LaTeX avec MikTeX et WinEdt (suite)
les macros

Marie-Claude Mondini

18 novembre 2006

Sommaire

1 Principe

Une macro est un mini-programme construit à partir de commandes directement compréhensibles par le compilateur (en l'occurrence TeX) et/ou d'autres macros définies préalablement. Ainsi, un utilisateur pourra vouloir se construire ses propres macros pour automatiser certaines tâches répétitives.

Prenons un exemple :

$$\ell^0(\mathbb{R}) \stackrel{\text{déf}}{=} \{(u_n)_n, u_n \to 0\}$$

et supposons que le symbole $\stackrel{\text{déf}}{=}$ soit souvent utilisé dans le document. Il y aura tout intérêt à définir une macro qui se charge de recopier la définition de ce symbole au lieu de retaper à chaque fois tous les ordres nécessaires à son obtention.

Un deuxième exemple, montrant qu'on n'est pas limité à du texte : correspond à une macro personnelle baptisée `\alt`

2 Macros "courtes" sans paramètre

Voici deux syntaxes possibles,

(1M) `\newcommand{\nomCommande}{La Macro}` où nomCommande est un nom composé de lettres seulement (ni chiffres, ni symboles)
Par exemple : `\newcommand{\doc}{document}` qu'on utilise ainsi :
`le \doc\ joint..` ce qui donne : le document joint..
autre exemple déjà vu :

```
\newcommand{\defeq}{\stackrel{\mathrm{d\acute ef}}{=}}
```

ou encore :

```
\newcommand{\alt}{\parbox{.8cm}{\unitlength=.2cm
\begin{picture}(3,4)
\put(1,2){\line(1,0){2}}
\put(1,2){\line(1,2){1}}\put(2,4){\line(1,-2){1}}
\put(2,1){\line(0,1){1}}\put(1.5,1){\line(1,0){1}}
\put(2,2.3){\line(0,1){.8}}
\end{picture}}}
```

(2M) `\def\nomCommande{La Macro}` (mêmes consignes pour nomCommande)
Par exemple : `\def\mq{montrer que}` qu'on utilise ainsi :
`\mq\ $x>0$` ce qui donne : montrer que $x > 0$

la première exposée ci-dessus est fortement conseillée par les défenseurs de LaTeX. Elle offre, entre autres avantages, la sécurité de ne pas redéfinir des macros déjà existantes. (J'avoue que j'utilise aussi la deuxième !)

Lorsque l'on cherche simplement à abrèger le nom d'une macro existante (ie créer un alias), on peut

(1M) utiliser `\def` (mais on a vu que c'est dangereux)
(2M) utiliser `\let` avec la syntaxe suivante : `\let\nomcourt=\nommacro` par exemple : `\let\ie=\leqslant` qui donnera : `$f(x)\ie x$` pour $f(x) \leqslant x$
(3M) utiliser `\renewcommand{\nomCommande}{La nouvelle Macro}` où nomCommande est de nom de la macro à modifier
Par exemple, pour renommer la table des matières en sommaire, j'ai déclaré :
`\renewcommand{\contentsname}{Sommaire}`

Remarques : avec let, on garde l'ancienne macro, pas avec renewcommand ; si renewcommand est dans le corps du document, sa définition est limitée à l'environnement où elle est placée ; pour la valider dans tout le document, il faut la placer dans le préambule

3 Macros avec paramètre

Syntaxe : `\newcommand{\nomCommande}[NbParam]{La Macro}` où
\nomCommande est bien sûr le nom qui désigne cette macro
NbParam est le nombre de paramètres (cad d'arguments) utilisés dans la macro (entier inférieur à 9, 0 par défaut)
La Macro est la succession des commandes à effectuer. Dans cette liste, chacun des arguments sera désigné par #n où n est son numéro d'ordre.
Exercice 1 : créer une macro `\pol` à 4 paramètres ayant l'action suivante : `\pol{x}{a}{b}{c}` donne : ax^2+bx+c que l'on soit ou non en mode mathématique (utiliser`{\ensuremath{La Macro}}` au lieu de `{La Macro}`)

Exercice 2 : créer une macro `\reper` à 3 paramètres ayant l'action suivante : `\reper{u}{v}{w}` donne : $(\vec{u},\vec{v},\vec{w})$

Exercice 3 : créer une macro `\numer` à 1 paramètre permettant d'écrire

XX$^{\text{ième}}$ siècle et non XXième siècle, ou 1$^{\text{er}}$ au lieu de 1er

Exercice 4 : créer une macro `\ds` qui donne : un en-tête avec "Devoir surveillé n°xx" centré avec les minuscules en capitales d'imprimerie plus petites que les majuscules, puis à la ligne suivante à gauche la classe et à droite la date du devoir en italique. Cette commande va prendre trois arguments : le numéro de devoir, la classe et la date.

DEVOIR SURVEILLÉ N°5

MP *18/11/06*

L'emploi de macros personnelles ne sert pas qu'à remplacer des séquences longues et répétitives de commandes, il permet également de réaliser des documents faciles à maintenir car mieux structurés. Supposons qu'on réalise un document produisant des fiches d'exercices de .. mathématiques. Chaque exercice va commencer avec un titre qu'on écrira dans un corps plus grand que le reste de la fiche. Il n'y a pas de problème particulier, un source tel que :

```
{\Large Intégration 1}
```

conviendra parfaitement. Après des semaines de travail, l'ouvrage comporte maintenant plus de 100 fiches lorsque, tout compte fait, le titre des exercices serait plus joli s'il était aussi en gras et centré en haut de la fiche, c'est-à-dire que chaque fiche devrait commencer sur une nouvelle page. Une solution très propre et compréhensible serait de définir dès le début du travail une macro `\titrexo` dont la déclaration serait :

```
\newcommand{\titrexo}[1]{{\Large #1}}
```

Au moment où on veut modifier la présentation des titres des exercices, il suffira de modifier la définition de la macro au niveau du préambule et tout sera recomposé selon les nouvelles exigences.

Exercice 5 : construire la nouvelle macro

4 Etude de cas : Exercices avec ou sans corrigé

Une solution est l'usage du module *ifthen* (.. à suivre) Voici une autre solution se prêtant à des exemples très complexes

créer deux fichiers avec les macros `\rep` à 1 paramètre ; sur l'un (nommé sanscor.tex) la macro est vide (= ne renvoie rien), sur l'autre (nommé cor) la macro permet d'obtenir le corrigé de l'exercice, au format voulu, par exemple

Exercice : Montrer que $\sqrt{2}$ est irrationnel

Solution : | sinon $\sqrt{2}$ s'écrit $\frac{p}{q}$ où $p \in \mathbb{Q}$, $q \in \mathbb{N}^*$ et $p \wedge q = 1$, d'où $2q^2 = p^2$
Ainsi 2 divise p^2 donc p qui s'écrit $p = 2r$ avec $r \in \mathbb{Q}$ ...

Enfin, le fichier des exercices dans lequel on fait appel à l'un ou l'autre des fichiers de déclaration selon le résultat souhaité : en préambule, on aura

```
% \input{\chemin sanscor}
 \input{\chemin cor}
```

et on valide l'une ou l'autre des 2 lignes (`\chemin` désigne le chemin d'accès aux fichiers s'ils sont dans un autre dossier)

Exercice 6 : écrire la macro `\rep`

5 environnement

Rappel : Un nouvel environnement sera déclaré par :

`\newenvironment{nomEnvir}[NbParam]{DéfinitionsDeb}{DéfinitionsFin}}` où

nomEnvir est le nom de l'environnement
NbParam est le nombre de paramètres utilisés dans la macro (entier entre 1 et 9, 0 par défaut)
DéfinitionsDeb indique la configuration et les commandes d'entrée d'environnement
DéfinitionsFin indique la configuration et les commandes de sortie d'environnement

Exercice 7 : Créer un environnement `\demo` qui permet d'obtenir la mise en forme suivante :

Démonstration : Ceci est une preuve
Ceci est la suite
Ceci en est la fin. ∎

où la démonstration est de taille "small", le texte ci-dessus étant obtenu par

```
\begin{demo}
Ceci est une preuve

 Ceci est la suite

 Ceci en est la fin.
\end{demo}
```

6 Solution des exercices

Exo 1 :

```
\newcommand{\pol}[4]{\ensuremath{#2 #1^2 + #3 #1 + #4}}
```

Exo 2 :

```
\newcommand{\numer}[1]{\(^{\mbox{\small #1}}\)}
```

Exo 3 :

```
\newcommand{\reper}[3]{\ensuremath{(\vec{#1}, \vec{#2}, \vec{#3})}}
```

Exo 4 :

```
\newcommand{\ds}[3]{
    \begin{center}
        \textsc{Devoir Surveillé n°#1}\\
        #2\hfill\textit{#3}
    \end{center}
\hrule\vspace{\baselineskip}}
```

Exo 5 :

```
\newcommand{\titrexo}[1]{
 \newpage
  \begin{center}
    \Large #1
  \end{center}}
```

Exo 6 :

```
\newcommand{\rep}[1]
{\begin{tabular}{l|p{15.5cm}}\textbf{\underline{Solution :}}& #1\end{tabular}}
```

Exo 7 :

```
\newenvironment{demo}{ \begin{tabular}{lp{10cm}}
\textit{Démonstration :}& \small }{\hfill\rule{3pt}{3pt}
\end{tabular}}
```