefully) compatible with `amsmath-2.x`.
We would be happy if someone knowing and using `amsmath` would join the development and maintenance of this style.

3.2.1 amsmath

Compatibility with amsmath (end marks for math environments, and handling of labels in math environments) is provided in the option `[amsmath]`, (i.e., if `\usepackage{amsmath}` is used then

[3]although 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.

- \usepackage[thmmarks]{ntheorem} must be completed to \usepackage[amsmath,thmmarks]{ntheorem}), and also
- \usepackage[thref]{ntheorem} must be completed to \usepackage[amsmath,thref]{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}.
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 nonumbered 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 LaTeX \contentsline-command, it breaks with ntheorem below version 1.17. Since version 1.17, the option [hyperref] makes ntheorem work with hyperref. The entries of theoremlists then act as hyperlinks to the actual theorems. Version 1.31 incorporated some bugfixes wrt. hyperref for theorem lists and for the thref option. One should always load \usepackage{hyperref} *before* the first use of \newtheorem to obtain correct handling and referencing of counters.

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

3.5 Mathtools and Empheq

The `mathtools` and `empheq` packages should be loaded *before* `ntheorem` as follows:

```
\usepackage[ntheorem]{empheq} % this loads amsmath as well
\usepackage[thmmarks,amsmath]{ntheorem}
```

Note that `empheq` provides an enhanced vertical placement of the endmarks (see the paragraph on `ntheorem` in the documentation of the `empheq` package) in math environments.

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{.}
\theoremprework{\bigskip\hrule}
\theorempostwork{\hrule\bigskip}
\newtheorem{Definition}{Definition}
```

- For Proofs (note that `\theoremprework` and `\theorempostwork` are automatically reset with the next `\newtheorem` – proofs do not have lines above and below):

```
\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:

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

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:

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

Just try to get this with the presented array structure ... without using dirty tricks, you can position the outer array either [t], [c], or [b], and you will not get the desired effect.

$$a = \begin{array}[t]{l} first\ line \\ second\ line \end{array} \quad \text{try to put this text in the lowest line} \qquad \heartsuit$$

β Lemma (Equation):
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) ... If anyone knows, please inform us.*

$$\int_\gamma f(z)\,dz := \int_a^b f(\gamma(t))\gamma'(t)\,dt \tag{1}$$ $\heartsuit$

With the `break`*-theoremstyles, if the environment is labeled and written as*

```
\begin{Lemma}[Breakstyle]\label{breakstyle}
```

γ Lemma (Breakstyle):
you see, there is a leading space . . .
If a percent (comment) (or an explicit `\ignorespaces`*) is put directly after the label, e.g.*

```
\begin{Lemma}[Breakstyle]\label{breakstyle}%,
```

the space disappears.
From the predefined styles, this is exactly the case for the break-styles. That's no bug, it's LaTeX-immanent.
The example goes on with an `eqnarray`*:*

$$\begin{aligned} f(z) &= \frac{1}{2\pi i}\int\limits_{\partial D} \frac{f(\zeta)}{\zeta - z} d\zeta && (2) \\ &= \frac{1}{2\pi}\int\limits_0^{2\pi} f(z_0 + re^{it})dt && (3) \end{aligned}$$ $\heartsuit$

Proof (of nothing)

$$\begin{aligned} f(z) &= \frac{1}{2\pi i}\int\limits_{\partial D} \frac{f(\zeta)}{\zeta - z} d\zeta \\ &= \frac{1}{2\pi}\int\limits_0^{2\pi} f(z_0 + re^{it})dt \end{aligned}$$ $\blacksquare$

That's it (the end of the Theorem). $\diamondsuit$

If there are some environments in the same thm-environment, the last one gets the endmark:

Definition 1 (With a list).

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

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

---

2 Corollary (Q.E.D.):
And here is a trivial corollary, which is ended by `\qedsymbol{\textrm{q.e.d}}` and `\qed`. q.e.d

3 Example

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

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 LaTeX from using its nested list management (a verbatim is also a trivlist), i.e. this causes LaTeX 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):
let's do something weird:

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. $\diamondsuit$

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. $\diamondsuit$

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 Theorems and is numbered together with Corollaries (Theorems 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. ${a \atop b}$

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 20) 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 Framed and Shaded Theorems

Framed theorem classes are defined as follows:

```
\theoremclass{Theorem}
\theoremstyle{break}
\newframedtheorem{importantTheorem}[Theorem]{Theorem}
```

defines important theorems to use the same design as for theorems (except that the break header style is used except the margin header style), number them with the same counter, and put a frame around them:
An instance is created by

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

```
\begin{importantTheorem}[Important Theorem]
This is an important theorem.
\end{importantTheorem}
```

Theorem 9 (Important Theorem):
This is an important theorem. ◇

Note that all skips have their default values (e.g. `\theoreminframepreskip` is `\topsep`). More important theorems are shaded – by default in grey:

```
\theoremclass{Theorem}
\theoremstyle{break}
\newshadedtheorem{moreImportantTheorem}[Theorem]{Theorem}
\begin{moreImportantTheorem}[More Important Theorem]
This is a more important theorem.
\end{moreImportantTheorem}
```

Theorem 10 (More Important Theorem):
This is a more important theorem. ◇

Even more important theorems are shaded in red, with 1/2cm space inside the frame before and 1 cm space after the text, but no additional space before/after the frame:

```
\theoremclass{Theorem}
\theoremstyle{break}
\theoreminframepreskip{0.5cm}
\theoreminframepostskip{1cm}
\theoremframepreskip{0cm}
\theoremframepostskip{0cm}
\shadecolor{red}
\newshadedtheorem{evenMoreImportantTheorem}[Theorem]{Theorem}
\begin{evenMoreImportantTheorem}[Even More Important Theorem]
This is an even  more important theorem.
\end{evenMoreImportantTheorem}
```

Theorem 11 (Even More Important Theorem):
This is an even more important theorem. ◇

Most important theorems get a framed, blue colored box with a shadow, no space inside the frame, and 1cm before and after the frame. Here, `\def\theoremframecommand` is used:

```
\theoremclass{Theorem}
\theoremstyle{break}
\theoreminframepreskip{0pt}
\theoreminframepostskip{0pt}
\theoremframepreskip{1cm}
\theoremframepostskip{1cm}
\theoremstyle{break}
\def\theoremframecommand{%
      \psshadowbox[fillstyle=solid,fillcolor=blue,linecolor=black]}
\newshadedtheorem{MostImportantTheorem}[Theorem]{Theorem}
\begin{MostImportantTheorem}[Most Important Theorem]
This is a most important theorem.
\end{MostImportantTheorem}
```

Theorem 12 (Most Important Theorem):
This is a most important theorem. ◇

4.3 Lists of Theorems and Friends

Note, that we put the following lists into the `quote`-environment to emphazise 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:

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

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

leads to

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}{@{}lrl@{\extracolsep{\fill}}r@{}}}%
  {##1&##2&##3&##4\\}%
  {\end{tabular*}}
```

Thus, by saying

```
\theoremlisttype{tab}
\listtheorems{Theorem,importantTheorem,moreImportantTheorem,
              evenMoreImportantTheorem,MostImportantTheorem,Lemma},
```

theorems (of all importance levels) and lemmata are listed:

Theorem	1	Long Theorem	17
Lemma	α	Display with array	17
Lemma	β	Equation	18
Lemma	γ	Breakstyle	18
Theorem	3		20
Theorem	5	Quote	20
Theorem	6	Manual End Mark	20
Lemma	4		21
Theorem	7	somename	21
Theorem	9	Important Theorem	23
Theorem	10	More Important Theorem	23
Theorem	11	Even More Important Theorem	23
Theorem	12	Most Important Theorem	24
Theorem	13	Correctness	28
Theorem	14	Completeness	28
Lemma	5		29
Lemma	6		29
Theorem	15		30

LaTeX-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. 19

4 Corollary: title in list 20

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 endmarks 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 is 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 ⟨*env*⟩ be a theorem-like environment. Then, additional to the counter ⟨*env*⟩, TEX maintains two counters `curr`⟨*env*⟩`ctr` and `end`⟨*env*⟩`ctr`. In the ith environment of type ⟨*env*⟩, `curr`⟨*env*⟩`ctr`$= i$ (the LATEX counter ⟨*env*⟩ cannot be used since a) environments can use the counter of other environments, and b) often counters are reinitialized inside a document). `end`⟨*env*⟩`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

```
\mark<\thm@romannum{currenvctr}>⟨env⟩<\thm@romannum{end⟨env⟩ctr}>
```

is called (where `\thm@romannum` just writes the value of a counter as its roman numeral representation, e.g., 17 as xvii).
(<\thm@romannum{curr⟨*env*⟩ctr}>⟨*env*⟩<\thm@romannum{end⟨*env*⟩ctr}> uniquely identifies all situations (2)–(4) in a document).
If at this position an endmark has to be set,

```
\mark<\thm@romannum{curr⟨env⟩ctr}>⟨env⟩<\thm@romannum{end⟨env⟩ctr}>
```

is defined in the `.aux` file to be `\end`⟨*env*⟩`Symbol`, otherwise it is undefined and simply ignored.
When TEX comes to an `\end{`⟨*env*⟩`}`, 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<\thm@romannum{currenvctr}>⟨env⟩%
<\thm@romannum{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.

13 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 ith situation in the jth 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 the kth* `\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.* ♢

14 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`*.* ♢

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 LaTeX: 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:

  15 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` $\neq 0$. 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

```
\typeout{Style ‘\basename’, Version \fileversion\space <\filedate>}
\ProvidesPackage{ntheorem}[\filedate \space\fileversion]
\RequirePackage{ifthen}%
```

```
\newif\if@thmmarks\@thmmarksfalse
\newif\if@thref\@threffalse
\newif\ifthm@inframe\thm@inframefalse
\newif\ifthm@tempif
```

general setup.

7.1.1 Thmmarks-Related Stuff

```
\DeclareOption{thmmarks}{%*********************************
\PackageInfo{\basename}{Option ‘thmmarks’ loaded}%
%
\@thmmarkstrue
\newcounter{endNonectr}
\newcounter{currNonectr}
\newif\ifsetendmark\setendmarktrue
```

activate placement of endmarks and define counters for upper level.
`\ifsetendmark`: true if an endmark has to be set in a complex situation which must be handled by the `.aux` file. For further comments see `\@endtheorem`.

`\thm@romannum` The functionality of `latex.ltx`'s `\roman` command converts numbers into strings, e.g., 17 into xvii. It is used to put notes into the `.aux` file. It must be locally defined, just duplicating the definition of `\roman` in latex.ltx since some packages redefine `\roman`:

```
\gdef\thm@romannum#1{\expandafter\thm@roman@num\csname c@#1\endcsname}%
\gdef\thm@roman@num#1{\romannumeral #1}%
```

In the following, all relevant environments are changed for handling potential end mark positions:

Changes to List Environment
Original: ltlists.dtx

`\endtrivlist` Replaces LaTeX's `\endtrivlist`. An augmented functionality of LaTeX's `\endtrivlist` is contained in `\@endtrivlist`.

```
\gdef\endtrivlist{%
  \@endtrivlist{\PotEndMark{\unskip\nobreak\hfill\nobreak}}}
```

At an `\endtrivlist` (which is called at the end of `\list` environments and several other environments), `\@endtrivlist` is called to end the `\trivlist` and set a potential position for an endmark at the end of the line if TeX is in a text line.

`\@endtrivlist` A new command] which augments LaTeX's functionality of `\endtrivlist` by checking if an end mark has to be set:

```
\gdef\@endtrivlist#1{%  % from \endtrivlist
  \if@inlabel \indent\fi
  \if@newlist \@noitemerr\fi
  \ifhmode
     \ifdim\lastskip >\z@ #1\unskip \par  %<<<<<<<<<<<<<<<<<<<<<<
           \else \unskip \par \fi
```

```
18     \fi
19  \if@noparlist \else
20    \ifdim\lastskip >\z@
21       \@tempskipa\lastskip \vskip -\lastskip
22      \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
23      \vskip\@tempskipa
24    \fi
25    \@endparenv
26  \fi}
```

New: parameter `#1`.
`#1` is executed when the `\trivlist` ends with a text line (ie the endmark can be put simply at the end of the line):
Line 16: case split: if in `hmode` and `\lastskip` > 0, then TEX is in a text line, the endmark is set here.

Changes to Math Environments
Original: ltmath.dtx

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

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

Line 27: As default, work for equation numbers at the right: Then, a `\quad` is placed between equation number and endmark.

Line 28: 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:

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

Lines 31–37, 45, 46: the old definition.

Lines 38–41: 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<\thm@romannum{curr#1ctr}>#1<\thm@romannum{end#1ctr}+1>`

is defined and not the empty symbol.

Lines 42–43: If so, the whole displayed stuff is put in an array with maximal depth 0.5`ex` and vertically adjusted with its bottom line (then, the endmarks will appear adjusted to its bottom line).

Line 44: The counter has to be re-decremented.

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

```
47 \gdef\]{%
48       \stepcounter{end\InTheoType ctr}%
49         \@ifundefined{mark\thm@romannum{curr\InTheoType ctr}%
50                        \InTheoType\thm@romannum{end\InTheoType ctr}}{\relax}%
51           {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
52               \end{array}\fi}%
53         \addtocounter{end\InTheoType ctr}{-1}%
54    \relax\ifmmode
55       \ifinner
56          \@badmath
57       \else
58           \PotEndMark{\eqno}\global\@ignoretrue$$%%$$ BRACE MATCH HACK
59       \fi
60    \else
61       \@badmath
62    \fi
63    \ignorespaces}
```

Lines 48–53: Look, if an endmark has to be set in this displaymath (analogous to lines 38–44 of `\def\[`) If so, there is an inner array which has to be closed (line 52).

Lines 54–63: the old definition.

Line 58: changed to set an endmark at the right of the line if necessary (this is done by `\eqno`).

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

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

Line 64: 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 66, 67, 72:

Line 66: save \@eqnnum.

Line 67: 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 68–71: from latex.ltx. Line 68 sets the equation number.

Line 72: restore \@eqnnum.

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

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

This is just LaTeX's \endeqnarray where lines 75–81 are inserted from \@@eqncr and augmented (line 80) 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 \`).

```
86 \gdef\endtabbing{%
87   \PotEndMark{\`}\@stopline\ifnum\@tabpush >\z@ \@badpoptabs
88   \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):

```
89 \gdef\endcenter{%
```

```
\@endtrivlist
  {\PotEndMark{\rightskip0pt%
       \settowidth{\leftskip}%
        { \csname mark\thm@romannum{curr\InTheoType ctr}\InTheoType
                 \thm@romannum{end\InTheoType ctr}\endcsname}%
       \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-thmmarks`

`\@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):

```
96 \gdef\@empty{}
97 \gdef\@endtheorem{%
98   \expandafter
99   \ifx\csname\InTheoType Symbol\endcsname\@empty\setendmarkfalse\fi
100  \@endtrivlist
101    {\ifsetendmark
102     \unskip\nobreak\hfill\nobreak\csname\InTheoType Symbol\endcsname
103     \setendmarkfalse \fi}%
104  \ifsetendmark\OrganizeTheoremSymbol\else\global\setendmarktrue\fi
105  \csname\InTheoType @postwork\endcsname
106  }
```

Lines 98, 99: 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 100, 104: (originally, it calls `\endtrivlist`):

Lines 100, 102, 103: `\@endtrivlist` is called to put ⟨*env*⟩`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 101: the endmark is not set, if `setendmark` is false.

Line 104: 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 102 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.

```
107 \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:

```
108 \gdef\qed{\expandafter\def\csname \InTheoType Symbol\endcsname
109          {\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:

```
110 \gdef\PotEndMark#1{
111   \@ifnextchar[%]
112     {\PotEndMark@opt{#1}}{\SetEndMark{\InTheoType}{#1}{\relax}}}%
113 \gdef\PotEndMark@opt#1[#2]{\SetEndMark{\InTheoType}{#1}{#2}}%
```

Arguments: ⟨*cmd_seq*⟩:=`#1` is a command sequence to be executed when setting the endmark.
⟨*else_cmd_seq*⟩= `#2`: a command sequence that is executed when no end mark is set (default is `\relax`; differs only in amsmath equation*).
It adds the current theorem type ⟨*env*⟩ to the parameters, and calls
`\SetEndMark{`⟨*env*⟩`}{`⟨*cmd_seq*⟩`}{`⟨*else_cmd_seq*⟩`}`.

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

```
114 \gdef\SetEndMark#1#2#3{%
115   \stepcounter{end#1ctr}%
116   \@ifundefined{mark\thm@romannum{curr#1ctr}#1\thm@romannum{end#1ctr}}%
117   {#3}%
118   {#2{\csname mark\thm@romannum{curr#1ctr}#1\thm@romannum{end#1ctr}\endcsname
119       \ifdim\rightmargin>\z@\hskip-\rightmargin\fi
120       \hbox to 0cm{}}}}%
```

Arguments:
⟨*env*⟩:=`#1`: current theorem-environment.
⟨*cmd_seq*⟩:= `#2`: is a command sequence to be executed when setting the endmark.
⟨*else_cmd_seq*⟩= `#3`: a command sequence that is executed when no end mark is set (usually `\relax`; differs only in amsmath tags).
All three arguments are transmitted by `\PotEndMark`.

Line 115: increments `end`⟨*env*⟩`ctr` for preparing the next situation for setting a potential endmark.

Line 116, 117: if

```
\mark<\thm@romannum{curr⟨env⟩ctr}>⟨env⟩<\thm@romannum{end⟨env⟩ctr}>
```

is undefined – which is the case iff at this position no endmark has to be set –, ⟨*else_cmd_seq*⟩ is executed,

Line 118: otherwise, ⟨*cmd_seq*⟩ and then

`\mark<\thm@romannum{curr⟨env⟩ctr}>\env <\thm@romannum{end⟨env⟩ctr}>`,

which is defined in the `.aux` file to be the end symbol are called.
The construction ⟨*cmd_seq*⟩`{...}` in line 118 allows the handling of the end symbol as an argument of ⟨*cmd_seq*⟩ as needed for `\endeqnarray`.

Line 119: 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).
(applied only if `\rightmargin` is more than 0 – otherwise bug if preceding line ends with hyphenation.)

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

```
121 \newskip\mysavskip
122 \gdef\@bbsphack{%
123     \ifvmode\else\mysavskip\lastskip
124     \unskip\fi}
125 %
126 \gdef\@eesphack{%
127     \ifdim\mysavskip>\z@
128     \vskip\mysavskip \else\fi}
```

Lines 122–124 and 125–127 are similar to `\@bsphack` and `\@bsphack` of latex.ltx. They undo resp. redo the last `skip`.

Note that `@bbsphack` and `@eesphack` are also part of the thref option. Change both if you change them.

`\OrganizeTheoremSymbol`

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

```
129 \gdef\OrganizeTheoremSymbol{%
130   \@bbsphack
131   \edef\thm@tmp{\expandafter\expandafter\expandafter\thm@meaning
132          \expandafter\meaning\csname\InTheoType Symbol\endcsname\relax}%
133   \protected@write\@auxout{}%
134      {\string\global\string\def\string\mark%
135       \thm@romannum{curr\InTheoType ctr}\InTheoType \thm@romannum{end\InTheoType ctr}%
136       {\thm@tmp}}%
137   \@eesphack}
```

Lines 133–135: Write
`\global\def\mark<\thm@romannum{curr⟨env⟩ctr}> ⟨env⟩ <\thm@romannum{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.

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

7.1.2 Option leqno to Thmmarks

```
139 \DeclareOption{leqno}{% *********************************************
140   \if@thmmarks
141   \PackageInfo{\basename}{Option ‘leqno’ loaded}%
142    \gdef\SetMark@endeqn#1{\hss\llap{#1}}
143    \gdef\SetMark@endeqnarray#1{\hss\llap{#1}}
144   \fi}%
```

`leqno` is only active it `thmmarks` is also active.

Line 142, 143: 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

```
145 \DeclareOption{fleqn}{% *********************************************
146 \if@thmmarks
147  \PackageInfo{\basename}{Option ‘fleqn’ loaded}%
```

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

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

```
148  \renewcommand\[{\relax
149     \ifmmode\@badmath
150     \else
151      \begin{trivlist}%
152         \@beginparpenalty\predisplaypenalty
153         \@endparpenalty\postdisplaypenalty
154         \item[]\leavevmode
155         \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
156         \hskip\mathindent\bgroup
157           \stepcounter{end\InTheoType ctr}%
158           \@ifundefined{mark\thm@romannum{curr\InTheoType ctr}%
159                         \InTheoType\thm@romannum{end\InTheoType ctr}}{\relax}%
160             {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
161               \boxmaxdepth=.5ex\begin{array}[b]{@{}l}%
162               \boxmaxdepth=\maxdimen\displaystyle\fi}%
163           \addtocounter{end\InTheoType ctr}{-1}%
164     \fi}
```

Lines 148–156, 164: the old definition.

Line 157–163: if an endmark has to be set in this displaymath, it is put into an array with depth ≤ 0.5`ex`, and vertically adjusted to the bottom line.

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

```
165  \renewcommand\]{%
166     \stepcounter{end\InTheoType ctr}%
167     \@ifundefined{mark\thm@romannum{curr\InTheoType ctr}%
168                   \InTheoType\thm@romannum{end\InTheoType ctr}}{\relax}%
```

```
169       {\ifx\csname\InTheoType Symbol\endcsname\@empty\else
170           \end{array}\fi}%
171     \addtocounter{end\InTheoType ctr}{-1}%
172     \relax\ifmmode
173                   \egroup $\hfil\PotEndMark{}% $
174                 \egroup
175               \end{trivlist}%
176             \else \@badmath
177             \fi}
```

Lines 166–170: Look, if an endmark has to be set in this displaymath. If so, close the inner array.

Lines 172–177: the old definition.

Line 173: Added `\PotEndMark`.

`\endequation`

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

```
178 \gdef\endequation{%
179      $\hfil % $
180          \displaywidth\linewidth\hbox{\@eqnnum \PotEndMark{\SetMark@endeqn}}%
181        \egroup
182       \endtrivlist}
```

Line 180: 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 184, 185, 190):

```
183 \gdef\endeqnarray{%
184     \global\let\Oldeqnnum=\@eqnnum
185     \gdef\@eqnnum{\Oldeqnnum\PotEndMark{\SetMark@endeqnarray}}%
186     \@@eqncr
187     \egroup
188     \global\advance\c@equation\m@ne$$% $$
189     \global\@ignoretrue
190     \global\let\@eqnnum\Oldeqnnum}

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

7.1.4 Extended Referencing Facilities

```
192 \DeclareOption{thref}{%*********************************************
193   \PackageInfo{\basename}{Option ‘thref’ loaded}%
194 \@threftrue
```

Option `thref` needs a special handling when combined with amsmath. This is also a reason why it is handled first.

`\bbsphack(2)`

```
195 \newskip\mysavskip
196 \gdef\@bbsphack{%
197     \ifvmode\else\mysavskip\lastskip
```

```
    \unskip\fi}
%
\gdef\@eesphack{%
    \ifdim\mysavskip>\z@
    \vskip\mysavskip \else\fi}
```

Note that `@bbsphack` and `@eesphack` are also part of the thmmarks option. Change both if you change them.

Communication of theorem types for references. The `thref` functionality needs to know the respective theorem type of the referenced labels. This is incorporated as additional arguments in `label` and `newlabel/@newl@abel`. Note that if the `hyperref` package is used, the handling is different (see Option `hyperref`).

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

```
\def\label#1{%
  \@ifnextchar[%]
     {\label@optarg{#1}}%
     {\thm@makelabel{#1}}}
%
\def\thm@makelabel#1{%
  \@bbsphack
  \edef\thm@tmp{\expandafter\expandafter\expandafter\thm@meaning
          \expandafter\meaning\csname\InTheoType Keyword\endcsname\relax}%
  \protected@write\@auxout{}%
     {\string\newlabel{#1}{{\@currentlabel}{\thepage}}[\thm@tmp]}%
  \@eesphack}
%
\def\label@optarg#1[#2]{%
  \@bsphack
  \protected@write\@auxout{}%
     {\string\newlabel{#1}{{\@currentlabel}{\thepage}}[#2]}%
  \@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:

```
\def\@newl@bel#1#2#3{%
  \@ifpackageloaded{babel}{\@safe@activestrue}\relax%
  \@ifundefined{#1@#2}%
```

```
224     \relax
225     {\gdef \@multiplelabels {%
226        \@latex@warning@no@line{There were multiply-defined labels}}%
227      \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
228   \global\@namedef{#1@#2}{#3}%
229   \@ifnextchar[{\set@label@type{#1}{#2}}%]
230                \relax}%
231 \def\set@label@type#1#2[#3]{%
232   \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 222: adaptation to babel

Lines 223–228: The original definition (both standard LaTeX and babel).

Line 229: if an optional argument follows (containing the environment-type), continue with `\set@label@type`, otherwise return (the original behavior).

Lines 231, 232: set `\r@`⟨*labelname*⟩`@type` to the type of the respective environment.

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

```
233 \def\thref#1{%
234    \expandafter\ifx\csname r@#1@type\endcsname\None
235      \PackageWarning{\basename}{thref: Reference Type of ‘#1’ on page
236       \thepage \space undefined}\G@refundefinedtrue
237      \else\csname r@#1@type\endcsname~\fi
238    \expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
```

Lines 233, 238: similar to `\ref`.

Line 222: if a legal theorem type is given, then output `\r@`⟨*labelname*⟩`@type` and avoid linebreaking between the type and the number.

`\testdef` A problem occurred, when about 250 labels to theorem-like environments have been defined: after the end of a document, the .aux file is read once more (to check if references changed). Here, LaTeX redefines `\@newl@bel` into `\@testdef` – and LaTeX does not know that ntheorem's `\label` has an additional optional argument. Thus, the argument values are not processed, but are output as normal text. Normally, this did not matter since output has already been finished by a `\clearpage` in `\end{document}`. For so many labels, a page gets filled and the output routine is called.

```
239 \newcommand\org@testdef{}
240 \let\org@testdef\@testdef
241 \def\@testdef#1#2#3{%
242   \org@testdef{#1}{#2}{#3}%
243   \@ifnextchar[{\thm@gobbleopt}{}%
244 }
245 \newcommand\thm@gobbleopt{}
```

```
246 \long\def\thm@gobbleopt[#1]{}
```

Line 242: process the optional argument.

```
247 }% end of option thref ***********************************************
```

7.1.5 Option amsmath to Thmmarks

Most of the commands are extensions of commands in `amsmath.sty`.

```
248 \DeclareOption{amsmath}{% **************************************************
249 \if@thref
250  \PackageInfo{\basename}{option ‘amsmath’ handling for ‘thref’ loaded}%
```

if `thref` is active, the handling of labels in amsmath equations has also to be adapted.

`ams-thref`

```
251 \let\ltx@label\label
```

keep the handling of `\label` ... (the one defined above in the thref option). amsmath implements a special handling of `\label` inside of displaymath environments. It is extended to process the optional argument provided be the thref option:

```
252 \global\let\thm@df@label@optarg\@empty
253 \def\label@in@display#1{%
254     \ifx\df@label\@empty\else
255         \@amsmath@err{Multiple \string\label’s:
256             label ’\df@label’ will be lost}\@eha
257     \fi
258     \gdef\df@label{#1}%
259     \@ifnextchar[{\thm@label@in@display@optarg}{\thm@label@in@display@noarg}%]
260 }
261 \def\thm@label@in@display@noarg{%
262     \global\let\thm@df@label@optarg\@empty
263 }
264 \def\thm@label@in@display@optarg[#1]{%
265     \gdef\thm@df@label@optarg{#1}%
266 }
```

The contents of `\df@label` is handled when the equation is finished. (Currently) this happens in three macros. The modification consists of the check if `\thm@df@label@optarg` is non-empty (i.e., holds the optional argument), and to handle it.

```
267 \def\endmathdisplay@a{%
268   \if@eqnsw \gdef\df@tag{\tagform@\theequation}\fi
269   \if@fleqn \@xp\endmathdisplay@fleqn
270   \else \ifx\df@tag\@empty \else \veqno \alt@tag \df@tag \fi
271     \ifx\df@label\@empty \else
272       \ifx\thm@df@label@optarg\@empty \@xp\ltx@label\@xp{\df@label}%
273                \else \@xp\ltx@label\@xp{\df@label}[\thm@df@label@optarg]\fi
274      \fi
```

```
  \fi
  \ifnum\dspbrk@lvl>\m@ne
    \postdisplaypenalty -\@getpen\dspbrk@lvl
    \global\dspbrk@lvl\m@ne
  \fi
}
\def\make@display@tag{%
    \if@eqnsw
        \refstepcounter{equation}%
        \tagform@\theequation
    \else
        \iftag@
            \df@tag
            \global\let\df@tag\@empty
        \fi
    \fi
    \ifmeasuring@
    \else
      \ifx\df@label\@empty\else
        \ifx\thm@df@label@optarg\@empty \@xp\ltx@label\@xp{\df@label}%
                \else \@xp\ltx@label\@xp{\df@label}[\thm@df@label@optarg]\fi
        \global\let\df@label\@empty
      \fi
    \fi
}
\def\endmathdisplay@fleqn{%
  $\hfil\hskip\@mathmargin\egroup
  \ifnum\badness<\inf@bad \let\too@wide\@ne \else \let\too@wide\z@ \fi
  \ifx\@empty\df@tag
  \else
    \setbox4\hbox{\df@tag
        \ifx\thm@df@label@optarg\@empty \@xp\ltx@label\@xp{\df@label}%
                \else \@xp\ltx@label\@xp{\df@label}[\thm@df@label@optarg]\fi
    }%
  \fi
  \csname emdf@%
    \ifx\df@tag\@empty U\else \iftagsleft@ L\else R\fi\fi
  \endcsname
}
```

```
\fi
% end of if-thref in option amsmath ******************************************
\if@thmmarks
 \PackageInfo{\basename}{option `amsmath' handling for `thmmarks' loaded}%
\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:

```
319 \gdef\TagsPlusEndmarks{%
320       \global\let\Old@maketag@@@=\maketag@@@
321       \global\let\Old@df@tag=\df@tag
322       \if@eqnsw\SetTagPlusEndMark
323         \else
324           \iftag@\SetTagPlusEndMark
325             \else\SetOnlyEndMark
326           \fi
327       \fi}
```

Lines 320, 321: store the original macros.

Line 322: if equation numbers are set as default, call `\SetTagPlusEndMark` to set tag and end mark.

Lines 323, 324: if a tag is set manually, call `\SetTagPlusEndMark` to set tag and end mark.

Line 325: otherwise, call `\SetOnlyEndMark` to set only an end mark.

`\SetOnlyEndMark`

```
328 \gdef\SetOnlyEndMark{%
329       \global\tag@true
330         \iftagsleft@
331            \gdef\df@tag{\hbox
332                         to \displaywidth{\hss\PotEndMark{\maketag@@@}}}%
333         \else
334            \gdef\df@tag{\PotEndMark{\maketag@@@}[\ifhmode\else\hbox to .1pt{}\fi]}%
335         \fi}
```

Set only an end mark:

Line 329: force setting the end mark as a tag:

Lines 331, 332: 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 334: if tags are set to the right, the tag consists only of the (potential) endmark. If no endmark is set and TeX is not in hmode, an empty hbox is output (otherwise `\abovedisplayskip` will be ignored in `equation*`; this is executed in `endmathdisplay@a` when it comes to `\veqno\alt@tag\df@tag`).

`\SetTagPlusEndMark`

```
336 \newdimen{\tagwidth}
337 \gdef\SetTagPlusEndMark{%
338         \iftagsleft@
339           \gdef\maketag@@@##1{%
340              \settowidth{\tagwidth}{$##1$}%%  %% WM 17.10.2007
341              \hbox to \tagwidth{%
342                \hbox to \displaywidth{\m@th\normalfont##1%
343                                     \hss\PotEndMark{\hss}}\hss}}%
344         \else
345           \gdef\maketag@@@##1{\hbox{\m@th\normalfont##1%
346                             \llap{\hss\PotEndMark{\raisebox{-1.3em}}}}}%
347         \fi}
```

Set a tag *and* an end mark:

Lines 337–346: redefine the `\maketag@@@` macro:

Lines 338–342: 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. Put this box with width 0 and continue.

Lines 343, 344: if the tags are set to the right, the (potential) end mark is put below it.

`\tagform@`

`\maketag@@@` is also used via `\tagform@` in `\eqref` that may be called inside an environment. There, the original functionality must be used. The (small) commands `\th@ams@tagopen{(}` and `\th@ams@tagopen{(}` are provided as a hook for the `empheq` package.

```
348 \let\th@ams@@maketag@@@\maketag@@@
349 \gdef\th@ams@tagopen{(}
350 \gdef\th@ams@tagclose{)}
351 \gdef\th@ams@tagform@#1{%
352   \th@ams@@maketag@@@{\th@ams@tagopen\ignorespaces#1%
353                         \unskip\@@italiccorr\th@ams@tagclose}}
354 \gdef\eqref#1{\textup{\th@ams@tagform@{\ref{#1}}}}
```

`\RestoreTags`

```
355 \gdef\RestoreTags{%
356     \global\let\maketag@@@=\Old@maketag@@@
357     \global\let\df@tag=\Old@df@tag}
```

Lines 356, 357: restore the original macros.

`\endgather`

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

```
358 \gdef\endgather{%
359       \TagsPlusEndmarks % <<<<<<<<<
360       \math@cr
361       \black@\totwidth@
362     \egroup
363     $$%
364     \RestoreTags        % <<<<<<<<<
365     \ignorespacesafterend}
366 %
367 \expandafter\let\csname endgather*\endcsname\endgather
```

New:

Line 359: the last tag contains the potential endmark.

Line 364: restore the original macros.

Line 367: 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:

```
368 \def\endalign{%
369         \ifingather@\else       % <<<<<<<<<
370             \TagsPlusEndmarks\fi % <<<<<<<<<
371         \math@cr
372         \black@\totwidth@
373     \egroup
374     \ifingather@
375         \restorealignstate@
376         \egroup
377         \nonumber
378         \ifnum0=`{\fi\iffalse}\fi
379     \else
380         $$%
381         \RestoreTags             % <<<<<<<<<
382     \fi
383     \ignorespacesafterend}
```

New:

Lines 369, 370: if the `align` is not inside another environment, its tags have to contain the endmarks.

Line 381: this case, the original macros have to be restored.

```
384 \expandafter\let\csname endalign*\endcsname\endalign
385 \let\endxalignat\endalign
386 \expandafter\let\csname endxalignat*\endcsname\endalign
387 \let\endxxalignat\endalign
388 \let\endalignat\endalign
389 \expandafter\let\csname endalignat*\endcsname\endalign
390 \let\endflalign\endalign
391 \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:

```
392 \def\lendmultline@{%
393         \global\@eqnswfalse\tag@false\tagsleft@false
394         \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 393, 394: display no equation number, don't set an equation tag (but use the tag mechanism for the end mark - see `\TagsPlusEndmarks` and `\SetOnlyEndMark`), set it at the right, and call `\rendmultline`.

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

```
395 \def\rendmultline@{%
396     \TagsPlusEndmarks            % <<<<<<<<<
397     \iftag@
```

```
398         $\let\endmultline@math\relax
399             \ifshifttag@
400                 \hskip\multlinegap
401                 \llap{\vtop{%
402                     \raise@tag
403                     \normalbaselines
404                     \setbox\@ne\null
405                     \dp\@ne\lineht@
406                     \box\@ne
407                     \hbox{\strut@\make@display@tag}%
408                 }}%
409             \else
410                 \hskip\multlinetaggap
411                 \make@display@tag
412             \fi
413     \else
414         \hskip\multlinegap
415     \fi
416     \hfilneg
417         \math@cr
418     \egroup$$%
419     \RestoreTags}                % <<<<<<<<<
```

New:
Line 396: last tag contains the potential endmark.
Line 420: restore the original macros

`\endmathdisplay`

```
420 \def\endmathdisplay#1{%
421      \ifmmode \else \@badmath \fi
422      \TagsPlusEndmarks % <<<<<<<<<
423      \endmathdisplay@a
424      $$%
425      \RestoreTags        % <<<<<<<<<
426      \global\let\df@label\@empty \global\let\df@tag\@empty
427      \global\tag@false \global\let\alt@tag\@empty
428      \global\@eqnswfalse
429 }
```

Added Line 423: set potential end mark at bottom niveau of displaymath.

`equation`

```
430 \renewenvironment{equation}{%
431   \incr@eqnum
432   \mathdisplay@push
433   \st@rredfalse \global\@eqnswtrue
434   \mathdisplay{equation}%
435 }{%
436   \endmathdisplay{equation}%
437   \mathdisplay@pop
```

```
438   \ignorespacesafterend
439 }
440 \renewenvironment{equation*}{%
441   \mathdisplay@push
442   \st@rredtrue \global\@eqnswfalse
443   \mathdisplay{equation*}%
444 }{%
445   \endmathdisplay{equation*}%
446   \mathdisplay@pop
447   \ignorespacesafterend
448 }
```

unchanged from `amsmath.sty`.

```
449 \fi
450 }% end of option amsmath/thmmarks *************************************
```

7.1.6 Theorem-Layout Stuff

```
451 \let\thm@usestd\@undefined
452 \DeclareOption{standard}{\let\thm@usestd\relax}
453 \let\thm@noconfig\@undefined
454 \DeclareOption{noconfig}{\let\thm@noconfig\relax}
```

Options for selection of a configuration: if no such option is given `ntheorem.cfg` will be loaded (which has to be provided by the user), `[standard]` will load `ntheorem.std`, a predefined setting, and `[noconfig]` does not preload any configuration.

```
455 \gdef\InTheoType{None}
456 \gdef\NoneKeyword{None}
457 \gdef\NoneSymbol{None}
458 \gdef\None{None}
```

Set `\InTheoType` to `none` on the upper document level.

`\newtheoremstyle` With `\newtheoremstyle`, new theorem-layout styles are defined.

```
459 \gdef\newtheoremstyle#1#2#3{%
460   \expandafter\@ifundefined{th@#1}%
461    {\expandafter\gdef\csname th@#1\endcsname{%
462     \def\@begintheorem####1####2{#2}%
463     \def\@opargbegintheorem####1####2####3{#3}}}%
464    {\PackageError{\basename}{Theorem style #1 already defined}\@eha}}
```

Arguments:
⟨*style*⟩:=`#1`: the name of the theoremstyle to be defined,
⟨*cmd_seq1*⟩:=`#2`: command sequence for setting the header for environment instances with no optional text,
⟨*cmd_seq2*⟩:=`#3`: command sequence for setting the header for environment instances with optional text.
Line 460: if this style is not yet defined, define it.
Line 461: define `\th@`⟨*style*⟩ to be a macro which defines

Line 462: a) the two-argument macro `\@begintheorem#1#2` to be ⟨*cmd_seq1*⟩,
Line 463: b) `\@opargbegintheorem#1#2#3` to be ⟨*cmd_seq2*⟩.

The predefined theorem styles use this command.

`\renewtheoremstyle`

```
\gdef\renewtheoremstyle#1#2#3{%
  \expandafter\@ifundefined{th@#1}%
   {\PackageError{\basename}{Theorem style #1 undefined}\@ehc}%
     {}%
  \expandafter\let\csname th@#1\endcsname\relax
  \newtheoremstyle{#1}{#2}{#3}}
```

Arguments:
⟨*style*⟩:=`#1`: the name of the theoremstyle to be defined,
`#2`, `#3` as for `\newtheoremstyle`.
Checks, if theoremstyle ⟨*style*⟩ is already defined. If so, `\th@`⟨*style*⟩ is made undefined and `\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.

```
\newtheoremstyle{plain}%
  {\item[\hskip\labelsep \theorem@headerfont ##1\ ##2\theorem@separator]}%
  {\item[\hskip\labelsep \theorem@headerfont ##1\ ##2\ (##3)\theorem@separator]}
%
\newtheoremstyle{break}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
          ##1\ ##2\theorem@separator}\hbox{\strut}}}]}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
          ##1\ ##2\ (##3)\theorem@separator}\hbox{\strut}}}]}
%
\newtheoremstyle{change}%
  {\item[\hskip\labelsep \theorem@headerfont ##2\ ##1\theorem@separator]}%
  {\item[\hskip\labelsep \theorem@headerfont ##2\ ##1\ (##3)\theorem@separator]}
%
\newtheoremstyle{changebreak}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
          ##2\ ##1\theorem@separator}\hbox{\strut}}}]}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
          ##2\ ##1\ (##3)\theorem@separator}\hbox{\strut}}}]}
%
\newtheoremstyle{margin}%
  {\item[\theorem@headerfont \llap{##2}\hskip\labelsep ##1\theorem@separator]}%
  {\item[\theorem@headerfont \llap{##2}\hskip\labelsep ##1\ (##3)\theorem@separator]}
%
\newtheoremstyle{marginbreak}%
  {\item[\rlap{\vbox{\hbox{\theorem@headerfont
```

```
    \llap{##2}\hskip\labelsep\relax ##1\theorem@separator}\hbox{\strut}}}]}
  {\item[\rlap{\vbox{\hbox{\theorem@headerfont
    \llap{##2}\hskip\labelsep\relax ##1\
    (##3)\theorem@separator}\hbox{\strut}}}]}
%
\newtheoremstyle{nonumberplain}%
  {\item[\theorem@headerfont\hskip\labelsep ##1\theorem@separator]}%
  {\item[\theorem@headerfont\hskip \labelsep ##1\ (##3)\theorem@separator]}
%
\newtheoremstyle{nonumberbreak}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
          ##1\theorem@separator}\hbox{\strut}}}]}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
          ##1\ (##3)\theorem@separator}\hbox{\strut}}}]}
%
\newtheoremstyle{empty}%
  {\item[]}%
  {\item[\theorem@headerfont \hskip\labelsep\relax ##3]}
\newtheoremstyle{emptybreak}%
  {\item[]}%
  {\item[\rlap{\vbox{\hbox{\hskip\labelsep\relax \theorem@headerfont
          ##3\theorem@separator}\hbox{\strut}}}]}
%
\@namedef{th@nonumbermargin}{\th@nonumberplain}
\@namedef{th@nonumberchange}{\th@nonumberplain}
\@namedef{th@nonumbermarginbreak}{\th@nonumberbreak}
\@namedef{th@nonumberchangebreak}{\th@nonumberbreak}
\@namedef{th@plainNo}{\th@nonumberplain}
\@namedef{th@breakNo}{\th@nonumberplain}
\@namedef{th@marginNo}{\th@nonumberplain}
\@namedef{th@changeNo}{\th@nonumberplain}
\@namedef{th@marginbreakNo}{\th@nonumberbreak}
\@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 TEX 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:

```
\gdef\theoremstyle#1{%
  \@ifundefined{th@#1}{\@warning
        {Unknown theoremstyle ‘#1’. Using ‘plain’}%
        \theorem@style{plain}}%
     {\theorem@style{#1}}}
\newtoks\theorem@style
\newtoks\theorem@@style
\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`

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

`\theoremnumbering`

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

`theoremskips`

```
\newskip\theorempreskipamount
\newskip\theorempostskipamount
\newskip\theoremframepreskipamount
\newskip\theoremframepostskipamount
\newskip\theoreminframepreskipamount
\newskip\theoreminframepostskipamount
\global\theorempreskipamount\topsep
\global\theorempostskipamount\topsep
\global\theoremframepreskipamount\topsep
\global\theoremframepostskipamount\topsep
\global\theoreminframepreskipamount\topsep
\global\theoreminframepostskipamount\topsep
```

```
\newif\ifuse@newframeskips\global\use@newframeskipsfalse
\newtoks\theorem@preskip
\global\theorem@preskip{\topsep}
\def\theorempreskip#1{%
  \theorem@preskip{#1}\global\use@newframeskipstrue}
```

```
559 \newtoks\theorem@postskip
560 \global\theorem@postskip{\topsep}
561 \def\theorempostskip#1{%
562   \theorem@postskip{#1}\global\use@newframeskipstrue}
563 \newtoks\theorem@framepreskip
564 \global\theorem@framepreskip{\topsep}
565 \def\theoremframepreskip#1{%
566   \theorem@framepreskip{#1}\global\use@newframeskipstrue}
567 \newtoks\theorem@framepostskip
568 \global\theorem@framepostskip{\topsep}
569 \def\theoremframepostskip#1{%
570   \theorem@framepostskip{#1}\global\use@newframeskipstrue}
571 \newtoks\theorem@inframepreskip
572 \global\theorem@inframepreskip{\topsep}
573 \def\theoreminframepreskip#1{%
574   \theorem@inframepreskip{#1}\global\use@newframeskipstrue}
575 \newtoks\theorem@inframepostskip
576 \global\theorem@inframepostskip{\topsep}
577 \def\theoreminframepostskip#1{%
578   \theorem@inframepostskip{#1}\global\use@newframeskipstrue}
```

Line 554: switch whether new skip scheme is used (default for compatibility, with old versions: no)

Line 555, 556: define and initialize internal token (not a skip, just a token),

Line 557, 558: define command to assign argument to token, and activate use of new skip scheme,

Line 559–578: analogously for the other skips.

The new theoremskip scheme is automatically activated if one of the above commands is invoked (for that, they are not directly implemented as `\newtoks`, but as complex commands).

`\theoremindent`

```
579 \newdimen\theoremindent
580 \global\theoremindent0cm
581 \newdimen\theorem@indent
582 \newdimen\theoremrightindent
583 \global\theoremrightindent0cm
584 \newdimen\theorem@rightindent
```

`\theoremheaderfont`

```
585 \newtoks\theoremheaderfont
586 \global\theoremheaderfont{\normalfont\bfseries}
587 \def\theorem@headerfont{\normalfont\bfseries}
```

`\theoremseparator`

```
588 \newtoks\theoremseparator
589 \global\theoremseparator{}
590 \def\theorem@separator{}
```

\theoremprework
\theorempostwork

```
591 \newtoks\theoremprework
592 \global\theoremprework{\relax}
593 \newtoks\theorempostwork
594 \global\theorempostwork{\relax}
595 \def\theorem@prework{}
```

\theoremsymbol

```
596 \newtoks\theoremsymbol
597 \global\theoremsymbol{}
```

\qedsymbol

```
598 \newtoks\qedsymbol
599 \global\qedsymbol{}
```

\theoremkeyword

```
600 \newtoks\theoremkeyword
601 \global\theoremkeyword{None}
```

\theoremclass

```
602 \gdef\theoremclass#1{%
603      \csname th@class@#1\endcsname}
604 \gdef\th@class@LaTeX{%
605      \theoremstyle{plain}%
606      \theoremheaderfont{\normalfont\bfseries}%
607      \theorembodyfont{\itshape}%
608      \theoremseparator{}%
609      \theoremprework{\relax}%
610      \theorempostwork{\relax}%
611      \ifuse@newframeskips
612        \theorempreskip{0cm}%
613        \theorempostskip{0cm}%
614        \theoremframepreskip{0cm}%
615        \theoremframepostskip{0cm}%
616        \theoreminframepreskip{0cm}%
617        \theoreminframepostskip{0cm}%
618      \fi
619      \theoremindent0cm
620      \theoremrightindent0cm
621      \theoremnumbering{arabic}%
622      \theoremsymbol{}}
```

Calling `\theoremclass{`⟨*env*⟩`}` calls `\th@class@`⟨*env*⟩ (which is defined in `\@newtheorem` in Lines 726–740). `\th@class@`⟨*env*⟩ restores all style parameters to their values given for ⟨*env*⟩. Especially, `\th@class@LaTeX` restores the standard LaTeX parameters.

\qedsymbol

```
623 \newtoks\qedsymbol
624 \global\qedsymbol{}
```

Compatibility with amsthm.

amsthm

```
\DeclareOption{amsthm}{% ***********************************************
  \PackageInfo{\basename}{Option `amsthm' loaded}%
\def\swapnumbers{\PackageError{\basename}{swapnumbers not implemented.
  Use theoremstyle change instead.}\@eha}

\gdef\th@plain{%
  \def\theorem@headerfont{\normalfont\bfseries}\itshape%
  \def\@begintheorem##1##2{%
      \item[\hskip\labelsep \theorem@headerfont ##1\ ##2.]}%
  \def\@opargbegintheorem##1##2##3{%
     \item[\hskip\labelsep \theorem@headerfont ##1\ ##2\ (##3).]}}
\gdef\th@nonumberplain{%
  \def\theorem@headerfont{\normalfont\bfseries}\itshape%
  \def\@begintheorem##1##2{%
      \item[\hskip\labelsep \theorem@headerfont ##1.]}%
  \def\@opargbegintheorem##1##2##3{%
     \item[\hskip\labelsep \theorem@headerfont ##1\ (##3).]}}
\gdef\th@definition{%
  \th@plain\def\theorem@headerfont{\normalfont\bfseries}\normalfont}
\gdef\th@nonumberdefinition{%
  \th@nonumberplain\def\theorem@headerfont{\normalfont\bfseries}\normalfont}
\gdef\th@remark{%
  \th@plain\def\theorem@headerfont{\itshape}\normalfont}
\gdef\th@nonumberremark{%
  \th@nonumberplain\def\theorem@headerfont{\itshape}\normalfont}
%%% TODO skips initialisieren
\newcounter{proof}%
\if@thmmarks
 \newcounter{currproofctr}%
 \newcounter{endproofctr}%
\fi
\newcommand{\openbox}{\leavevmode
  \hbox to.77778em{%
  \hfil\vrule
  \vbox to.675em{\hrule width.6em\vfil\hrule}%
  \vrule\hfil}}
\gdef\proofSymbol{\openbox}
\newcommand{\proofname}{Proof}
\newenvironment{proof}[1][\proofname]{
  \th@nonumberplain
  \def\theorem@headerfont{\itshape}%
  \normalfont
  \theoremsymbol{\ensuremath{_\blacksquare}}
  \@thm{proof}{proof}{#1}}%
  {\@endtheorem}
}% end of option amsthm ***********************************************
```

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

7.1.7 Theorem-Environment Handling Stuff

Original: ltthm.dtx

```
\newskip\thm@topsep
\newskip\thm@topsepadd
```

Two auxiliary variables.

Defining New Theorem-Environments.

`\newtheorem`

```
\gdef\newtheorem{%
  \newtheorem@i%
}
```

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

```
\gdef\newtheorem@i{%
   \@ifstar
     {\expandafter\@ifundefined{th@nonumber\the\theorem@style}%
   {\PackageError{\basename}{Theorem style {nonumber\the\theorem@style}
            undefined (you need it here for newtheorem*) }\@ehc}%
     {}%
     \edef\@tempa{{nonumber\the\theorem@style}}%
      \expandafter\theorem@@style\@tempa\@newtheorem}%
     {\edef\@tempa{{\the\theorem@style}}%
      \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`

```
\gdef\renewtheorem{%
   \@ifstar
     {\expandafter\@ifundefined{th@nonumber\the\theorem@style}%
   {\PackageError{\basename}{Theorem style {nonumber\the\theorem@style}
            undefined (you need it here for newtheorem*) }\@ehc}%
     {}%
     \edef\@tempa{{nonumber\the\theorem@style}}%
      \expandafter\theorem@@style\@tempa\@renewtheorem}%
     {\edef\@tempa{{\the\theorem@style}}%
      \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`.

```
\gdef\@newtheorem#1{%
  \thm@tempiffalse
  \expandafter\@ifdefinable\csname #1\endcsname
  {\expandafter\@ifdefinable\csname #1*\endcsname
   {\thm@tempiftrue
    \thm@definelthm{#1}% for lists
    \if@thmmarks
      \expandafter\@ifundefined{c@curr#1ctr}%
        {\newcounter{curr#1ctr}}{}%
      \expandafter\@ifundefined{c@end#1ctr}%
        {\newcounter{end#1ctr}}{}%
    \fi
    \expandafter\protected@xdef\csname #1Symbol\endcsname{\the\theoremsymbol}%
    \expandafter\protected@xdef\csname #1@postwork\endcsname{%
       \the\theorempostwork}%
    \expandafter\gdef\csname#1\endcsname{%
       \let\thm@starredenv\@undefined
       \csname mkheader@#1\endcsname}%
    \expandafter\gdef\csname#1*\endcsname{%
       \let\thm@starredenv\relax
       \csname mkheader@#1\endcsname}%
    \def\@tempa{\expandafter\noexpand\csname end#1\endcsname}%
    \expandafter\xdef\csname end#1*\endcsname{\@tempa}%
    \expandafter\xdef\csname setparms@#1\endcsname
     {\noexpand \def \noexpand \theorem@headerfont
        {\the\theoremheaderfont\noexpand\theorem@checkbold}%
      \noexpand \def \noexpand \theorem@separator
        {\the\theoremseparator}%
      \noexpand \def \noexpand \theorem@prework
        {\the\theoremprework}%
      \noexpand\ifuse@newframeskips
        \noexpand \theorempreskipamount \the\theorem@preskip
        \noexpand \theoremframepreskipamount \the\theorem@framepreskip
        \noexpand \theoreminframepreskipamount \the\theorem@inframepreskip
        \noexpand \theorempostskipamount \the\theorem@postskip
        \noexpand \theoremframepostskipamount \the\theorem@framepostskip
        \noexpand \theoreminframepostskipamount \the\theorem@inframepostskip
      \noexpand\fi
      \noexpand \def \noexpand \theorem@indent
        {\the\theoremindent}%
      \noexpand \def \noexpand \theorem@rightindent
       {\the\theoremrightindent}%
      \the \theorembodyfont
      \noexpand\csname th@\the \theorem@@style \endcsname}%
    \expandafter\xdef\csname th@class@#1\endcsname
     {\noexpand\theoremstyle{\the\theorem@style}%
      \noexpand\theoremheaderfont{\the\theoremheaderfont}%
```

```
743         \noexpand\theorembodyfont{\the \theorembodyfont}%
744         \noexpand\theoremseparator{\the\theoremseparator}%
745         \noexpand\theoremprework{\the\theoremprework}%
746         \noexpand\theorempostwork{\the\theorempostwork}%
747         \noexpand\ifuse@newframeskips
748           \noexpand\theorempreskip {\the\theorem@preskip}%
749           \noexpand\theoremframepreskip {\the\theorem@framepreskip}%
750           \noexpand\theoreminframepreskip {\the\theorem@inframepreskip}%
751           \noexpand\theorempostskip {\the\theorem@postskip}%
752           \noexpand\theoremframepostskip {\the\theorem@framepostskip}%
753           \noexpand\theoreminframepostskip {\the\theorem@inframepostskip}%
754         \noexpand\fi
755         \noexpand\theoremindent\the\theoremindent%
756         \noexpand\theoremrightindent\the\theoremrightindent%
757         \noexpand\theoremnumbering{\the\theoremnumbering}%
758         \noexpand\theoremsymbol{\the\theoremsymbol}}%
759       }}%
760     \theoremprework{\relax}%
761     \theorempostwork{\relax}%
762     \@ifnextchar[{\@othm{#1}}{\@nthm{#1}}}% MUST NOT BE IN ANY IF !!!
```

Argument: $\langle \mathit{env} \rangle$:=`#1` is the (internal) environment name to be defined, which is read from the LaTeX source.

Line 698: check if $\langle \mathit{env} \rangle$ is not yet defined (or is redefined).

Lines 700–725 are executed exactly if $\langle \mathit{env} \rangle$ and $\langle \mathit{env} \rangle$* are not yet defined.

Line 700: `\thm@tempif=true` iff $\langle \mathit{env} \rangle$ and $\langle \mathit{env} \rangle$* are not yet defined.

Line 701: Initialize theorem list handling for $\langle \mathit{env} \rangle$.

Lines 702–707: if `thmmarks` is active and the counters are not yet defined, for every theorem-like, define
`curr`$\langle \mathit{env} \rangle$`ctr`: in the ith environment of type $\langle \mathit{env} \rangle$, `curr`$\langle \mathit{env} \rangle$`ctr` $= i$, and
`end`$\langle \mathit{env} \rangle$`ctr`: when the innermost environment is of type $\langle \mathit{env} \rangle$, in the jth potential position for an end mark in this environment, `end`$\langle \mathit{env} \rangle$`ctr` $= j$. (if the counters are already defined, $\langle \mathit{env} \rangle$ is redefined, and these internal counters have to be continued).

Lines 708–756: define several commands: (`\xdef` expands the definition at the time it is called and makes it global):

Line 708: store the current value of `\theoremsymbol` as `\`$\langle \mathit{env} \rangle$`Symbol`.

Line 709, 710: store the current value of `\theorempostwork` as `\`$\langle \mathit{env} \rangle$`postwork`.

Lines 711–713, 714–716: Define the commands `\env` and `\env*` to set the header of $\langle \mathit{env} \rangle$ by calling `\mkheader@`$\langle \mathit{env} \rangle$. (using a switch `\thm@starredenv`: `\relax` iff starred).

Lines 717, 718: Set `\end`$\langle \mathit{env} \rangle$`*` to `\end`$\langle \mathit{env} \rangle$.

Lines 719–737: define `\setparms@`$\langle \mathit{env} \rangle$ to set the style parameters of the header for every $\langle \mathit{env} \rangle$ environment (in the sequel, *current* means, at the moment `\@newtheorem` is called):

Lines 720, 721: setting `\theorem@headerfont` to the *current* value of `\theoremheaderfont`, followed by a check if it is a bold style,

Lines 722, 723: setting `\theorem@separator` to the *current* value of `\theoremseparator`,

Lines 724, 725: setting `\theorem@prework` to the *current* value of `\theoremprework`,

Lines 726–733: if new skip schema is used: setting the skips to the *current* values hold in the respective tokens,

Line 734, 735: setting `\theorem@indent` to the *current* value of `\theoremindent`,

Line 736: executing the command sequence currently stored in `\theorembodyfont`, and

Line 737: calling `th@\the\theorem@@style` (which initializes `\@begintheorem` and `\@opargbegintheorem` according to the *current* value of `\theoremstyle` by calling `th@\the\theorem@@style`).

Line 738–755: define `\th@class@`⟨*env*⟩ to initialize all style parameters as they are set for the ⟨*env*⟩ environment. (call skip-initializing commands only if new skip scheme is activated).

Note, that the `\@ifdefinable` from lines 698/699 end in line 754.

Line 757, 758: reset `\@theoremprework/postwork`.

Line 759: According to the next character, call `\@othm{`⟨*env*⟩`}` (if another counter is used) or `\@nthm{`⟨*env*⟩`}`.

Thus, when calling `\@newthm` with `#1`=⟨*env*⟩, for current values `\theoremstyle=plain`, `\theorembodyfont=\upshape`, `\theoremheaderfont=\bf`, `\theoremseparator=:`, `\theoremindent=1cm`, `\theoremnumbering=arabic`, and `\theoremsymbol=\Box`, the macro `\setparms@`⟨*env*⟩ is defined as

```
\setparms@⟨env⟩ == \def\theorem@headerfont{\bf\theorem@checkbold}
                   \def\theorem@separator{:}
                   \def\theorem@indent{1cm}
                   \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{1cm}
                  \def\theoremnumbering{arabic}
                  \def\theoremsymbol{\Box}
```

Note, that line 759 must not be inside *any* `\if...\fi` construct.

`\@renewtheorem`

```
763 \gdef\@renewtheorem#1{%
764   \expandafter\@ifundefined{#1}%
```

```
765     {\PackageError{\basename}{Theorem keyword #1 undefined}\@ehc}%
766       {}%
767   \expandafter\let\csname #1\endcsname\relax
768   \expandafter\let\csname #1*\endcsname\relax
769   \@newtheorem{#1}}
```

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

```
770 \gdef\@nthm#1#2{%
771   \expandafter\protected@xdef\csname num@addtheoremline#1\endcsname{%
772                 \noexpand\@num@addtheoremline{#1}{#2}}%
773   \expandafter\protected@xdef\csname nonum@addtheoremline#1\endcsname{%
774                 \noexpand\@nonum@addtheoremline{#1}{#2}}%
775   \theoremkeyword{#2}%
776   \expandafter\protected@xdef\csname #1Keyword\endcsname
777                 {\the\theoremkeyword}%
778   \@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 LaTeX source).

Lines 771–774: Define \(no)`num@addtheoremline`⟨*env*⟩ to call `\@`(no)`num@addtheoremline{`⟨*env*⟩`}{`⟨*output_name*⟩`}`.
For comments on `\@num@addtheoremline` and `\@nonum@addtheoremline` see Section 7.1.9.

Lines 775–777: Define \⟨*env*⟩`Keyword`⟨*env*⟩ to typeset/output ⟨*output_name*⟩. (note the similarity with the handling of `\theoremsymbol` for handling complex keywords)

Line 778: According to the next character, call `\@xnthm{`⟨*env*⟩`}{`⟨*output_name*⟩`}` (if ⟨*env*⟩-environments should be numbered relative to some structuring level) or `\@ynthm{`⟨*env*⟩`}{`⟨*output_name*⟩`}`.

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

```
779 \gdef\@othm#1[#2]#3{%
780   \@ifundefined{c@#2}{\@nocounterr{#2}}%
781    {\ifthm@tempif
782      \global\@namedef{the#1}{\@nameuse{the#2}}%
783      \expandafter\protected@xdef\csname num@addtheoremline#1\endcsname{%
784                 \noexpand\@num@addtheoremline{#1}{#3}}%
785      \expandafter\protected@xdef\csname nonum@addtheoremline#1\endcsname{%
786                 \noexpand\@nonum@addtheoremline{#1}{#3}}%
```

```
787      \theoremkeyword{#3}%
788      \expandafter\protected@xdef\csname #1Keyword\endcsname
789               {\the\theoremkeyword}%
790       \expandafter\gdef\csname mkheader@#1\endcsname
791         {\csname setparms@#1\endcsname
792                  \@thm{#1}{#2}{#3}}%
793       \global\@namedef{end#1}{\@endtheorem}\fi}}
```

Arguments:
⟨*env*⟩:=`#1` is the (internal) environment name to be defined (transmitted from `\@newtheorem`).
⟨*use_ctr*⟩:=`#2` is the internal name of the theorem which counter is used, and
⟨*output_name*⟩:=`#3` is its "name" to be used in the output (both read from the LaTeX source).

Line 780: if the counter to be used is undefined, goto error, else set `\the`⟨*env*⟩ to use `\the`⟨*use_ctr*⟩ and do the following:

Lines 782–790 happen only if ⟨*env*⟩ is not yet defined or gets redefined:

Line 782: (from latex.ltx) make ⟨*env*⟩ use the counter ⟨*use_ctr*⟩.

Lines 783–789 similar to lines 771–777 of `\@nthm`.

Lines 790–792 define `\mkheader@`⟨*env*⟩ to set the style parameters of the header and set the header (by `\@thm`):

`\mkheader@`⟨*env*⟩ == `\setparms@`⟨*env*⟩`\@thm{`⟨*env*⟩`}{`⟨*use_ctr*⟩`}{`⟨*output_name*⟩`}`.

(`\setparms@`⟨*env*⟩ is defined when `\@newtheorem{`⟨*env*⟩`}` is carried out).

Line 793: (from latex.ltx): `\end`⟨*env*⟩ calls `\@endtheorem`.

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

```
794 \gdef\@xnthm#1#2[#3]{%
795   \ifthm@tempif
796       \expandafter\@ifundefined{c@#1}%
797          {\@definecounter{#1}}{}%
798       \@newctr{#1}[#3]%
799       \expandafter\xdef\csname the#1\endcsname{%
800         \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
801            {\noexpand\csname\the\theoremnumbering\endcsname{#1}}}%
802       \expandafter\gdef\csname mkheader@#1\endcsname
803         {\csname setparms@#1\endcsname
804          \@thm{#1}{#1}{#2}}%
805       \global\@namedef{end#1}{\@endtheorem}\fi}
```

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,
⟨*level*⟩:=`#3` is the structuring level relative to which ⟨*env*⟩ has to be numbered (both read from the LaTeX source).

Lines 796–805 happen only if ⟨*env*⟩ is not yet defined or gets redefined:

Lines 796, 797: in not yet defined, define ⟨*env*⟩- counter (otherwise, ⟨*env*⟩ is redefined).

Line 799: (from latex.ltx): define the counter for ⟨*env*⟩ and add ⟨*level*⟩ to its reset-triggers.

Lines 800, 801: define `\the`⟨*env*⟩ to be the command sequence

`\the`⟨*level*⟩`\@thmcountersep`⟨*numbering*⟩`{`⟨*env*⟩`}` ,

where ⟨*numbering*⟩ is the value of `\theoremnumbering` when `\@xnthm` (and thus, `\newtheorem{`⟨*env*⟩`}`) is called.

Lines 802–804: define `\mkheader@`⟨*env*⟩ to set the style parameters of the header and set the header (by `\@thm`):

`\mkheader@`⟨*env*⟩ == `\setparms@`⟨*env*⟩`\@thm{`⟨*env*⟩`}{`⟨*env*⟩`}{`⟨*output_name*⟩`}`.

(`\setparms@`⟨*env*⟩ is defined when `\@newtheorem{`⟨*env*⟩`}` is carried out).

Line 805: (from latex.ltx): `\end`⟨*env*⟩ calls `\@endtheorem`.

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

```
\gdef\@ynthm#1#2{%
  \ifthm@tempif
     \expandafter\@ifundefined{c@#1}%
        {\@definecounter{#1}}{}%
     \expandafter\xdef\csname the#1\endcsname
        {\noexpand\csname\the\theoremnumbering\endcsname{#1}}%
     \expandafter\gdef\csname mkheader@#1\endcsname
       {\csname setparms@#1\endcsname
        \@thm{#1}{#1}{#2}}%
     \global\@namedef{end#1}{\@endtheorem}\fi}
```

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

Notes on vertical spacing In LaTeX and in `ntheorem`, theorems are organized as `\trivlist`s, using the following skips:

- `\@topsep`: above,
- `\@topsepadd`: below (the name is misleading).

In LaTeX(cf. `latex.ltx` and its sources), the handling is as follows:

- `\@begintheorem`: calls `\trivlist` and afterwards `\item`.
- `\trivlist`: calls `\@trivlist`.
- `\@trivlist`: sets `\@topsepadd := \topsep + ifvmode: \partopsep`, `\@topsep := \@topsepadd`, and `\@topsep += \parskip` – this latter will be undone later in `\@item`!

- `\item`: calls `\@item`.
- `\@item`: does `\addvspace\@topsep` and `\addvspace{-\parskip}`.

⇒ only `\@topsep` must be set to ⟨*value*⟩+`\parskip` before the actual text is output by `\@item`

- `\@endtheorem`: calls `\endtrivlist`.
- `\endtrivlist`: note `\@noparlist` is only true if list is itself at the beginning of a list item – this is (usually) not the case for theorems.
  The `\ifdim\lastskip ...` part actually leaves skips unchanged in default cases (`\@outerparskip := \parskip` in `\@trivlist`).
  Afterwards, it calls `\@endparenv`
- `\@endparenv`: does `\addvspace\@topsepadd`.

In `ntheorem`, `\`⟨*env*⟩is used instead of a common `\@begintheorem` (defined by `\newtheorem` in Lines 711–713, 714–716) that call `\mkheader@`⟨*env*⟩. The actual definition of `\mkheader@`⟨*env*⟩ is done in `\@othm`, `\@xnthm`, or `\@ynthm` as `\mkheader@`⟨*env*⟩ == `\setparms@`⟨*env*⟩`\@thm{...}{...}{...}`.
Amongst this, `\@thm{...}{...}{...}`, whose code is given next,

- generates the actual output of the theorem header (via `\@xthm`/`\@begintheorem` or `\@ythm`/`\@opargbegintheorem`) where `\@begintheorem` and `\@opargbegintheorem` are redefined by `ntheorem`:
- `\@begintheorem` and `\@opargbegintheorem` do *not* contain the `\trivlist` command, but only set the `\item` (internally `\@item` that adds space `\@topsep-\parskip`).
- the `\trivlist` command itself is contained in `\@thm`, recall that as described above, it is not concerned with skips at all.

⇒ `\@thm` has to set `\@topsep :=` ⟨*value*⟩`+\parskip` for the space *above* the theorem.

- `\end`⟨*env*⟩is defined in `\@othm`, `\@xnthm`, or `\@ynthm` as `\@endtheorem` which is defined in Lines 95 (thmmarks active) and 858 (thmmarks off).
- `\@endtheorem` with thmmarks on calls ntheorem's `\@endtrivlist` (Line 11) which calls `\@endparenv`.
- `\@endtheorem` with thmmarks off calls the original `\endtrivlist` which calls the original `\@endtrivlist` which calls `\@endparenv`.

⇒ `\@topsepadd` is used for the vertical space after the theorem as usual.

Handling Instances of Theorem-Environments.

`\@thm` `\@thm` is called by `\@`⟨*env*⟩ (which is defined by `\@othm`/`\@xnthm`/`\@ynthm`).

```
\gdef\@thm#1#2#3{%
  \if@thmmarks
    \stepcounter{end\InTheoType ctr}%
  \fi
  \renewcommand{\InTheoType}{#1}%
  \if@thmmarks
    \stepcounter{curr#1ctr}%
```

```
823       \setcounter{end#1ctr}{0}%
824     \fi
825     \refstepcounter{#2}%
826     \theorem@prework
827     \trivlist % latex's \trivlist, calling latex's \@trivlist unchanged
828     \ifuse@newframeskips  % cf. latex.ltx for topsepadd: \@trivlist
829       \ifthm@inframe
830         \thm@topsep\theoreminframepreskipamount
831         \thm@topsepadd\theoreminframepostskipamount
832        \else
833         \thm@topsep\theorempreskipamount
834         \thm@topsepadd\theorempostskipamount
835        \fi
836      \else% oldframeskips
837        \thm@topsep\theorempreskipamount
838        \thm@topsepadd \theorempostskipamount
839        \ifvmode\advance\thm@topsepadd\partopsep\fi
840     \fi
841     \@topsep\thm@topsep
842     \@topsepadd\thm@topsepadd
843     \advance\linewidth -\theorem@indent
844     \advance\linewidth -\theorem@rightindent
845     \advance\@totalleftmargin \theorem@indent
846     \parshape \@ne \@totalleftmargin \linewidth
847     \@ifnextchar[{\@ythm{#1}{#2}{#3}}{\@xthm{#1}{#2}{#3}}}
```

Changed to three instead of two 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; all arguments are transmitted from `\@othm`/`\@xnthm`/`\@ynthm`.

Lines 817–819: if `thmmarks` is active, the counter for the current environment ⟨*env'*⟩ is incremented, since the last endmark in environment ⟨*env'*⟩ is definitely not the position for its endmark (necessary for nested environments ending at the same time).

Line 820: set `\InTheoType` to ⟨*env*⟩.

Lines 821–824: if `thmmarks` is active, increment `curr`⟨*env*⟩`ctr` and set `end`⟨*env*⟩`ctr` to 0.

Line 825: adapted from latex.ltx: increment the corresponding counter.

Line 826: perform `prework` (before theorem structure is generated).

Line 827: call LaTeX's original `\trivlist`. It does just organizational things, no actual skips! The skip is added later when the first `\item` is typeset.

Lines 828–840: handle `\theorempreskipamount`, `\theorempostskipamount`, `\theoreminframepreskipamount`, and `\theoreminframepostskipamount` (old skip schema: if in `vmode`, there is additional space, cf. `\trivlist` and `\@trivlist` in latex.ltx).

Lines 841–842: initialize `\@topsep` (the space before the first item) and `\@topsepadd` (the space after the first item). (see explanation of spacing in the note preceding this macro.)

Lines 843–845: handle `\theoremindent`.

Line 846: if there is an optional argument, call
`\@ythm{`⟨*env*⟩`}{`⟨*use_ctr*⟩`}{`⟨*output_name*⟩`}`, otherwise call
`\@xthm{`⟨*env*⟩`}{`⟨*use_ctr*⟩`}{`⟨*output_name*⟩`}`.

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

```
848 \def\@xthm#1#2#3{%
849   \@begintheorem{#3}{\csname the#2\endcsname}%
850   \ifx\thm@starredenv\@undefined
851     \thm@thmcaption{#1}{{#3}{\csname the#2\endcsname}{}}\fi
852   \ignorespaces}
```

Changed to three instead of two 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.
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.

```
853 \def\@ythm#1#2#3[#4]{%
854   \expandafter\global\expandafter\def\csname#1name\endcsname{#4}%
855   \@opargbegintheorem{#3}{\csname the#2\endcsname}{#4}%
856   \ifx\thm@starredenv\@undefined
857     \thm@thmcaption{#1}{{#3}{\csname the#2\endcsname}{#4}}\fi
858   \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 LaTeX source.

Line 854: define `\`⟨*env*⟩`name` to be the optional argument.

Line 855: call

`\@opargbegintheorem{`⟨*output_name*⟩`}{\the`⟨*use_ctr*⟩`}{`⟨*opt_text*⟩`}`

which outputs the header.

Line 856, 857: 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` is called for every `\end{`⟨*env*⟩`}`, where ⟨*env*⟩ is a theorem-like environment. (note that `\@endtheorem` it is also changed by option [thmmarks] to organize the placement of the corresponding end mark). `\InTheoType` gives the innermost theorem-like environment, i.e. the one to be ended:

```
\gdef\@endtheorem{%
  \endtrivlist
  \csname\InTheoType @postwork\endcsname
  }
```

7.1.8 Framed and Boxed Theorems

The option 'framed' activates framed and boxed layouts. It requires to load the `framed` package and the `pstricks` package.

`framed`

```
\DeclareOption{framed}{%*********************************
\newtoks\shadecolor
\shadecolor{gray}
\let\theoremframecommand\relax
```

`\newshadedtheorem`

```
\def\thm@framedprework{%
    \ifdim\lastskip <\theoremframepreskipamount
     \vskip -\lastskip
     \vskip\theoremframepreskipamount
    \fi
    \ifuse@newframeskips\vspace{-\topsep}\fi
    \thm@inframetrue
    \framed}%
\def\thm@framedpostwork{%
    \endframed
    \ifuse@newframeskips\unskip\fi
    \vskip\theoremframepostskipamount}%
```

Note: since `framed.sty` adds `\topsep` before and after a frame, `\vspace{-\topsep}` and `\unskip` are used to remove it. For compatibility with previous versions, this is only done if the new skip schema is used.

`\newshadedtheorem`

```
\def\newshadedtheorem#1{%
  \expandafter\xdef\csname#1@shadecolor\endcsname{%
    \the\shadecolor}%
  \ifx\theoremframecommand\relax
    \expandafter\global\expandafter\xdef\csname#1@framecommand\endcsname{%
      \noexpand\psframebox[fillstyle=solid,
                     fillcolor=\csname#1@shadecolor\endcsname,
                     linecolor=\csname#1@shadecolor\endcsname]}%
  \else
   \expandafter\global\expandafter\let\csname#1@framecommand\endcsname%
     \theoremframecommand%
  \fi
```

```
  \expandafter\xdef\csname#1@@prework\endcsname{%
    \the\theoremprework}%
  \theoremprework{%
    \csname#1@@prework\endcsname
    \def\FrameCommand{\csname#1@framecommand\endcsname}%
    \thm@framedprework}%
  \expandafter\xdef\csname#1@@postwork\endcsname{%
    \the\theorempostwork}%
  \theorempostwork{%
    \thm@framedpostwork
    \csname#1@@postwork\endcsname}%
  \newtheorem@i{#1}%
  }
```

\newframedtheorem

```
\def\newframedtheorem#1{%
  \expandafter\xdef\csname#1@@prework\endcsname{%
    \the\theoremprework}%
  \theoremprework{\csname#1@@prework\endcsname\thm@framedprework}%
  \expandafter\xdef\csname#1@@postwork\endcsname{%
    \the\theorempostwork}%
  \theorempostwork{\thm@framedpostwork\csname#1@@postwork\endcsname}%
  \newtheorem@i{#1}%
  }
}% end of option framed ********************************************
```

7.1.9 Generation of Theorem Lists

The generation of lists of theorems, definitions, etc. is based on the `.thm` file. The following macros are 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)

```
\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.

```
\def\thm@@thmcaption#1#2#3#4{%
    \thm@parseforwriting{#2}%
    \let\thm@tmpii\thm@tmp
    \thm@parseforwriting{#4}%
    \edef\thm@t{{\thm@tmpii}{#3}{\thm@tmp}}%
    \addcontentsline{thm}{#1}{\thm@t}}
```

Arguments: ⟨*env*⟩:=`#1` is the internal environment name, ⟨*output_name*⟩:=`#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 915, 916: the command sequence for the output keyword is prepared by `\thm@parseforwriting` (which returns `\thm@tmpii`) and then stored in `\thm@tmpii`.

Line 917: the optional text is also prepared by `\thm@parseforwriting`

Lines 918, 919: 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`

```
921 \def\thm@@thmline@noname#1#2#3#4{%
922             \@dottedtocline{-2}{0em}{2.3em}%
923                    {\protect\numberline{#2}#3}%
924                    {#4}}
```

`\thm@@thmline@name`

```
925 \def\thm@@thmline@name#1#2#3#4{%
926         \@dottedtocline{-2}{0em}{2.3em}%
927                    {#1 \protect\numberline{#2}#3}%
928                    {#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}
```

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

`\thm@lgobble` The following macros are used to ignore entries for theorem sets, that should not occur in a given list:

```
930 \long\def\thm@lgobble@entry#1#2{\ignorespaces}
931 \long\def\thm@lgobble@freetext#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`.

```
932 \def\theoremlistall{%
```

```
    \let\thm@@thmlstart=\relax
    \let\thm@@thmlend=\relax
    \let\thm@@thmline=\thm@@thmline@noname}
```

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

```
\def\theoremlistallname{%
    \let\thm@@thmlstart=\relax
    \let\thm@@thmlend=\relax
    \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`.

```
\def\theoremlistoptional{%
    \let\thm@@thmlstart=\relax
    \let\thm@@thmlend=\relax
    \def\thm@@thmline##1##2##3##4{%
         \ifx\empty ##3%
         \else
            \thm@@thmline@noname{##1}{##2}{##3}{##4}%
         \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`.

```
\def\theoremlistoptname{%
    \let\thm@@thmlstart=\relax
    \let\thm@@thmlend=\relax
    \def\thm@@thmline##1##2##3##4{%
         \ifx\empty ##3%
         \else%
            \thm@@thmline@name{##1}{##2}{##3}{##4}%
         \fi}}
```

`\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.

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

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

```
\def\thm@thml@all{\theoremlistall}
\def\thm@thml@opt{\theoremlistoptional}
\def\thm@thml@optname{\theoremlistoptname}
\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 defininig the macro `\thm@thml@`⟨*type*⟩, which *locally* redefines the commands `\thm@thmlstart`, `\thm@@thmline` and `\thm@@thmlend`.

```
\def\newtheoremlisttype#1#2#3#4{%
  \@ifundefined{thm@thml@#1}%
  {\expandafter\gdef\csname thm@thml@#1\endcsname{%
    \def\thm@@thmlstart{#2}%
    \def\thm@@thmline####1####2####3####4{#3}%
    \def\thm@@thmlend{#4}}%
  }{\PackageError{\basename}{list type #1 already defined}\@eha}}
```

`\renewtheoremlisttype`

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

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

`\thm@definelthm` For each theorem-set, we need to initialize two commands:

- how to typeset entries in the list, `\l@`⟨*theorem-set*⟩. it is called for each theorem when the list is generated.
- how to typeset additional text in the list, `\thm@listdo`⟨*theorem-set*⟩. It is called, when something is to a list with `\addtotheoremfile`.

These macros are initially defined by `\newtheorem` to discard the input by calling `\thm@lgobble@entry` (for actual entries) and `\thm@lgobble@freetext` (for free text added by the user). These macros must be adapted if a package uses another format for `\contentsline` entries in the `.aux` file (e.g., `hyperref`).

```
\def\thm@definelthm#1{%
  \expandafter\gdef\csname l@#1\endcsname{\thm@lgobble@entry}%
  \expandafter\gdef\csname thm@listdo#1\endcsname{\thm@lgobble@freetext}}
```

`\thm@inlistdo` When additional text is added to a theorem list via `\addtotheoremfile`, this is typeset by the following is macro. It simply discards the first argument and strips of the outer brackets from the second one.

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

`\listtheorems` The following macro provides the user interface:

```
980 \def\listtheorems#1{\begingroup
981   \c@tocdepth=-2%
982   \def\thm@list{#1}\thm@processlist
983   \endgroup}
```

Line 980: `#1` is a list of theorem sets, i.e., of the form `Theorem` or `Theorem, Definition, ...`.

Line 981: set `tocdepth` to -2 to assure that the predefined list-types work.

Line 982: store the list of names in `thm@list` and call `\thm@processlist`, which actually generates the list.

\thm@processlist

The file ⟨*jobname*⟩`.thm` contains commands of the form

`\contentsline{`⟨*list-of-theoremsets*⟩`}{{`⟨*header*⟩`}{`⟨*number*⟩`}}{`⟨*page*⟩`}`.

Thus, dependent on which theoremsets should be listed, `\contentsline` must be defined to evaluate the first argument and then to output all arguments, or to discard the second and third one.

This is done as follows: The commands `\l@`⟨*theorem-set*⟩ and `\thm@listdo`⟨*theorem-set*⟩ (which initially were set to ignore everything by `\newtheorem`) are redefined for the theorem sets which should be listed to generate output. `\contentsline` is defined to call `\l@`⟨*theorem-set*⟩, adding a line to the list or ignoring the entry. Since for theorem sets which are not yet known (i.e., if the list is created at the beginning of the document, and the theoremset is only defined later), `\l@`⟨*theorem-set*⟩ is not yet defined, `\contentsline` has to check if the command is defined, otherwise ignore the arguments.

Then, the `.thm` file is processed, evaluating the `\contentsline` commands. After processing the `.thm` file, the mentioned commands are again redefined to discard everything. We need to define the macros globally for dealing with complex, user-defined, list-types.

```
\def\thm@processlist{%
 \begingroup
 \typeout{** Generating table of \thm@list}%
 \def\contentsline##1{%
       \expandafter\@ifundefined{l@##1}%
          {\thm@lgobble@entry}{\csname l@##1\endcsname}}%
 \thm@@thmlstart
 \@for\thm@currentlist:=\thm@list
  \do{%
  \ifx\thm@currentlist\@empty\else
   \expandafter\gdef\csname l@\thm@currentlist\endcsname{\thm@thmline}%
   \expandafter\gdef\csname thm@listdo\thm@currentlist\endcsname{\thm@inlistdo}%
  \fi
  }%
 \@input{\jobname .thm}%
 \thm@@thmlend
 \@for\thm@currentlist:=\thm@list
  \do{%
  \ifx\thm@currentlist\@empty\else
   \expandafter\gdef\csname l@\thm@currentlist\endcsname
           {\thm@lgobble@entry}%
   \expandafter\gdef\csname thm@listdo\thm@currentlist\endcsname
           {\thm@lgobble@freetext}%
  \fi
  }%
 \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 LaTeX-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`.

```
\def\thm@enablelistoftheorems{%
  \begingroup
    \makeatletter
    \if@filesw
      \expandafter\newwrite\csname tf@thm\endcsname%
      \immediate\openout \csname tf@thm\endcsname \jobname.thm\relax%
    \fi
    \@nobreakfalse
  \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@addtheoremline` is called. `\num/nonum@addtheoremline{`⟨*theorem-set*⟩`}{`⟨*entry*⟩`}` calls `\num/nonum@addtheoremline`⟨*theorem-set*⟩`{`⟨*entry*⟩`}` which are defined when ⟨*theorem-set*⟩ is declared (cf. `\@nthm`). These in turn call `\@num/nonum@addtheoremline{`⟨*theorem-set*⟩ `}{`⟨*keyword*⟩`}{`⟨*entry*⟩`}` which write information to the theorem file.

```
\def\addtheoremline{\@ifstar{\nonum@addtheoremline}{\num@addtheoremline}}
\def\nonum@addtheoremline#1{\csname nonum@addtheoremline#1\endcsname}%
\def\num@addtheoremline#1{\csname num@addtheoremline#1\endcsname}%
```

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

Syntax: `\@num/nonum@addtheoremline{` ⟨*theorem-set*⟩`}{`⟨*keyword*⟩`}{`⟨*entry*⟩`}`

```
\def\@nonum@addtheoremline#1#2#3{%
    \thm@parseforwriting{#3}%
    \edef\thm@t{{#2}{}{\thm@tmp}}%
    \addcontentsline{thm}{#1}{\thm@t}}
```

`\@num@addtheoremline`

```
\def\@num@addtheoremline#1#2#3{%
   \thm@parseforwriting{#3}%
    \edef\thm@t{{#2}{\csname the#1\endcsname}{\thm@tmp}}%
    \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.

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

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

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

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

```
\long\def\@@addtotheoremfile[#1]#2{%
   \thm@parseforwriting{#2}%
   \protected@write\@auxout%
      {}{\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.

```
\long\def\theoremlistdo#1#2{\expandafter\@ifundefined{thm@listdo#1}%
     \relax{\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.

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

Theoremlists and Hyperref Since the `hyperref`-package redefines `\contentsline`, some commands are redefined:

1. Let the different versions of `\thm@@thmline@..` take a 5th argument, the one provided by `hyperref`.
2. handle contentsline: restore the normal definition at the beginning of `\thm@processlist` (see there), that calls `l@`⟨*theorem-set*⟩ that in turn calls the adapted commands for typestting the entries (see below). .
3. Let `\thm@lgobble@entry` take one more argument, the one provided by hyperref.
4. Do the hyperlinks manually in the different versions of `\thm@@thmline` as defined by the theoremtypes.

`hyperref`

```
\DeclareOption{hyperref}{% *********************************************
     \def\thm@@thmline@noname#1#2#3#4#5{%
         \ifx\\#5\\%
             \@dottedtocline{-2}{0em}{2.3em}%
                 {\protect\numberline{#2}#3}%
                 {#4}%
         \else
             \ifHy@linktocpage\relax\relax
                 \@dottedtocline{-2}{0em}{2.3em}%
                     {\protect\numberline{#2}#3}%
                     {\hyper@linkstart{link}{#5}{#4}\hyper@linkend}
             \else
                 \@dottedtocline{-2}{0em}{2.3em}%
                     {\hyper@linkstart{link}{#5}{\protect\numberline{#2}#3}%
                       \hyper@linkend}%
                     {#4}%
             \fi
```

```
        \fi}%
    \def\thm@@thmline@name#1#2#3#4#5{%
        \ifx\\#5\\%
            \@dottedtocline{-2}{0em}{2.3em}%
                {#1 \protect\numberline{#2}#3}%
                {#4}
        \else
            \ifHy@linktocpage\relax\relax
                \@dottedtocline{-2}{0em}{2.3em}%
                    {#1 \protect\numberline{#2}#3}%
                    {\hyper@linkstart{link}{#5}{#4}\hyper@linkend}%
            \else
                \@dottedtocline{-2}{0em}{2.3em}%
                    {\hyper@linkstart{link}{#5}%
                      {#1 \protect\numberline{#2}#3}\hyper@linkend}%
                    {#4}%
            \fi
        \fi}
    \def\thm@thmline#1#2#3{\thm@@thmline#1{#2}{#3}}
    \long\def\thm@lgobble@entry#1#2#3{\ignorespaces}
    \def\theoremlistoptional{%
        \let\thm@@thmlstart=\relax
        \let\thm@@thmlend=\relax
        \def\thm@@thmline##1##2##3##4##5{%
             \ifx\empty ##3%
             \else%
                \thm@@thmline@noname{##1}{##2}{##3}{##4}{##5}%
             \fi}}
    \def\theoremlistoptname{%
        \let\thm@@thmlstart=\relax
        \let\thm@@thmlend=\relax
        \def\thm@@thmline##1##2##3##4##5{%
             \ifx\empty ##3%
             \else%
                \thm@@thmline@name{##1}{##2}{##3}{##4}{##5}%
             \fi}}
```

Theorem References and Hyperref

hyperref-thref

When `hyperref` is active, the handling of `thref` described above via the `.aux` file redefinition of `\@newl@bel` is not possible (`hyperref` forces its definitions at `\AtBeginDocument`). Instead, an internal identifier of the form `Theorem.1.1` is used in the `.aux` file for the hypertarget (using the type of the counter; thus when a theorem type uses another counter, this does not give the theorem type itself). The same id is stored in the `.thm` file for the respective theorem. by this, given the id from the `\newlabel` in the `.aux` file, the `.thm` file can be searched for the actual type information.

```
\if@thref
```

```
1097 \def\@firstofthree#1#2#3{#1}%
1098 \def\getKeywordOf#1{%
1099  \let\thm@oldcontentsline\contentsline
1100  \def\contentsline##1##2##3##4{%
1101   \ifthenelse{\equal{#1}{##4}}{\@firstofthree##2}{}%
1102   \ignorespaces}%
1103  \@input{\jobname .thm}%
1104  \let\contentsline\thm@oldcontentsline
1105 }
1106 \def\thm@fmt@hyplabel@i#1#2#3#4#5{%
1107   \getKeywordOf{#4}~\thm@fmt@hyplabel@ii#4}
1108 \def\thm@fmt@hyplabel@ii#1.#2{#2}%
1109 \def\thref#1{%
1110   \expandafter\@setref\csname r@#1\endcsname\thm@fmt@hyplabel@i{#1}}%
1111 \fi % end of \if@thref
1112 }% end of option hyperref *********************************************
```

Lines 1097-1105: given an id `#1` of the form `Theorem.1.1`, scan the `.thm` file for a `\contentsline` whose 4th argument equals the id. If found, the third component of its second argument gives its theorem type.

Lines 1106-1108: this command must have 5 arguments because it is applied to the information stored with `\newlabel` in the `.aux` file. The 4th argument is the id `#4` of the form `Theorem.1.1`.
Get the correct keyword by `\getKeywordOf{#4}` and its number (which is the part following the first ".").

Lines 1109-1110: create a hyperlink via `\@setref` (see `hyperref.sty`): `\@setref` takes three arguments: `r@`⟨*label*⟩ $:= arg_1$ is the information from `\newlabel` in the `.aux` file (consisting of 5 components). The 2nd argument arg_2 must be a command that uses 5 arguments, here `\thm@fmt@hyplabel@i{#1}` as defined in Lines 1106-1108. The 3rd one is the label, and is only used for error messages. `\@setref` then –roughly– applies arg_2 on arg_1.

7.1.10 Auxiliary macros

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

```
\def\thm@meaning#1->#2\relax{#2}% remove "macro: ->"
\long\def\thm@parseforwriting#1{%
    \def\thm@tmp{#1}%
    \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:

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

`\@greek` Accoding to LaTeX-base, this is the internal command for generating lowercase greek numberings.

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

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

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

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

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

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

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

7.1.11 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.

```
\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`.

```
\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.

```
\ifx\thm@noconfig\@undefined
\InputIfFileExists{ntheorem.cfg}%
  {\PackageInfo{\basename}{Local config file ntheorem.cfg used}}%
  {\ifx\thm@usestd\@undefined%
   \else%
     \InputIfFileExists{ntheorem.std}%
      {\PackageInfo{\basename}{Standard config file ntheorem.std used}}{}
   \fi}
\fi
```

7.2 The Standard Configuration

```
\theoremnumbering{arabic}
\theoremstyle{plain}
\RequirePackage{latexsym}
\theoremsymbol{\ensuremath{_\Box}}
```

```
\theorembodyfont{\itshape}
\theoremheaderfont{\normalfont\bfseries}
\theoremseparator{}
\newtheorem{Theorem}{Theorem}
\newtheorem{theorem}{Theorem}
\newtheorem{Satz}{Satz}
\newtheorem{satz}{Satz}
\newtheorem{Proposition}{Proposition}
\newtheorem{proposition}{Proposition}
\newtheorem{Lemma}{Lemma}
\newtheorem{lemma}{Lemma}
\newtheorem{Korollar}{Korollar}
\newtheorem{korollar}{Korollar}
\newtheorem{Corollary}{Corollary}
\newtheorem{corollary}{Corollary}

\theorembodyfont{\upshape}
\newtheorem{Example}{Example}
\newtheorem{example}{Example}
\newtheorem{Beispiel}{Beispiel}
\newtheorem{beispiel}{Beispiel}
\newtheorem{Bemerkung}{Bemerkung}
\newtheorem{bemerkung}{Bemerkung}
\newtheorem{Anmerkung}{Anmerkung}
\newtheorem{anmerkung}{Anmerkung}
\newtheorem{Remark}{Remark}
\newtheorem{remark}{Remark}
\newtheorem{Definition}{Definition}
\newtheorem{definition}{Definition}

\theoremstyle{nonumberplain}
\theoremheaderfont{\scshape}
\theorembodyfont{\normalfont}
\theoremsymbol{\ensuremath{_\blacksquare}}
\RequirePackage{amssymb}
\newtheorem{Proof}{Proof}
\newtheorem{proof}{Proof}
\newtheorem{Beweis}{Beweis}
\newtheorem{beweis}{Beweis}
\qedsymbol{\ensuremath{_\blacksquare}}
\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` begun in October 1996 by a thread asking for a routine for setting endmarks in *de.comp.tex* initiated by Boris Piwinger. Version 0.1 incorporated the main features for setting endmarks automagically 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.
With 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 Schedler)

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 TEXnische 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 hided 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 alltogether.

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:

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):

Donald Arseneau, Giovanni Dore, Oliver Karch, Frank Mittelbach, Gerd Neugebauer, Heiko Oberdiek, Boris Piwinger, Bernd Raichle, Rainer Schöpf, Didier Verna.