An Extension of the LaTeX theorem environment*

Frank Mittelbach
Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56
D-65424 Rüsselsheim
Federal Republic of Germany

August 8, 1999

Abstract

The macros described in this paper yield an extension of the LaTeX theorem mechanism. It is designed is to satisfy the different requirements of various journals. Thus, the layout of the "theorems" can be manipulated by determining a "style". This article describes not only the use, but also the definition, of the necessary macros.

Preface to version 2.2

For LaTeX 2_ε this package did not need any fundamental changes. I only modified the messages generated so that theorem layout styles will show up with the `\listfiles` command and cleaned the section on the New Font Selection Scheme since this is now included in LaTeX.

Preface to version 2.1

This version is identical to 2.0g described in *TUGboat* 10#3 except for some internal defaults which are now set depending on the used font selection scheme.

This was done to avoid unpleasant surprises if the new font selection scheme is in force. For further details see section 3 and [1].

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, and remarks are all instances of "theorems". The "header" of these structures is composed of a label (such as THEOREM or REMARK) and a number which serializes an item in the sequence of items with the same label.

*This file has version number v2.2c, last revised 1995/11/23.

Shortly after the introduction of LaTeX at the Fachbereich Mathematik in Mainz, the desire to manipulate the layout of "theorems" arose. In Mainz, the following two conventions came into general use:

1. The number of the theorem is shown in the margin.
2. There is a line break at the end of the theorem header.

Additionally, some journals require different formats which depend on the "sort of theorem": e.g. often remarks and definitions are set in `\upshape`, while `\itshape` is employed for main theorems.

Confronted with these requirements, a theorem environment was developed in Mainz which allows separate determination of the layout of the "theorems sets", comparable to `\pagestyle`.

2 The user interface

2.1 Defining new theorem sets

`\newtheorem` As in the original LaTeX version, the command `\newtheorem` defines a new "theorem set" or "theorem-like structure". Two required arguments name the new environment 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}{bar3}[section]` The theorem set `foo3` (printed name `bar3`) 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.

`\theoremstyle` Additionally, the command `\theoremstyle` can define the layout of various, or all, theorem sets. It should be noted that any theorem set defined by `\newtheorem` is typeset in the `\theoremstyle` that is current at the time of the definition. Thus, the following

```
\theoremstyle{break}        \newtheorem{Cor}{Corollary}
\theoremstyle{plain}        \newtheorem{Exa}{Example}[section]
```

leads to the result that the set `Cor` is formatted in the style `break`, while the set `Exa` and all the following ones are formatted in the style `plain`, unless another `\theoremstyle` follows. Since the definitions installed by `\newtheorem` are global, one also can limit `\theoremstyle` locally by grouping braces.

`\theorembodyfont` The choice of the font for the theorem body is completely independent of the chosen `\theoremstyle`; this has proven to be very advantageous. For example,

```
{\theorembodyfont{\upshape}          \newtheorem{Rem}{Remark}}
```

defines a theorem set Rem, which will be set in \upshape in the current layout (which in our example is plain). As with \theoremstyle, the \theorembodyfont chosen is that current at the time of \newtheorem. If \theorembodyfont is not specified or one defines \theorembodyfont{}, then the font used will be that defined by the \theoremstyle.

\theoremheaderfont

It is also possible to customize the font used for the theorem headers. This is, however, a global declaration, and therefore there should be at most one \theoremheaderfont declaration in the preamble.[1]

\theorempreskipamount
\theorempostskipamount

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

Since the definition of theorem sets should—most sensibly—be placed in the preamble, we only allow installation there. It is therefore possible to release the memory used here after \begin{document}, in order to make room for other applications.

2.2 Existing theorem styles

The following theorem styles exist to date:

- **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.
- **marginbreak** The theorem number is set in the margin, and there is a line break as in break.
- **changebreak** Like break, but with header number and text interchanged.
- **change** Header number and text are interchanged, without a line break.
- **margin** The number is set in the left margin, without a line break.

All styles (except plain) select \slshape as the default \theorembodyfont.

2.3 Examples

Given the above theorem sets Cor, Exa and Rem, suppose that the preamble also contains the declarations:

```
\theoremstyle{marginbreak}    \newtheorem{Lem}[Cor]{Lemma}
\theoremstyle{change}
\theorembodyfont{\itshape}        \newtheorem{Def}[Cor]{Definition}

\theoremheaderfont{\scshape}
```

Then the following are some typical examples of the typeset output resulting from their use.

[1] If it is actually necessary to have different header fonts, one has to define new theorem styles (substituting the desired font) or specify the information directly in the \newtheorem declaration (the unclean variant).

Corollary 1
This is a sentence typeset in the theorem environment `Cor`.

Example 2.1 *This is a sentence typeset in the theorem environment* `Exa`.

Remark 1 This is a sentence typeset in the theorem environment `Rem`.

2 Lemma (Ben User)
This is a sentence typeset in the theorem environment `Lem`.

3 Definition (Very impressive Definition) *This is a sentence typeset in the theorem environment* `Def`.

The last two examples show the effect of the optional argument to a theorem environment (it is the text typeset in parentheses).

3 Special Considerations

Theoremheader and body are implemented as a unit. This means that the `\theoremheaderfont` will inherit characteristics of the `\theorembodyfont` in LaTeX 2ε. 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 one should set the `\theoremheaderfont` to something like

```
\theoremheaderfont{\normalfont\bfseries}
```

i.e. supplying all necessary font informations explicitly.

4 Acknowledgements

The publication of this set of macros was only possible with the help of Christina Busse (translating the manuscript into English), Joachim Pense (playing the rôle of typist), Chris Rowley (looking everything over) and many others providing useful suggestions.

5 The documentation driver file

The next bit of code contains the documentation driver file for TeX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the `docstrip` program. Since it is the first code in the file one can alternatively process this file directly with LaTeX 2ε to obtain the documentation.

```
⟨*driver⟩
\documentclass{ltxdoc}

\usepackage{theorem}

⟨+driver⟩% The next few lines define theorem sets which are used
⟨+driver⟩% in the example section of the documentation.

```

```
\theoremstyle{break}       \newtheorem{Cor}{Corollary}
\theoremstyle{plain}       \newtheorem{Exa}{Example}[section]
{\theorembodyfont{\upshape}\newtheorem{Rem}{Remark}}
\theoremstyle{marginbreak} \newtheorem{Lem}[Cor]{Lemma}
\theoremstyle{change}
\theorembodyfont{\itshape} \newtheorem{Def}[Cor]{Definition}

\theoremheaderfont{\scshape}

\RecordChanges

\begin{document}
   \DocInput{theorem.dtx}
\end{document}
⟨/driver⟩
```

6 Definition of the Macros

If the file has been loaded before, we abort immediately. If not the package announces itself (this is actually done at the very top if the file—the way it is done isn't good style so don't copy it).

```
⟨*package⟩
%\@ifundefined{theorem@style}{}{\endinput}
%\def\FMithmInfo{1995/11/19 v2.2b Theorem extension package (FMi)}
%\ProvidesPackage{theorem}[\FMithmInfo]
```

6.1 Definition of theorem styles and fonts

All the definitions in this file are done globally to allow inputting this file inside a group.

`\theoremstyle`

Before a theorem style can be installed, the chosen style must be known. For that reason, we must test to see that `\th@`⟨*style*⟩ is known or, more precisely, that it is different from `\relax`. If the style is not known then `\th@plain` is used.

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

We save the theorem style to be used in the token register `\theorem@style`.

```
      {\theorem@style{#1}}%
```

Now we "evaluate" the theorem style: this means, we call the macro `\th@`⟨*style*⟩ which will activate the relevant definitions which are contained in a separate file. This is done in a group to suppress changes to the current font. This could otherwise pose problems together with the new font selection scheme[2] if the `\th@`⟨*style*⟩ is evaluated a second time.

```
      \begingroup
        \csname th@\the\theorem@style \endcsname
      \endgroup}
```

[2]When I printed the original article using the new font selection scheme I ended with a document with slanted typefaces (text headings and all) simply because one of the theorem styles used `\sl` at toplevel.

`\@begintheorem` `\@opargbegintheorem`

We reset `\@begintheorem` and `\@opargbegintheorem` to `\relax` since these commands are no longer necessary at toplevel. This will save a few tokens.

```
\global\let\@begintheorem\relax
\global\let\@opargbegintheorem\relax
```

`\theorem@style`

Obviously the `token` register used above has to be allocated. To assure the utmost compatibility with the original LaTeX definition, we set the default theorem style to `plain`, which implements the usual LaTeX convention.

```
\newtoks\theorem@style
\global\theorem@style{plain}
```

`\theorembodyfont` `\theorem@bodyfont`

If the `\theorembodyfont` is set by the user then it should not interact with the default font set in the theorem style. When the new font selection is in force this may happen if, for example, the default is `\itshape` and the new `\theorembodyfont` is `\sffamily`. So we add a `\reset@font` command in front of the user definition.

```
\gdef\theorembodyfont#1{%
```

We check if the argument supplied is empty and if so put nothing into the `\theorem@bodyfont` token register to allow for `\theorembodyfont{}` as a mean of using the default of the current `\theoremstyle`.

```
    \def\@tempa{#1}%
    \ifx\@tempa\@empty
     \theorem@bodyfont{}%
    \else
     \theorem@bodyfont{\reset@font#1}%
    \fi
    }
\newtoks\theorem@bodyfont
\global\theorem@bodyfont{}
```

`\theoremheaderfont`

The font for the theorem headers is handled differently because this definition applies to all theorem styles.

```
\gdef\theoremheaderfont#1{\gdef\theorem@headerfont{#1}%
```

After using the macro once it is redefined to produce an error message.

```
        \gdef\theoremheaderfont##1{%
         \typeout{\string\theoremheaderfont\space should be used
                  only once.}}}
```

`\theorem@headerfont`

To set the `\theorem@headerfont` default we first test if the new fontselection scheme is in force.

```
\ifx\upshape\undefined
```

If not we define it to expand into `\bfseries`. We don't use `\let` just in case a following style option redefines this macro.

```
\gdef\theorem@headerfont{\bfseries}
```

Otherwise we reset the current shape before calling `\bfseries`.

```
\else \gdef\theorem@headerfont{\normalfont\bfseries}\fi
```

`\th@plain` `\th@break` `\th@marginbreak` `\th@changebreak` `\th@change` `\th@margin`

The different styles are defined in macros such as `\th@plain`. Since memory space is precious in "non-Big-versions", we have to avoid offering too many unused definitions. Therefore we define these styles in separate files that can be loaded on demand. Thus the commands themselves only load these files. We use `\@input@` a LaTeX 2_{ε} internal command that ensures that the file will be listed with `\listfiles`

```
\gdef\th@plain{\@input@{thp.sty}}
\gdef\th@break{\@input@{thb.sty}}
\gdef\th@marginbreak{\@input@{thmb.sty}}
\gdef\th@changebreak{\@input@{thcb.sty}}
\gdef\th@change{\@input@{thc.sty}}
\gdef\th@margin{\@input@{thm.sty}}
```

This list will be expanded when new styles become available. For testing, just append new theorem substyles as document options.

6.2 Definition of a new theorem set

As already pointed out, a new theorem environment can be defined in three different ways:

```
\newtheorem{Lem}{Lemma}
\newtheorem{Lem}{Lemma}[section]
\newtheorem{Lem}[Theorem]{Lemma}
```

The function of the macro `\newtheorem` is to recognize these cases and then to branch into one of the three macros `\@ynthm`, `\@xnthm` or `\@othm`. This mechanism is adopted unchanged from [2]; the essential point here is that, for example, in the second case, the arguments `Lem`, `Lemma` and `section` are passed over to the macro `\@xnthm`.

We inspect this case first because the others present fewer problems, and thus are easily derived from this one.

`\@xnthm` For our example arguments, the macro `\@xnthm` must fulfill the following:

- Define a new LaTeX-counter 'Lem'
- reset this counter within a `\section`
- define the macro `\theLem`
- define the environment macros `\Lem` and `\endLem` using the current `\theoremstyle` and `\theorem@bodyfont`.

Obviously, all this should happen only if the first argument of `\@xnthm` (i.e. `Lem` in our example) is chosen so as not to conflict with any previously defined commands or environments. This test is performed by the LaTeX macro `\@ifdefinable`.

```
\gdef\@xnthm#1#2[#3]{\expandafter\@ifdefinable\csname #1\endcsname
```

Therefore, the first argument of `\@ifdefinable` is the expansion (in the example, `\Lem`) of `\csname#1\endcsname`. The second argument is executed only if the test has been completed successfully.

```
    {%
```

Now we define the new counter. The names of the LaTeX macros employed should speak for themselves:

```
    \@definecounter{#1}\@newctr{#1}[#3]%
```

Using `\@newctr` will give a proper error message if the counter in `#3` is not defined. In defining '`\theLem`' we must generate the desired macro name by use of `\expandafter` and `\csname`.

```
      \expandafter\xdef\csname the#1\endcsname
```

An `\xdef` is used in order to make the definition global, and to ensure that it contains the replacement texts of `\@thmcountersep` and `\@thmcounter`.[3] However, not everything should be expanded. For example, it saves space to use `\thesection` instead of its—at times—lengthy expansion.

```
        {\expandafter \noexpand \csname the#3\endcsname
         \@thmcountersep \@thmcounter{#1}}%
```

Thus with the defaults of LaTeX, `\theLem` would be replaced by the command sequence `\thesection.\arabic{Lem}`.

We will now look at the definition of the macro which is executed at the beginning of the actual environment (in our example this macro is `\Lem`). It should be noted that we use an "`\expandafter` trick" to expand only certain parts of the replacement text at the time of the definition.

```
      \def\@tempa{\global\@namedef{#1}}%
      \expandafter \@tempa \expandafter{%
```

First, the macro that contains the current definitions of `\@begintheorem` and `\@opargtheorem` should be called up. The name of this macro—as is already known—has the form `\th@`⟨*theorem style*⟩; therefore, it must be called by

```
        \csname th@\the \theorem@style
```

In addition the default theorem font should be changeable, i.e. we have to insert the contents of `\theorem@bodyfont`. For that reason, we expand even further, beyond `\endcsname`, and thus insert the contents of the token register `\theorem@bodyfont` in the replacement text.

```
              \expandafter \endcsname \the \theorem@bodyfont
```

Now it is time to call the macro `\@thm` which takes over the further processing. It has two arguments: the current counter name (in our example, `Lem`), and the text of the label (in our example, `Lemma`).

```
      \@thm{#1}{#2}}%
```

With this, the 'sub-definition' is complete. The macro `\@endtheorem` ends a theorem environment and is, so far, nothing but an `\endtrivlist`. (Hence it is defined globally, and not within the theorem styles.[4]) Therefore, we can set it equivalent to the macro that ends the theorem set (in our example, `\endLem`). However, if some day theorem styles exist that do change `\@endtheorem`, we would have to use the commented-out line instead.

```
      \global \expandafter \let \csname end#1\endcsname \@endtheorem
%     \global\@namedef{end#1}{\@endtheorem}%
```

With these commands all the required definitions are employed, unless the test `\@ifdefinable` has failed. Therefore, we end the second argument of this macro and with it the definition of `\@xnthm`.

```
    }}
```

[3] These two macros can be defined by the document style. Their default values produce a '`.`' as separation and an arabic representation of the number.

[4] This has to be changed as soon as theorem styles that change `\@endtheorem` exist. In such a case, all existing styles must be changed as well since they will have to reset the macro.

\@ynthm The definition of \@ynthm is completely analogous. In this case the new counter that is defined is not reset within another counter; thus the definition of \the... is simplified:

```
\gdef\@ynthm#1#2{\expandafter\@ifdefinable\csname #1\endcsname
  {\@definecounter{#1}%
   \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
```

The rest of the definition corresponds literally to that of \@xnthm:

```
    \def\@tempa{\global\@namedef{#1}}\expandafter \@tempa
     \expandafter{\csname th@\the \theorem@style \expandafter
     \endcsname \the\theorem@bodyfont \@thm{#1}{#2}}%
    \global \expandafter \let \csname end#1\endcsname \@endtheorem}}
```

\@othm The definition of \@othm does not contain anything new.

```
\gdef\@othm#1[#2]#3{%
```

We do not define a new counter but instead use one that has already been defined. Thus the only definition we need is that of this pseudo-counter (i.e. \the⟨*env. name*⟩). First we check if #2 corresponds to a known counter name.

```
  \expandafter\ifx\csname c@#2\endcsname\relax
   \@nocounterr{#2}%
  \else
   \expandafter\@ifdefinable\csname #1\endcsname
   {\expandafter \xdef \csname the#1\endcsname
     {\expandafter \noexpand \csname the#2\endcsname}%
```

All other parts of the definition can be adopted from \@xnthm. We have to remember, though, that in this case the name of the current counter and the theorem label have moved to the second and third arguments.

```
    \def\@tempa{\global\@namedef{#1}}\expandafter \@tempa
     \expandafter{\csname th@\the \theorem@style \expandafter
     \endcsname \the\theorem@bodyfont \@thm{#2}{#3}}%
    \global \expandafter \let \csname end#1\endcsname \@endtheorem}%
  \fi}
```

6.3 Macros that are employed in a theorem environment

\@thm The macro \@thm has to increase the current counter. Then, depending on whether the environment has (or does not have) an optional argument, it has to branch into either \@begintheorem or \@opargtheorem.

```
\gdef\@thm#1#2{\refstepcounter{#1}%
```

Now we start a trivlist environment, and give \@topsep and \@topsepadd the values of the skip registers \theorempreskipamount and \theorempostskipamount. The value in \@topsep is the vertical space that is inserted by the first (and only) \item in our \trivlist whilst \@topsepadd is inserted by \@endparenv at the end of that trivlist environment. By using these registers, we obtain the desired space around a theorem environment.

```
  \trivlist
  \@topsep \theorempreskipamount               % used by first \item
  \@topsepadd \theorempostskipamount           % used by \@endparenv
```

Now we have to test whether an optional argument has been given.

```
  \@ifnextchar [%
```

If there is an optional argument, we will call `\@ythm`, and move the arguments read back into the input stream.

```
    {\@ythm{#1}{#2}}%
```

If not, we call `\@begintheorem`. Its first argument is the name of the theorem set (hence the second argument of `\@thm`). Its second argument is the macro that produces the current number.

```
    {\@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}}
```

`\@xthm` `\@ythm` Both these macros were originally called by `\@thm`. We do not need `\@xthm` anymore, hence we reset it to `\relax`. The definition of `\@ythm` has not changed at all from its definition in LaTeX. In order to make the macros easier to understand, we will nevertheless present it (commented out).

```
\global\let\@xthm\relax
% \def\@ythm#1#2[#3]{\@opargbegintheorem{#2}{\csname
%             the#1\endcsname}{#3}\ignorespaces}
⟨/package⟩
```

The primitive `\ignorespaces` in `\@ythm` and `\@thm` is needed to remove the spaces between the `\begin{...}` and the actual text.

6.4 Definition of the theorem substyles

As already pointed out, the theorem substyles, defined below, are only loaded when necessary. Note that all these substyles, except `plain`, have `\slshape` as the default body font.

6.4.1 The `plain` style

As the following macros use `@`, we have to locally set the `\catcode` of this symbol to "letter". This happens within a group, so that we do not have to worry about which `\catcode` that symbol had before.

```
⟨*thp⟩
\begingroup \makeatletter
```

Since we are now within a group, we must make all definitions globally. First we make sure that `theorem.sty` is loaded. This will allow us to use this file as a document style option without having to call `theorem` itself as an option. At the same time, we assure that at least version 2 is loaded, since `\theorem@style` was not defined in earlier versions.

```
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
\ProvidesFile{thp.sty}
              [\FMithmInfo]
```

`\th@plain` `\@begintheorem` `\@opargbegintheorem` `\theoremstyle{plain}` corresponds to the original definition, except that the distances to the surrounding text are determined by the parameters `\theorempreskipamount` and `\theorempostskipamount`. First we set the default body font.

```
\gdef\th@plain{\normalfont\itshape
```

Then we define `\@begintheorem` and `\@opargbegintheorem`. These two macros define how the header of a theorem is typeset. `\@opargbegintheorem` will be called if a theorem environment with an optional argument is encountered; otherwise, the header is constructed by calling `\@begintheorem`. If one of these macros is executed, we are within a trivlist environment started by `\@thm`. So the theorem header is produced with an `\item` command.

Instead of specifying the header font directly, all standard theorem styles use the `\theorem@headerfont` macro to allow customization. The extra space (`\labelsep`) is necessary because of problems in the trivlist environment.

```
\def\@begintheorem##1##2{%
      \item[\hskip\labelsep \theorem@headerfont ##1\ ##2]}%
```

The definition of `\@opargbegintheorem` is completely analogous. The only difference is the fact that there exists a third argument (which is the optional parameter of the environment and contains additional information about the theorem). Customarily we enclose it in parentheses.

```
\def\@opargbegintheorem##1##2##3{%
  \item[\hskip\labelsep \theorem@headerfont ##1\ ##2\ (##3)]}}
```

We conclude with an `\endgroup` to restore the `\catcode` of `@`.

```
\endgroup
⟨/thp⟩
```

6.4.2 The break style

This style option is stored in the file `thb.sty`. For the next two lines see the documentation for `\th@plain` on page 10.

```
⟨*thb⟩
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
\ProvidesFile{thb.sty}
             [\FMithmInfo]
```

`\th@break`

`\theoremstyle{break}` produces a line break after the name of the theorem. The font is `\slshape`. Hence, we define `\th@break` as follows:

```
\gdef\th@break{\normalfont\slshape
  \def\@begintheorem##1##2{\item[%
```

We run into the following problem: it is not possible to create the header with `\item[`⟨*title*⟩`]` and then start a new line by, for example, `\mbox{}\\`. Such a definition will fail whenever a list environment follows immediately. With the above construction, the `\mbox{}` causes the switch `@inlabel` (cf. definition of `\list` and `\trivlist` in [2]) to be set to `false` and so the following list will insert additional vertical space (`\topskip`). This is quite annoying. Therefore, we create the line break within the `\item`. In order to ensure that the text will begin at the proper position in the following line, we simply pretend that the label does not take any room.[5]

```
      \rlap{\vbox{\hbox{\hskip \labelsep\theorem@headerfont ##1\ ##2}%
                  \hbox{\strut}}}]}%
```

Again, the definition of `\@opargbegintheorem` is completely analogous.

```
\def\@opargbegintheorem##1##2##3{%
  \item[\rlap{\vbox{\hbox{\hskip \labelsep \theorem@headerfont
                     ##1\ ##2\ (##3)}%
```

[5]This will lead to problems whenever very high symbols occurring in the line tower into the heading. So, something else has to be done here sometime.

```
                    \hbox{\strut}}}]}}
\endgroup
⟨/thb⟩
```

6.4.3 The changebreak style

```
% This style option is stored in the file |thcb.sty|.
%    \begin{macrocode}
⟨*thcb⟩
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
\ProvidesFile{thcb.sty}
             [\FMithmInfo]
```

`\th@changebreak` The `change-break` theorem style is like `break` but with interchange of theorem name and theorem number. Thus we define `\th@changebreak` as follows:

```
\gdef\th@changebreak{\normalfont\slshape
  \def\@begintheorem##1##2{\item
       [\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont ##2\ ##1}%
                    \hbox{\strut}}}]}%
\def\@opargbegintheorem##1##2##3{%
  \item[\rlap{\vbox{\hbox{\hskip\labelsep \theorem@headerfont
                     ##2\ ##1\ (##3)}%
                    \hbox{\strut}}}]}}
\endgroup
⟨/thcb⟩
```

6.4.4 The change style

This style option is stored in the file `thc.sty`.

```
⟨*thc⟩
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
\ProvidesFile{thc.sty}
             [\FMithmInfo]
```

`\th@change` The `change` theorem style corresponds to the `change break` style without a line-break after the header. To say it in another way, it's the same as the `plain` style but with number and name interchanged and `\slshape` as the default font.

```
\gdef\th@change{\normalfont\slshape
  \def\@begintheorem##1##2{\item
       [\hskip\labelsep \theorem@headerfont ##2\ ##1]}%
\def\@opargbegintheorem##1##2##3{%
  \item[\hskip\labelsep \theorem@headerfont ##2\ ##1\ (##3)]}}
\endgroup
⟨/thc⟩
```

6.4.5 The marginbreak style

This style option is the one used most often at Mainz. It is saved in the file `thmb.sty`.

```
⟨*thmb⟩
\begingroup \makeatletter
```

```
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
\ProvidesFile{thmb.sty}
              [\FMithmInfo]
```

`\th@marginbreak` The `margin break` style is nearly the same as the `change break` style. The only difference is the placement of the theorem number. We use `\llap` to place it in the left margin.

In this style `\labelsep` denotes the separation between the number and the text.

```
\gdef\th@marginbreak{\normalfont\slshape
  \def\@begintheorem##1##2{\item
       [\rlap{\vbox{\theorem@headerfont
                    \hbox{\llap{##2}\hskip\labelsep ##1}%
                    \hbox{\strut}}}]}%
\def\@opargbegintheorem##1##2##3{%
  \item[\rlap{\vbox{\theorem@headerfont
                    \hbox{\llap{##2}\hskip\labelsep ##1\ (##3)}%
                    \hbox{\strut}}}]}}
\endgroup
⟨/thmb⟩
```

6.4.6 The `margin` style

This style option is stored in the file `thm.sty`.

```
⟨*thm⟩
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
\ProvidesFile{thm.sty}
              [\FMithmInfo]
```

`\th@margin` Again this is only a variant of the theorem styles described above without any new ideas.

```
\gdef\th@margin{\normalfont\slshape
  \def\@begintheorem##1##2{\item
       [\theorem@headerfont \llap{##2}\hskip\labelsep ##1]}%
\def\@opargbegintheorem##1##2##3{%
  \item[\theorem@headerfont \llap{##2}\hskip\labelsep ##1\ (##3)]}}
\endgroup
⟨/thm⟩
```

6.5 Final Definitions

`\theorempreskipamount` `\theorempostskipamount` The skip parameters that regulate the vertical empty space before and after the `theorem` environment have to be allocated as well.

```
⟨*package⟩
\newskip\theorempreskipamount
\newskip\theorempostskipamount
```

Since we have used the same values for all theorem sets, we now can assign them.

```
\global\setlength\theorempreskipamount{12pt plus 5pt minus 3pt}
\global\setlength\theorempostskipamount{8pt plus 3pt minus 1.5pt}
```

`\@endtheorem` The same holds for the macro `\@endtheorem`, which ends a `theorem` environment. Since it is the same for all theorem sets, it is removed from the macros `\th@`⟨*style*⟩. It simply ends the `trivlist` environment, which was begun in `\@thm`.

```
\global\let\@endtheorem=\endtrivlist
```

`\@preamblecmds` All macros defined above are to be used only in the preamble. Therefore, we insert them in `\@preamblecmds` which will disable them at begin document. This is done by the internal LaTeX 2_ε command `\@onlypreamble`.

```
\@onlypreamble\@xnthm
\@onlypreamble\@ynthm
\@onlypreamble\@othm
\@onlypreamble\newtheorem
\@onlypreamble\theoremstyle
\@onlypreamble\theorembodyfont
\@onlypreamble\theoremheaderfont
```

Finally we declare the `plain` theorem style to be the default.

```
\theoremstyle{plain}
⟨/package⟩
```

References

[1] M. Goossens, F. Mittelbach and A. Samarin. The LaTeX Companion. Addison-Wesley, Reading, Massachusetts, 1994.

[2] Lamport, Leslie. `latex.tex`, version 2.09, date Feb. 1990.

Change History