

CHAPITRE 12

Personnaliser L^AT_EX


Dans ce chapitre, on va voir comment définir ses propres commandes et environnements. On verra également quelques packages et commandes permettant d'effectuer des tâches répétitives et de manière automatique. Enfin, on terminera en voyant comment définir ses propres environnements flottants et comment écrire dans des fichiers externes.

12.1 Commandes et environnements

On définit une nouvelle commande avec `\newcommand`. En option, on spécifie le nombre de paramètres qu'elle prend. Elle peut en avoir au maximum neuf.

Arachnophobie : Peur des araignées.

Dans la définition de la nouvelle commande, on utilise `#i` avec `i` étant un chiffre entre 1 et 9, pour obtenir la valeur du i^{e} paramètre.

**Code**

```
\newcommand{\definition}[2]{\textbf{#1 : } #2.}
\definition{Arachnophobie}{Peur des araignées}
```

On définit un nouvel environnement avec la commande `\newenvironment`. Cette commande prend trois paramètres qui sont le nom de l'environnement, le code à placer avant et celui à placer après. En option, on peut préciser le nombre de paramètres que l'environnement prend.

————— Le titre —————

Le contenu de la boîte qui est délimitée
par un trait horizontal en haut, mais
également en bas.

—————

On a donc ici défini un environnement `titledbox` qui permet d’avoir une boîte d’une certaine largeur avec un titre. La boîte est délimitée en haut et en bas par un trait horizontal.



Code

```
\newenvironment{titledbox}[2]({\begin{minipage}{#1}%
\hrulefill-\raisebox{-0.4ex}{#2}~\hrulefill\par\smallskip}%
{\par\hrulefill\end{minipage}}

\begin{titledbox}{6cm}{Le titre}
Le contenu de la boîte qui est délimitée par un trait horizontal
en haut, mais également en bas.
\end{titledbox}
```

On désire parfois redéfinir une commande ou un environnement qui existait préalablement. Pour cela, il faut utiliser `\renewcommand` et `\renewenvironment`.

12.2 Tests et répétitions

Voyons une série de packages qui proposent des commandes qui vont permettre de faire un peu de programmation. On va pouvoir faire des tests, des boucles et des traitements plus sophistiqués sur du texte.


12.2.1 Conditions

Le package `ifthen` définit la commande `\ifthenelse` qui permet d’écrire du code L^AT_EX seulement si une certaine condition est vérifiée. La commande prend trois paramètres qui sont respectivement une condition, le code à insérer si la condition est vraie et celui à insérer si elle est fausse.

12 est plus grand que 7
42 est plus petit que 69

Pour définir la condition, on peut utiliser les opérateurs logiques `\AND`, `\OR` et `\NOT`, les parenthèses `\(` et `\)`, les opérateurs de comparaison `<`, `=` et `>` pour comparer des nombres. La commande `\isodd` teste si un nombre est impair. La commande `\isundefined` teste si une commande

n'est pas définie. La commande `\equal` teste si deux chaînes de caractères sont identiques. Pour comparer des longueurs, on ne peut pas utiliser directement `<`, `=` et `>`, mais il faut le faire dans la commande `\lengthtest`.


Code
ifthen

```

\newcommand{\greater}[2]{#1 est plus \ifthenelse{#1>#2}{grand}{
  petit} que #2}

\greater{12}{7} \\
\greater{42}{69}

```

12.2.2 Boucles

On a parfois besoin de répéter plusieurs fois le même code \LaTeX . Pour ce faire, on peut utiliser la commande `\multido` définie dans le package de même nom.


```

blablablabla
1864679sp 2051146sp 2237613sp 2424080sp 2610547sp
4 3 2 1 0
0.0 0.15 0.29999 0.44998 0.59998
0.00 0.15 0.30 0.45 0.60

```

On peut utiliser la commande pour répéter un certain nombre de fois du code \LaTeX . On peut également définir un compteur, le type de ce dernier dépendant de la première lettre du nom utilisé. La déclaration du compteur a toujours la forme `\nom=initial+increment`.

- `d` pour une longueur
- `i` pour un entier
- `r` pour un réel
- `n` pour un nombre avec un nombre fixé de chiffres après la décimale


Code
multido

```

\multido{}{5}{bla} \\
\multido{\d=1cm+1mm}{5}{\d, } \\
\multido{\i=4+-1}{5}{\i, } \\
\multido{\r=0+0.15}{5}{\r, } \\
\multido{\n=0.00+0.15}{5}{\n, }

```

Le nombre d'itérations est accessible par `\multidocount`. On peut arrêter de boucler avec la commande `\multidostop`. Enfin, remarquez que pour faire un décompte à l'envers, il faut écrire `\i=4+-1` car `\i=4-1` ne fonctionnerait pas.


12.2.3 Manipuler des strings

Le package `coolstr` définit trois commandes qui permettent de tester si une chaîne de caractères correspond à un nombre décimal, à un nombre ou à un nombre entier : `\isdecimal`, `\isnumeric` et `\isint`.

Le package définit également la commande `\substr` qui permet d'extraire une sous-chaîne d'une chaîne. Le premier paramètre correspond à la chaîne à traiter, le second à l'indice du caractère de début et le dernier au nombre de caractères à extraire.

Né en 1983.

Ici, on part donc du septième caractère et on en prend quatre. Cela permet donc d'extraire l'année de naissance. Il faut savoir que le résultat de la commande `\substr` est un string et ne pourra donc par exemple pas être utilisé dans un calcul.

 **Code**
coolstr

```
\newcommand{\printinfo}[1]{Né en \substr{#1}{7}{4}.}
\printinfo{06-11-1983}
```

Le package `xstring` propose beaucoup plus de commandes. On va pouvoir faire des tests, des recherches et des remplacements dans une chaîne et, enfin, compter. Nous n'allons pas toutes les détailler ici, mais voici la liste des essentielles :

1. Commandes qui effectuent un test

- `\IfSubStr` teste si une chaîne est une sous-chaîne d'une autre
- `\IfBeginWith` et `\IfEndWith` testent si une chaîne commence ou se termine par une autre
- `\IfInteger` et `\IfDecimal` testent si une chaîne est un entier ou un nombre décimal
- `\IfStrEq` et `\IfStrEqCase` testent si deux chaînes sont égales, en ignorant ou non la casse


2. Commandes qui renvoient une chaîne

- `\StrBefore` et `\StrBehind` renvoient la sous-chaîne se trouvant avant ou après une sous-chaîne d’une autre chaîne
- `\StrBetween` renvoie une sous-chaîne se trouvant entre deux sous-chaînes d’une autre chaîne
- `\StrSubstitute` renvoie une chaîne dont les occurrences d’une sous-chaîne ont été remplacées par une chaîne
- `\StrDel` renvoie une chaîne dont les occurrences d’une sous-chaîne ont été supprimées
- `\StrLeft` et `\StrRight` renvoient une chaîne contenant les i premiers ou derniers caractères d’une chaîne
- `\StrChar` renvoie le i^{e} caractère d’une chaîne
- `\StrMid` renvoie la sous-chaîne allant du i^{e} caractère au j^{e}

3. Commandes qui renvoient un nombre

- `\StrLen` renvoie la longueur d’une chaîne
- `\StrCount` renvoie le nombre de fois qu’une sous-chaîne apparaît dans une autre
- `\StrPosition` renvoie la position d’une sous-chaîne dans une autre

Voyons un petit exemple avec le résultat qui est produit par ce code :

 **Code**
xstring

```

\StrRight{LaTeX}{3} \\
\StrSubstitute{toto est tout}{to}{tu} \\
\StrLen{Anticonstitutionnellement} \\
\IfBeginWith{Maison}{Mais}{OUI}{NON}

```

```

TeX
tutu est tuut
25
OUI

```

12.3 Définir un flottant

Il existe deux types de flottants par défaut en \LaTeX : les figures et les tables. Un des intérêts de ceux-ci est la possibilité d’ajouter une légende avec la commande `\caption`. Un autre est de pouvoir générer une liste

de tous ceux insérés dans le document. On définit un nouveau type de flottant avec la commande `\newfloat` définie dans le package `float`.


Exemple 1 Exemple d'utilisation de la formule de Pythagore.

Soit un triangle rectangle dont les longueurs de deux des côtés adjacents à l'angle droit valent respectivement 3 et 4 centimètres. Grâce au théorème de Pythagore, on peut écrire :

$$h^2 = 3^2 + 4^2 = 9 + 16 = 25$$

La longueur de l'hypothénuse vaut donc $h = \sqrt{25} = 5$ centimètres.

La commande `\floatstyle` permet de définir le style de tous les nouveaux flottants qui seront créés avec `\newfloat`. La commande `\newfloat` prend trois paramètres qui sont le nom du flottant, son positionnement et enfin une extension pour le fichier auxiliaire utilisé pour les lister. Le nom est défini avec `\floatname` et on insère la liste des flottants avec `\listof`.

 **Code**
float

```

\floatstyle{ruled}
\newfloat{example}{!ht}{lex}
\floatname{example}{Exemple}

\begin{example}
  Soit un triangle rectangle dont les longueurs % ...
  \caption{Exemple d'utilisation de la formule de Pythagore.}
\end{example}

\listof{example}{Liste des exemples}

```

Les différents styles sont `plain`, `plaintop`, `boxed` et `ruled`. Enfin, avec le package `float`, le nouveau descripteur de positionnement `H` permet de forcer le flottant à être placé à l'endroit où il a été déclaré. En fait, cela annule le comportement flottant.

On peut également modifier le style d'un flottant déjà existant avec la commande `\restylefloat`. Par exemple, pour faire en sorte que toutes les figures soient entourées d'une boîte, on peut écrire :

```

\floatstyle{boxed}
\restylefloat{figure}


```

12.4 Définir un compteur

On est parfois amené à devoir définir un nouveau compteur. On les utilise pour la numérotation ou l'énumération. Un nouveau compteur se définit avec la commande `\newcounter` qui prend en paramètre un nom.

On initialise la valeur d'un compteur en utilisant la commande `\setcounter` et on modifie sa valeur avec `\addtocounter`. Enfin, on affiche la valeur d'un compteur avec `\the` suivi du nom du compteur.

Il y avait 12 bières,
Marie en apporte quatre de plus, ce qui en fait 16,
mais Pierre est passé et il n'en reste plus que 9.

 **Code**

```
\newcounter{cntBeers}
\setcounter{cntBeers}{12}
Il y avait \thecntBeers{} bières,

\addtocounter{cntBeers}{4}
Marie en apporte quatre de plus, ce qui en fait \thecntBeers,

\addtocounter{cntBeers}{-7}
mais Pierre est passé et il n'en reste plus que \thecntBeers.
```

On peut afficher un compteur avec un autre style en utilisant l'une des commandes suivantes :

- `\arabic` : chiffres arabes
- `\roman` ou `\Roman` : chiffres romains minuscules ou majuscules
- `\alph` ou `\Alph` : lettres romaines minuscules ou majuscules
- `\fnsymbol` : symboles (pour un compteur ne dépassant pas 9)

12.5 Définir une longueur

On a parfois besoin de définir des longueurs, et pour cela, on utilise la commande `\newlength` qui prend un nom en paramètre. Attention cependant que ce dernier doit être précédé d'un backslash, contrairement au nom qu'on donnait à un nouveau compteur.

Une fois la longueur créée, on peut l'initialiser avec `\setlength` et on peut modifier sa valeur avec `\addtolength`. On ne peut pas directement afficher la valeur d'une longueur, mais seulement l'utiliser dans des commandes

comme `\hspace` par exemple. Si on veut l'afficher, il faut utiliser des packages dédiés comme celui décrit à la section 2.4.5 (page 31).

Papa
est là !



Code

```
\newlength{\sep}
\setlength{\sep}{1cm}
\hspace{\sep} Papa

\addtolength{\sep}{5mm}
\hspace{\sep} est là !
```

Il existe également les commandes `\settowidth`, `\settoheight` et `\settodepth` qui permettent de fixer une longueur égale à la largeur, la hauteur ou la profondeur d'un texte spécifié en second paramètre. On peut par exemple obtenir un 0 traversé par un I (⓪) avec le code suivant :



Code

```
\newlength{\oisp}\settowidth{\oisp}{0}
(0\hspace{-0.75\oisp}I\hspace{0.25\oisp})
```