Une extension de l'environment theorem de LaTeX *

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

Traduction française par Jean-Pierre Drucbert

31 juillet 2016

Résumé

Les commandes décrites dans cette documentation fournissent une extension du mécanisme des théorèmes de LaTeX. Cette extension est conçue pour satisfaire les exigences de divers journaux. Donc la présentation des « théorèmes » peut être manipulée en déterminant un « style ». Les commandes nécessaires sont décrites ci-dessous.

Préface à la version 2.2

Pour pouvoir être utilisé avec LaTeX 2_ε, ce package n'a pas eu besoin de changements fondamentaux. J'ai simplement modifié les messages engendrés pour que les styles d'affichage de théorèmes apparaissent avec la commande `\listfiles`, et j'ai purgé la section sur le nouveau schéma de sélection des fontes (NdT : New Font Selection Scheme, ou NFSS, en anglais) parce qu'il fait maintenant partie de LaTeX.

Préface à la version 2.1

Cette version est identique à la version 2.0g décrite dans *TUGboat* 10#3 sauf pour quelques valeurs par défaut, qui dépendent maintenant du schéma de sélection des fontes utilisé.

Cela a été fait pour éviter des surprises désagréables si le nouveau schéma de sélection des fontes est en vigueur. Lire la section **??** et [**?**] pour les détails.

*Ce fichier a le numéro de version v2.2c, révisé le 1995/11/23.

1 Introduction

Pour nos besoins présents, les « théorèmes » sont des énonciations étiquetées, souvent détachées du texte principal par un espace supplémentaire et un changement de fonte. Les théorèmes, corollaires, conjectures, définitions et remarques sont tous des instances de « théorèmes ». L'« en-tête » de ces structures est composé d'une étiquette (telle que Théorème ou Remarque) et d'un numéro qui « sérialise » un item dans la séquence des items de même étiquette.

Peu après l'introduction de LaTeX à la « Fachbereich Mathematik » à Mainz (Mayence), le désir de pouvoir manipuler la présentation des « théorèmes » s'est fait sentir. À Mainz, les deux conventions suivantes sont devenues d'un usage général :

1. Le numéro d'un théorème est placé dans la marge.
2. Il y a une coupure de ligne à la fin de l'en-tête du théorème.

De plus, certains journaux demandent des formats différents selon le « genre de théorème » : par exemple, les remarques et définitions sont souvent composées en `\upshape`, tandis que `\itshape` est employé pour les théorèmes principaux.

Face à ces exigences, un environnement theorem a été développé à Mainz pour permettre la détermination séparée de la présentation des « ensembles de théorèmes », d'une façon comparable à `\pagestyle`.

2 Interface utilisateur

2.1 Définir de nouveaux ensembles de théorèmes

`\newtheorem` Comme dans la version LaTeX originale, la commande `\newtheorem` définit un nouvel « ensemble de théorèmes » ou une nouvelle « structure de type théorème ». Deux arguments obligatoires nomment le nouvel environnement et donnent le texte à composer avec chaque instance du nouvel « ensemble », alors qu'un argument optionnel détermine comment cet « ensemble » est numéroté :

`\newtheorem{foo}{bar}` L'ensemble de théorèmes `foo` (dont le nom est `bar`) utilise son propre compteur.

`\newtheorem{foo2}[foo]{bar2}` L'ensemble de théorèmes `foo2` (nom imprimé `bar2`) utilise le même compteur que l'ensemble de théorèmes `foo`.

`\newtheorem{foo3}{bar3}[section]` L'ensemble de théorèmes `foo3` (nom imprimé `bar3`) est numéroté à l'intérieur de chaque valeur du compteur `section`, c'est-à-dire qu'à chaque nouvelle `\section` la numérotation recommence à 1, et que le numéro imprimé est formé du numéro de section et du numéro du théorème lui-même.

`\theoremstyle` De plus, la commande `\theoremstyle` peut définir la présentation (au sens mise en page) de divers ensembles de théorèmes, ou de tous. Il faut noter que tout ensemble de théorèmes défini par `\newtheorem` sera composé dans le `\theoremstyle` qui est celui courant au moment de la définition. Donc les définitions suivantes

```
\theoremstyle{break}        \newtheorem{Cor}{Corollaire}
\theoremstyle{plain}        \newtheorem{Exa}{Exemple}[section]
```

font finalement que l'ensemble `Cor` est mis en page dans le style `break`, alors que l'ensemble `Exa` et tous les suivants sont mis en page dans le style `plain`, à moins qu'une autre commande `\theoremstyle` suive. Bien que les définitions installées par `\newtheorem` soient globales, on peut aussi limiter la portée de `\theoremstyle` localement par des accolades de groupement.

`\theorembodyfont` Le choix de la fonte pour le corps du théorème est complètement indépendant du `\theoremstyle` choisi ; ceci s'est révélé très avantageux. Par exemple,

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

définit un ensemble de théorèmes `Rem`, qui sera composé en `\upshape` dans la présentation courante (qui dans notre exemple est `plain`). Comme pour `\theoremstyle`, la fonte `\theorembodyfont` choisie est celle courante lors de la commande `\newtheorem`. Si la fonte `\theorembodyfont` n'est pas spécifiée ou si vous définissez `\theorembodyfont{}`, alors la fonte utilisée sera celle définie par le `\theoremstyle`.

`\theoremheaderfont` Il est aussi possible d'adapter la fonte utilisée pour les en-têtes de théorèmes. Ceci est, cependant, une déclaration globale et il devrait donc y avoir au plus une déclaration `\theoremheaderfont` dans le préambule [1].

`\theorempreskipamount` `\theorempostskipamount` Deux paramètres supplémentaires affectent l'espace vertical autour des environnements theorem : `\theorempreskipamount` et `\theorempostskipamount` définissent respectivement l'espacement avant et après un tel environnement. Ces paramètres (qui sont des longueurs) s'appliquent à tous les ensembles de théorèmes et peuvent être manipulés par les commandes sur longueurs habituelles. Ce sont des longueurs élastiques, ('skips'), et donc peuvent comporter des parties `plus` et `minus`.

Comme la définition des ensembles de théorèmes devrait — raisonnablement — être placée dans le préambule, nous ne la permettons que là. Il est donc possible de libérer la mémoire utilisée pour cela après `\begin{document}`, pour laisser de la place à d'autres applications.

2.2 Styles de théorèmes existants

À ce jour les styles suivants de théorèmes sont disponibles :

`plain` Ce style de théorème émule la définition originale de LaTeX, sauf qu'en plus les deux paramètres `\theorem{post,pre}skipamount` sont utilisés.

`break` Dans ce style, l'en-tête du théorème est suivi d'une coupure de ligne.

`marginbreak` Le numéro du théorème est placé dans la marge, et il y a une coupure de ligne comme dans le style `break`.

1. S'il est en fait nécessaire d'avoir différentes fontes pour les en-têtes, vous devez définir de nouveaux styles de théorèmes (en substituant la fonte désirée) ou spécifier l'information directement dans la déclaration `\newtheorem` (la méthode impropre).

`changebreak` Comme `break`, mais avec numéro et texte de l'en-tête intervertis.

`change` Numéro et texte de l'en-tête intervertis, sans coupure de ligne.

`margin` Le numéro est placé dans la marge, sans coupure de ligne.

Tous les styles (sauf `plain`) choisissent `\slshape` comme `\theorembodyfont` par défaut.

2.3 Exemples

Étant donnés les ensembles de théorèmes ci-dessus `Cor`, `Exa` et `Rem`, supposons que le préambule contienne aussi les déclarations :

```
\theoremstyle{marginbreak}    \newtheorem{Lem}[Cor]{Lemme}
\theoremstyle{change}
\theorembodyfont{\itshape}    \newtheorem{Def}[Cor]{D\'efinition}
\theoremheaderfont{\scshape}
```

Alors voici quelques exemples typiques de la sortie composée résultant de leur utilisation.

Corollaire 1
Ceci est une phrase composée dans l'environnement theorem `Cor`*.*

Exemple 2.1 *Ceci est une phrase composée dans l'environnement theorem* `Exa`*.*

Remarque 1 Ceci est une phrase composée dans l'environnement theorem `Rem`.

2 Lemme (Ben User)
Ceci est une phrase composée dans l'environnement theorem `Lem`*.*

3 Définition (Définition Très Impressionnante) *Ceci est une phrase composée dans l'environnement theorem* `Def`*.*

Les deux derniers exemples montrent l'effet de l'argument optionnel sur un environnement theorem (c'est le texte composé entre parenthèses).

3 Considérations spéciales

L'en-tête et le corps du théorème forment une même unité dans la réalisation. Ceci signifie que la fonte `\theoremheaderfont` héritera des caractéristiques de la fonte `\theorembodyfont` avec LaTeX 2$_\varepsilon$. Donc par exemple, si `\theorembodyfont` est `\itshape` et `\theoremheaderfont` est `\bfseries`, la fonte sélectionnée pour l'en-tête aura les caractéristiques 'bold extended italic' (italique gras-étendu). Si ce n'est pas ce que vous voulez, vous pouvez définir `\theoremheaderfont` avec quelque chose comme

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

c'est-à-dire fournir explicitement toutes les informations nécessaires sur les fontes.

4 Remerciements

La publication de cet ensemble de commandes n'a été possible que grâce à l'aide de Christina Busse (traduisant le manuscrit en anglais), Joachim Pense (jouant le rôle de typographe), Chris Rowley (chapeautant tout) et de beaucoup d'autres qui ont apporté des suggestions très intéressantes.

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 2_ε to obtain the documentation.

```
⟨*driver⟩
\documentclass{ltxdoc}

\usepackage{theorem}
⟨/driver⟩
⟨*driver⟩

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

\theoremstyle{break}       \newtheorem{Cor}{Corollaire}
\theoremstyle{plain}       \newtheorem{Exa}{Exemple}[section]
{\theorembodyfont{\upshape}\newtheorem{Rem}{Remarque}}
\theoremstyle{marginbreak} \newtheorem{Lem}[Cor]{Lemme}
\theoremstyle{change}
\theorembodyfont{\itshape} \newtheorem{Def}[Cor]{D\'efinition}

\theoremheaderfont{\scshape}

\RecordChanges

\begin{document}
   \DocInput{f-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}
```

`\@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{%
```

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.

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 [**?**] ; 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`.

```
65 \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.

```
66    {%
```

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

```
67      \@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`.

```
68      \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.

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

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.

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`.

```
}}
```

`\@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}}%
```

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.

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.

```
104    {\@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).

```
105 \global\let\@xthm\relax
106 % \def\@ythm#1#2[#3]{\@opargbegintheorem{#2}{\csname
107 %            the#1\endcsname}{#3}\ignorespaces}
108 ⟨/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.

```
109 ⟨*thp⟩
110 \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.

```
111 \@ifundefined{theorem@style}{\input{theorem.sty}}{}
112 \ProvidesFile{thp.sty}
113              [\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.

```
114 \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 ??.

```
⟨*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 [?]) 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}}}]%
```

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.

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

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

Références

[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.