Karnaj

Zeste de savoir

Des théorèmes personnalisés en LaTeX

6 juin 2019

Table des matières

Lorsque nous écrivons un document avec LaTeX, il peut arriver que nous ayons besoin de définir des théorèmes, lemmes, définitions, etc. Pour cela, LaTeX dispose de la commande `\newtheorem`. Ainsi, nous pouvons définir un environnement `definition` associé au mot «Définition» avec `\newtheorem{definition}{Définition}`. On peut ensuite utiliser cet environnement pour écrire des définitions.

Cependant, cette méthode n'est pas très personnalisable et tous les environnements définis de cette manière se ressemblent alors que nous préférerions peut-être avoir quelque chose d'un peu plus fou. Heureusement, plusieurs *packages* viennent combler ce manque. Dans ce tutoriel, nous allons voir comment utiliser le *package* **ntheorem** pour obtenir quelque chose de personnalisé.

Prérequis
Connaissance des bases du LaTeX.
Objectifs
Découvrir un moyen simple de composer ses propres théorèmes et autres propriétés en LaTeX.

1. Introduction au package ntheorem

1.1. Utilisation basique de ntheorem

Pour commencer, il nous faut inclure le *package* ntheorem. Nous ajoutons juste la ligne suivante a notre préambule.

```
1 \usepackage{ntheorem}
```

Certains bouts de code et leurs rendus sont disponibles sur ce PDF . Bien sûr, tester d'autres codes et voir ce qu'ils donnent est une bonne idée.

1.1.1. Des environnements prédéfinis

Maintenant, nous pouvons nous lancer dans l'aventure. Pour commencer, il nous faut savoir que le *package* ntheorem a quelques environnements prédéfinis. Pour les utiliser, il nous faut charger ntheorem avec l'option `standard` sans quoi nous obtiendrons des erreurs comme «*LaTeX Error : Environment Proposition undefined*». Lorsque nous utilisons l'option `standard` nous avons accès à plusieurs types d'environnements.

- Des environnements de théorèmes: `Theorem`, `Lemma`, `Proposition`, `Corollary`, `Satz` et `Korollar`.
- Des environnements de preuves: `Proof` et `Beweis`.
- Des environnements de définitions: `Example` et `Beispiel`.
- Des environnements de remarques: `Anmerkung`, `Bemerkung` et `Remark`.

Utilisons `Proposition` par exemple.

```
\documentclass[12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[standard]{ntheorem}

\begin{document}
Voici une proposition.
\begin{Proposition}
   Voici une belle proposition.
\end{Proposition}
\end{document}
```

Ces environnements vont nous permettre de voir les bases de l'utilisation de ntheorem. Ce n'est qu'ensuite que nous allons créer nos propres théorèmes.

1.1.2. Les bases

Le résultat est ultra-basique. On a en gras le nom associé à l'environnement (ici «Proposition») et son numéro. Le numéro est calculé automatiquement par LaTeX grâce aux compteurs. La proposition en elle-même est en italique. Chaque environnement a son compteur et son style d'affichage. Observons par exemple le résultat produit par le code suivant.

```latex
\begin{Proposition}
   Voici notre première proposition.
\end{Proposition}
\begin{Definition}
   Voici une belle définition.
\end{Definition}
\begin{Proposition}
   Notre deuxième proposition.
\end{Proposition}
\begin{Definition}
   Notre deuxième définition.
\end{Definition}
```

On observe bien que chaque environnement a son propre compteur. De plus, alors que le texte de la proposition est en italique, celui de la définition ne l'est pas.

Ces environnements prennent un texte en argument facultatif. Ce texte est affiché entre parenthèses après le numéro du théorème. Nous pouvons ainsi donner un nom à une proposition ou indiquer qu'elle est importante.

```latex
\begin{Proposition}[ZdS]
   La proposition ZdS.
\end{Proposition}
\begin{Proposition}[Important]
   Cette proposition est importante.
\end{Proposition}
```

1.1.3. Le package amsthm

Le *package* **amsthm** fournit également des outils pour faire des théorèmes. Il est en conflit avec le *package* ntheorem. Si nous voulons quand même utiliser certaines fonctionnalités de amsthm, il est possible de les faire cohabiter. Pour cela, il faut inclure ntheorem avec l'option `amsthm` **au lieu de** charger amsthm. Dans ce cas, nous ne pouvons pas utiliser l'option `standard` qui est en conflit avec l'option `amsthm` car elles définissent toutes les deux un environnement `Proof`. Notons qu'il est conseillé d'utiliser ntheorem sans chercher à y mettre des fonctionnalités de amsthm. En effet, ntheorem est plus complet, mais semble également être plus intuitif à utiliser.

1.2. Référence à un théorème

Le *package* ntheorem complète le système de références de LaTeX. Pour avoir accès à ce système, il nous faut le charger avec l'option `thref`. Cette option rajoute à un argument facultatif à la commande `\label` et nous donne la commande `\thref` pour faire références aux étiquettes. On peut alors écrire ce code.

```
\section{Section 1}\label{sec:premier}[Section]
On fait référence à la \thref{sec:premier}.
```

L'argument facultatif de `\label` correspond au texte qui est affiché avant le numéro de l'objet étiqueté. Ainsi, ici, nous avons étiqueté une section avec l'argument facultatif `Section`. Le texte qui sera affiché est alors «On fait référence à la Section 1».

On peut alors utiliser ce système pour étiqueter nos propositions.

```
\begin{Definition}[Zeste de Savoir]\label{def:zds}[Definition]
   Zeste de Savoir est une association dont le but est de
       promouvoir le partage de connaissances
   à travers des ressources pédagogiques gratuites et de préférence
       sous licence libre.
\end{Definition}
Zeste de Savoir a été défini en tant que \thref{def:zds}.
```

Et on a encore mieux: lorsque l'on fait une étiquette pour un environnement défini avec ntheorem, le paramètre facultatif vaut automatiquement le nom associé à l'environnement. Ceci veut dire qu'on est obligé de le mettre seulement dans le cas où l'on ne veut pas ce nom. Ainsi le code qui suit produit le même résultat que celui que nous avons précédemment écrit.

```
\begin{Definition}[Zeste de Savoir]\label{def:zds}
   Zeste de Savoir est une association dont le but est de
       promouvoir le partage de connaissances
   à travers des ressources pédagogiques gratuites et de préférence
       sous licence libre.
\end{Definition}
Zeste de Savoir a été défini en tant que \thref{def:zds}.
```

> Le paramètre facultatif n'est automatique que pour les environnements définis avec ntheorem. Il ne l'est pas avec les sections ou encore avec les équations.

Le code qui suit produira alors l'avertissement «*LaTeX Warning : There were undefined references*».

```
\section{Section 1}\label{sec:premier}
On fait référence à la \thref{sec:premier}.
```

Avec ce code on obtient alors «On fait référence à la 1». `\thref` ne sait pas quel mot afficher et n'en affiche donc pas.

La commande `\ref` reste utile. Déjà dans le cas où nous ne voulons pas avoir de mot d'affiché. Nous pourrions utiliser `\thref` avec un argument facultatif vide, mais c'est quand même plus simple d'utiliser `\ref`. De même, `\ref` est utile si nous voulons obtenir quelque chose comme «Les propositions 1 et 3».

1.2.1. Conflits possibles

Lorsque nous chargeons ntheorem avec l'option `thref`, il peut y avoir des conflits avec plusieurs *packages*.

L'un de ces conflits est avec le *package* **amsmath**. Pour le résoudre, il nous faudra charger ntheorem **après** amsmath et avec l'option `amsmath`. Cela signifie également que les *packages* qui chargent automatiquement amsmath doivent être chargés avant ntheorem. Ainsi, *les packages* **mathtools** ou **empheq** doivent être chargés avant ntheorem.

Il peut aussi interférer avec **babel**. Pour régler ce conflit, il suffira de charger babel avant ntheorem. De même, pour utiliser **hyperref**, il faut charger ntheorem avec l'option `hyperref` et après avoir chargé hyperref.

1.3. Lister les théorèmes

Avec le *package* ntheorem, on peut même faire une liste des théorèmes. Cette action repose sur la commande `\listtheorems`. Elle prend en paramètre le type à lister. Par exemple, pour lister les propositions, nous allons utiliser cette commande.

```
1 \listtheorems{Proposition}
```

Tout comme pour les références, la liste des théorèmes peut nécessiter plusieurs compilations.

On peut alors voir toutes les propositions du document dans leur ordre d'apparition. L'affichage est formaté de la manière suivante. Chaque ligne correspond à une proposition. On a d'abord le numéro de la proposition, puis son nom (s'il y en a un) suivi d'une suite de points. La ligne se termine par la page de la proposition. Notons que les données sont parfaitement alignées.

Nous pouvons choisir d'afficher plusieurs types de théorèmes. Pour cela, il faut passer à `\listtheorems` tous les types de théorèmes dont on veut la liste, séparés par une virgule. Ainsi, nous pouvons lister les propositions et les définitions.

```
1 \listtheorems{Proposition,Definition}
```

On obtient alors la liste des propositions et des définitions toujours dans l'ordre d'apparition. Pour obtenir la liste des propositions et la liste des définitions, il nous faudra utiliser deux fois `\listtheorems`.

```
\listtheorems{Definition}
\listtheorems{Proposition}
```

Notons que nous pouvons ajouter une ligne à la fin de la liste des théorèmes avec la commande `\addtotheoremfile`. Elle prend en paramètre obligatoire le texte à ajouter à la fin des listes et en paramètre facultatif le type de théorème pour lequel on veut ajouter cette ligne. Si on ne lui donne pas de paramètre facultatif, le texte sera ajouté à la fin de toutes les listes.

```
\addtotheoremfile{Cette ligne sera ajoutée à la fin de toutes les
    listes.}
\addtotheoremfile[Proposition]{Cette ligne sera ajoutée à la fin
    des listes de propositions.}
```

Il est possible de faire un théorème qui ne sera pas affiché dans la liste. Pour cela, il nous faut utiliser la version étoilée de l'environnement.

```
\begin{Proposition*}[Liste]
   Cette proposition ne sera pas affiché dans la liste des
       propositions.
\end{Proposition*}
\listtheorems{Proposition}
```

Avec ce code, la proposition «Liste» ne sera pas affichée dans la liste des propositions.

1.3.1. Formater l'affichage des listes

Nous pouvons même formater l'affichage des listes différemment. Pour le moment, l'affichage est formaté de cette manière.

```
<numéro du théorème>   <argument optionnel de l'environnement>
    \dotfill <page du théorème>
```

Nous pouvons changer la manière dont est formaté l'affichage grâce à la commande `\theoremlisttype` qui prend en paramètre le type d'affichage que l'on veut obtenir. Le *package* ntheorem définit quatre types d'affichage.

Celui par défaut (celui que nous avons défini plus haut) correspond à l'argument `all`. On peut également utiliser l'argument `allname` qui a pour caractéristique d'afficher le nom associé à l'environnement avant son numéro.

```
<nom associé à l'environnement> <numéro du théorème>   <argument
    optionnel> \dotfill <page du théorème>
```

Ces deux arguments ont une variante qui affiche uniquement les théorèmes définis avec un argument facultatif. Ces variantes sont `opt` (variante de `all`) et `optname` (variante de `allname`). On peut alors faire un code qui affiche plusieurs listes.

```
\listtheorems{Definition}               % Toutes les définitions,
    option « all »
\theoremlisttype{optname}
\listtheorems{Definition}               % Définitions avec argument
    optionnel, option « optname »
\listtheorems{Proposition}              % Propositions avec
    argument optionnel, option « optname »
\theoremlisttype{allname}
\listtheorems{Definition,Proposition}  % Toutes les Propositions et
    définitions, option « allname »
```

2. Création d'environnements

2.1. Les commandes de création

Maintenant que nous avons vu comment utiliser les théorèmes, il nous reste à voir comment en créer. Pour cela, nous devons utiliser la commande `newtheorem`. Elle a deux arguments obligatoires:

— le nom de l'environnement qu'on veut définir;
— le nom associé à cet environnement.

Pour éviter tout problème, il est conseillé de ne pas utiliser d'accents dans les noms d'environnements. Ainsi, plutôt que de définir `Théorème`, nous définirons `Theoreme`.

Nous pouvons par exemple définir un environnement `Exemple` associé au mot «Exemple».

```
\newtheorem{Exemple}{Exemple}
```

On peut alors l'utiliser comme les autres environnements de `ntheorem`.

```
\begin{Exemple}[Premier exemple]
   Voici notre premier exemple.
\end{Exemple}
```

Puisque nous définissons nos propres environnements, nous n'avons plus besoin de charger ntheorem avec l'option `standard`. Cette option n'est nécessaire que si on veut utiliser les environnements prédéfinis.

Notons que nous ne sommes pas obligés de choisir pour nom de l'environnement le nom que l'on veut afficher. Ainsi, avec le code qui suit, nous obtenons le même résultat qu'avec l'environnement `Définition` que nous avons défini.

```
\newtheorem{ex}{Exemple}
\begin{ex}[Premier exemple]
   Voici notre premier exemple.
\end{ex}
```

Lorsque nous utilisons la commande `\newtheorem{<environnement>}{<nom>}`, deux environnements sont définis, l'environnement `<environnement>` et sa version étoilée `<environnement>*` qui permet, rappelons-le, de créer des théorèmes qui ne seront pas affichés dans les listes. Ainsi, en définissant un théorème, nous définissons également sa version étoilée.

2.1.1. Redéfinir un théorème

Il est possible de redéfinir un théorème avec la commande `\renewtheorem`. Elle a comme argument le nom de l'environnement à redéfinir et le nouveau nom qu'on veut y associer. Ainsi, on peut par exemple redéfinir les environnements prédéfinis pour que les noms soient en français (il ne faudra pas oublier l'option `standard` dans ce cas).

```
\renewcommand{Definition}{Définition}
```

Maintenant, lorsque nous utilisons `Definition`, le texte que nous obtiendrons sera «Définition». Bien sûr, nous pouvons redéfinir n'importe quel environnement de ntheorem, même ceux que nous avons définis. Nous ne sommes pas limités aux environnements prédéfinis.

Lorsque nous redéfinissons un environnement, le compteur qui lui est associé n'est pas remis à zéro.

Pour illustrer ce fait, observons le résultat obtenu avec ce code.

```
\begin{Definition}[Première]
   Première « Definition ».
\end{Definition}
\renewtheorem{Definition}{Def}
```

```
\begin{Definition}[Première Def]
   Première « Def ».
\end{Definition}
```

Ici, «Première Def» est la «Def 2».

Nous pouvons définir (et redéfinir) nos théorèmes n'importe où dans notre document (après avoir inclus ntheorem bien entendu). Cependant, ils sont généralement définis dans le préambule.

2.2. Arguments facultatifs de `\newtheorem`

Imaginons que nous avons défini un environnement `Theoreme`. Nous voulons maintenant un environnement `Corollaire`, mais, considérant que les corollaires sont des théorèmes, nous voulons qu'ils suivent la même numérotation, c'est-à-dire qu'ils aient le même compteur.

```
\begin{Theoreme}

\end{Theoreme}
\begin{Theoreme}

\end{Theoreme}
\begin{Corollaire}

\end{Corollaire}
\begin{Theoreme}

\end{Theoreme}
```

Notre but est d'avoir «Théorème 1», «Théorème 2», «Corollaire 3» et «Théorème 4». Pour obtenir ce résultat, nous pourrions ruser et ne pas définir d'environnements `Corollaire`. À la place, il nous suffirait de redéfinir `Theoreme` à chaque fois que l'on veut changer le mot à afficher. Cependant, cette solution n'est pas très propre, et surtout, ntheorem nous donne un moyen de faire ceci très simplement.

Pour cela, il nous faut utiliser `newtheorem` avec un argument facultatif.

```
\newtheorem{Corollaire}[Theoreme]{Corollaire}
```

Avec ce code, l'environnement `Corollaire` est défini et utilise le même compteur que l'environnement `Theoreme`.

Il faut faire attention à la place de l'argument facultatif. `\newtheorem{}[]{}` et `\newtheorem{}{}[]` ont deux comportements différents.

2.2.1. Formater la numérotation

Puisque nous avons parlé de la syntaxe `\newtheorem{}{}[]`, voyons ce qu'elle fait. En fait, elle permet de changer la manière dont l'environnement est numéroté. L'argument facultatif doit être un environnement qui est lui-même numéroté. En utilisant la commande `\newtheorem{<environnement>}{<nom>}[<argument>]`, la numérotation de l'environnement recommencera à 1 à chaque fois que l'environnement `<argument>` est rencontré. De plus, le numéro de l'environnement sera précédé du numéro de l'environnement `<argument>`. Un exemple sera plus parlant.

```
\newtheorem{Theoreme}{Théorème}[section]

\section{Un}
\begin{Theoreme}

\end{Theoreme}
\begin{Theoreme}

\end{Theoreme}

\section{Deux}
\begin{Theoreme}

\end{Theoreme}
\begin{Theoreme}

\end{Theoreme}

\section{Trois}
\begin{Theoreme}

\end{Theoreme}
\begin{Theoreme}

\end{Theoreme}
```

Compilons ce code et observons le résultat. Nous obtenons «Théorème 1.1», «Théorème 1.2», «Théorème 2.1», etc.

Encore une fois, il ne faut pas confondre `\newtheorem{}[]{}` et `\newtheorem{}{}[]`.

En écrivant `\newtheorem{Theoreme}[section]{Théorème}`, nous définissons un environnement `Theoreme` qui utilise le même compteur que `section`, alors qu'avec `\newtheorem{Theoreme}{Théorème}[section]`, nous définissons un environnement `Theoreme` qui recommencera à chaque nouvelle section et qui sera également numéroté avec le numéro de la section dans laquelle il se trouve.

3. Personnalisation des théorèmes

Après avoir vu comment créer des théorèmes, il ne nous reste plus qu'à voir comment personnaliser leur apparence.

3.1. Création d'environnements personnalisés

La création d'environnements se fait avec la commande `newtheorem`. Pourtant nous n'avons pas mentionné quelque chose: cette création se fait avec le style que **nous** avons choisi. Pour le moment, nous avons le style par défaut. Pour modifier ce style, il faut utiliser quelques commandes avant de faire un nouveau théorème. Les modifications se font avec des commandes de bascule, de sorte que le style est défini jusqu'au prochain changement (à moins qu'elle ne se fasse dans un groupe, auquel cas le changement ne sera effectif que dans le groupe). En fait, la création fonctionne de cette manière.

```
1 % Utilisation de commande pour faire un style qu'on appellera
      « style 1»
2 \newtheorem % Création d'un théorème. Il suit le style 1.
3 \newtheorem % Création d'un autre théorème. Il suit le style 1.
4 % Utilisation de commande pour faire un style qu'on appellera
      « style 2»
5 \newtheorem % Création d'un nouveau théorème. Il suit le style 2.
```

3.1.1. La commande `\theoremstyle`

La commande `\theoremstyle` permet de définir le style général du théorème. Elle prend en paramètre l'un des styles existants. Quelques styles sont déjà définis.

— Le style `plain` est le style par défaut de ntheorem. Il tente de reproduire au mieux le style par défaut des théorèmes de LaTeX. Avec lui, l'en-tête de notre théorème est du type «Définition 4 (Exemple de définition)».
— Le style `empty` permet de supprimer le nom associé à l'environnement et le numéro du théorème. Nous n'aurons donc que l'argument optionnel d'affiché (par exemple «Exemple de définition»).
— Le style `change` intervertit le nom associé à l'environnement et le numéro du théorème (nous aurons alors «4 Définition (Exemple de définition)» à la place de «Définition 4 (Exemple de définition)»).
— Le style `margin` place le numéro du théorème légèrement dans la marge gauche.

— Le style `nonumberplain` supprime le numéro du théorème. Nous obtiendrons alors «Définition (Exemple de définition)» à la place «Définition 4 (Exemple de définition)».

Nous disposons également du style `break`. Avec lui, l'en-tête du théorème est séparé de son corps par une ligne verticale. Les derniers styles prédéfinis combinent le style `break` avec un autre style.

— Le style `changebreak` combine les styles `break` et `change` de sorte que l'en-tête du théorème soit séparé de son corps par une ligne verticale et que les postions du nom associé à l'environnement et du numéro du théorème soient inversées.
— Le style `marginbreak` combine les styles `break` et `margin`. On a donc la ligne verticale de séparation et le numéro du théorème légèrement dans la marge.
— Avec le style `nonumberbreak`, le numéro du théorème est supprimé et une ligne verticale sépare l'en-tête du théorème de son corps.

Le *package* nous donne la possibilité de créer nos propres styles grâce à la commande `\newtheoremstyle`. Cette opération un peu plus avancée ne sera pas vue dans ce tutoriel. Nous pouvons nous reporter à la documentation pour plus d'informations à ce sujet.

3.1.2. Les marques de fin

Il est possible d'obtenir des marques de fin de théorème, c'est-à-dire des symboles pour indiquer la fin d'un théorème. Par exemple, nous pourrions vouloir placer un trèfle «» à la fin de chaque théorème.

Pour cela, il nous faut tout d'abord charger ntheorem avec l'option `thmmarks`. Ensuite, il nous suffit d'utiliser la commande `\theoremsymbol`. Elle prend en paramètre le symbole que l'on veut associer aux théorèmes. Ainsi avec le code suivant, on associe l'étoile « » aux «Définitions».

```
1 \theoremsymbol{$\star$}
2 \newtheorem{Definition}{Définition}
```

Tout comme la commande `\theoremstyle`, la commande `\theoremsymbol` est une commande de bascule et le symbole choisi est donc le symbole courant jusqu'à ce qu'on le change.

Notons cependant qu'il est possible de changer ponctuellement le symbole utilisé dans un environnement en utilisant la commande `\qed` dans l'environnement où la modification doit être effectué. La commande `\qed` indique qu'il ne faut pas utiliser le symbole associé au type de théorème, mais le symbole défini avec la commande `\qedsymbol` (la commande `\qedsymbol` prend en paramètre un symbole. Si on ne veut pas de symbole, c'est la commande `\NoEndMark` qu'il faudra utiliser dans l'environnement. Observons le résultat produit par le code qui suit.

```
\theoremsymbol{$\bullet$}
\newtheorem{Definition}{Définition}

\qedsymbol{$\star$}
\begin{Definition}
   Le symbole de fin est $\bullet$.
\end{Definition}
\begin{Definition}
   \qed
   Le symbole de fin est $\star$.
\end{Definition}
\begin{Definition}
   Le symbole de fin est $\bullet$.
\end{Definition}
\begin{Definition}
   \NoEndMark
   Il n'y a pas de symbole de fin.
\end{Definition}
\qedsymbol{$\alpha$}
\begin{Definition}
   \qed
   Le symbole de fin est $\alpha$.
\end{Definition}
```

Remarquons enfin que n'importe quel caractère ou groupe de caractères peut servir de symbole. Nous pouvons parfaitement utiliser `12 ab $\alpha\beta\$` comme symbole.

3.2. Des théorèmes encadrés

Nous pouvons de plus faire des théorèmes encadrés. Pour utiliser cette fonctionnalité, nous devons inclure le package **framed** avant ntheorem et charger ntheorem avec l'option `framed`. Il nous suffit ensuite d'utiliser la commande `\newframedtheorem` pour créer un théorème encadré.

La commande `\newframedtheorem` prend les mêmes arguments (obligatoires et facultatifs) que la commande `\newtheorem`. De même, les commandes de personnalisation fonctionnent de la même manière. En revanche, les commandes `\theorempreskip` et `\theorempostskip` n'ont pas d'effet et il nous faut utiliser les commandes `\theoremframepreskip` et `\theorem framepostskip` qui prennent en argument une longueur. Notons que cette longueur peut être négative; dans ce cas, la bordure de la boîte se rapprochera du texte.

Nous disposons de plus des commandes `\theoreminframepreskip` et `\theoreminframe postskip` qui permettent de spécifier l'espacement entre le texte du théorème et la boîte (donc elles correspondent aux marges intérieures).

3.2.1. Des boîtes colorées

Nous pouvons donc placer nos théorèmes dans des boîtes. Nous pouvons également les placer dans des boîtes colorés. Pour cela, nous utilisons la commande `\newshadedtheorem` qui est similaire à `\newtheorem`. La couleur de fond est indiquée grâce à la commande `\shadecolor` qui prend en paramètre la couleur voulue. Pour utiliser `\newshadedtheorem`, il nous faudra charger le package **pstricks** (en plus de framed).

Si nous compilons avec le moteur PdfTeX (avec la commande `pdflatex` par exemple), il ne faudra pas charger pstricks, mais nous pouvons charger **color** (ou **xcolor**) et utiliser la commande `\def\theoremframecommand{\colorbox{<couleur>}}` au lieu de `\shadecolor`.

Bien sûr, les théorèmes définis avec `\newshadedtheorem` et `\newframedtheorem` suivent le style choisi avec `\theoremstyle`.

Avec le code qui suit, on définit un nouveau type de théorème dont la couleur de fond est le cyan et dont on change les marges intérieures.

```
\theoreminframepreskip{0.8cm}
\theoreminframepostskip{0.4cm}
\theoremframepreskip{0.4cm}
\theoremframepostskip{0.4cm}
\shadecolor{cyan}
\newshadedtheorem{TheoremeImportant}[Theoreme]{Théorème}
```

Et ça y est, nous pouvons maintenant avoir des boîtes colorées.

En fait, en définissant `\theoremframecommand`, nous choisissons la boîte que nous voulons pour nos théorèmes. Par exemple, on pourrait utiliser `\fcolorbox{<couleur1}{couleur2}` pour avoir une boîte avec un cadre de couleur `couleur1` et un fond de couleur `couleur2`. Nous pouvons même créer une commande qui s'occupe du `\def` pour nous et ensuite l'utiliser dans notre document.

Ici, par exemple, nous faisons un théorème avec une boîte dont le cadre est noir et le fond cyan et ceci grâce à la commande `\theroemcolorbox` que nous définissons dans notre préambule.

```
\newcommand{\theoremcolorbox}[2]{\def\theoremframecommand{\fcolorbox{#1}{#2}}}

\theoremcolorbox{black}{cyan}
\newshadedtheorem{TheoremeImportant}[Theoreme]{Théorème}

\begin{document}
```

```
\begin{TheoremeImportant}[Théorème très important]
    Voici un théorème très important.
\end{TheoremeImportant}
\end{document}
```

Nous sommes maintenant capables de créer nos propres types de théorèmes. Pour voir plus de fonctionnalités de ntheorem, nous pouvons aller voir sa documentation .

Cependant, le monde des théorèmes ne se réduit pas à ntheorem et nous pouvons par exemple nous renseigner sur les *packages* **thmtools** et **tcolorbox** .

Merci à @Holosmos pour avoir validé ce tutoriel et merci à tous ceux qui ont aidé et apporté des conseils et corrections.