

# La personnalisation du $\LaTeX$

## Une courte introduction

Rémi Lambert

Universite de Liège  
Département de Mathématique  
R.Lambert@ulg.ac.be

28 novembre 2007

## 1 Plan

---

# Plan

---

## ① Plan

## ② Déclaration de nouvelles commandes

Prototype d'une commande

Première approche

Conditions et boucles ( $\TeX$  et  $\LaTeX$ )

Redéclaration et surcharge de commandes

Le `\makeatletter`, qu'est-ce que c'est ?

Déclaration de variantes étoilées

`\newcommand` vs `\def`

Trois astuces (qui a dit « un peu subtiles » ?)

Deux arguments optionnels

Pourquoi commenter les fins de lignes ?

L'intérêt du `\relax`

---

# Plan

---

## 1 Plan

## 2 Déclaration de nouvelles commandes

Prototype d'une commande

Première approche

Conditions et boucles (T<sub>E</sub>X et L<sup>A</sup>T<sub>E</sub>X)

Redéclaration et surcharge de commandes

Le `\makeatletter`, qu'est-ce que c'est ?

Déclaration de variantes étoilées

`\newcommand` vs `\def`

Trois astuces (qui a dit « un peu subtiles » ?)

Deux arguments optionnels

Pourquoi commenter les fins de lignes ?

L'intérêt du `\relax`

## 3 Déclaration de nouveaux environnements

---

# Plan

---

- 1 Plan
- 2 Déclaration de nouvelles commandes
  - Prototype d'une commande
  - Première approche
  - Conditions et boucles (T<sub>E</sub>X et L<sup>A</sup>T<sub>E</sub>X)
  - Redéclaration et surcharge de commandes
  - Le `\makeatletter`, qu'est-ce que c'est ?
  - Déclaration de variantes étoilées
  - `\newcommand` vs `\def`
  - Trois astuces (qui a dit « un peu subtiles » ?)
    - Deux arguments optionnels
    - Pourquoi commenter les fins de lignes ?
    - L'intérêt du `\relax`
- 3 Déclaration de nouveaux environnements
- 4 Création d'un fichier de style

---

# Plan

---

- 1 Plan
- 2 Déclaration de nouvelles commandes
  - Prototype d'une commande
  - Première approche
  - Conditions et boucles ( $\TeX$  et  $\LaTeX$ )
  - Redéclaration et surcharge de commandes
  - Le `\makeatletter`, qu'est-ce que c'est ?
  - Déclaration de variantes étoilées
  - `\newcommand` vs `\def`
  - Trois astuces (qui a dit « un peu subtiles » ?)
    - Deux arguments optionnels
    - Pourquoi commenter les fins de lignes ?
    - L'intérêt du `\relax`
- 3 Déclaration de nouveaux environnements
- 4 Création d'un fichier de style
- 5 Création d'une classe de document... basique

---

# Plan

---

## 1 Plan

## 2 Déclaration de nouvelles commandes

Prototype d'une commande

Première approche

Conditions et boucles ( $\TeX$  et  $\LaTeX$ )

Redéclaration et surcharge de commandes

Le `\makeatletter`, qu'est-ce que c'est ?

Déclaration de variantes étoilées

`\newcommand` vs `\def`

Trois astuces (qui a dit « un peu subtiles » ?)

Deux arguments optionnels

Pourquoi commenter les fins de lignes ?

L'intérêt du `\relax`

## 3 Déclaration de nouveaux environnements

## 4 Création d'un fichier de style

## 5 Création d'une classe de document... basique

## 6 Bibliographie

---

# Déclaration de nouvelles commandes

---

---

# Les commandes

---

## Prototype d'une commande

```
\nom[opt]{arg}
```

`\nom` identifiant de la commande (A-Z, a-z);

`opt` paramètre optionnel ;

`arg` argument de la commande.

---

# Les commandes

---

## Prototype d'une commande

```
\nom[opt]{arg}
```

`\nom` identifiant de la commande (A-Z, a-z);

`opt` paramètre optionnel;

`arg` argument de la commande.

```
$ \sqrt{25} $, $ \sqrt[3]{40} $
```

```
 $\sqrt{25}$ ,  $\sqrt[3]{40}$ 
```

---

# Déclaration de nouvelles commandes

---

## Trois fonctions utiles

```
\newcommand{\nom}[nargs] [valopt] {def}
```

`\nom` commande à définir

(A-Z, a-z, pas de surcharge) ;

`nargs` optionnel - nombre d'arguments (1 à 9) ;

`valopt` optionnel - valeur par défaut du *premier*  
argument ;

`def` actions à réaliser.

---

## Déclaration de nouvelles commandes

---

### Trois fonctions utiles

```
\newcommand{\nom}[nbargs] [valopt] {def}
```

`\nom` commande à définir  
(A-Z, a-z, pas de surcharge) ;

`nbargs` optionnel - nombre d'arguments (1 à 9) ;

`valopt` optionnel - valeur par défaut du *premier*  
argument ;

`def` actions à réaliser.

Pour *redéfinir* une commande, utiliser plutôt

```
\renewcommand{\nom}[nbargs] [valopt] {def}
```

---

## Déclaration de nouvelles commandes

---

### Trois fonctions utiles

```
\newcommand{\nom}[nbargs][valopt]{def}
```

`\nom` commande à définir  
(A-Z, a-z, pas de surcharge) ;

`nbargs` optionnel - nombre d'arguments (1 à 9) ;

`valopt` optionnel - valeur par défaut du *premier*  
argument ;

`def` actions à réaliser.

Pour *redéfinir* une commande, utiliser plutôt

```
\renewcommand{\nom}[nbargs][valopt]{def}
```

Pour définir une commande *si elle n'existe pas déjà*,

```
\providecommand{\nom}[nbargs][valopt]{def}
```

---

# Déclaration de nouvelles commandes

---

## Premier exemple

La commande `\TeX` qui produit  $\TeX$  pourrait être définie par :

```
\newcommand{\TeX}{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX}
```

---

# Déclaration de nouvelles commandes

---

## Premier exemple

La commande `\TeX` qui produit T<sub>E</sub>X pourrait être définie par :

```
\newcommand{\TeX}{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX}
```

La commande `\kern` introduit un « *crénage* » (1)

Le « *crénage* » est l'ajustement de l'espace entre les lettres d'une police à chasse variable.

---

# Déclaration de nouvelles commandes

---

## Premier exemple

La commande `\TeX` qui produit T<sub>E</sub>X pourrait être définie par :

```
\newcommand{\TeX}{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX}
```

La commande `\kern` introduit un « *crénage* » (1) tandis que `\lower` déplace vers le bas une boîte (horizontale).

---

# Déclaration de nouvelles commandes

---

## Aliases

---

# Déclaration de nouvelles commandes

---

## Aliases

On veut raccourcir la saisie de la commande

```
\longleftarrow
```

---

# Déclaration de nouvelles commandes

---

## Aliases

On veut raccourcir la saisie de la commande

`\longleftarrow`

```
\def\llra{\longleftarrow}
\let\llra\longleftarrow
\newcommand{\llra}{\longleftarrow}
```

---

# Déclaration de nouvelles commandes

---

## Aliases

On veut raccourcir la saisie de la commande

`\longleftarrow`

```
\def\llra{\longleftarrow}
\let\llra\longleftarrow
\newcommand{\llra}{\longleftarrow}
```

```
$A \llra B$
```

$A \longleftrightarrow B$

## Plan

### Déclaration de nouvelles commandes

Prototype d'une  
commande

#### Première approche

Conditions et boucles

Redéclaration

`\makeatletter`

Variantes étoilées

`\newcommand` vs `\def`

Trois astuces

Deux arguments  
optionnels

Pourquoi commenter les  
fins de lignes ?

L'intérêt du `\relax`

### Déclaration de nouveaux environnements

### Fichier de style

### Classe de document

### Bibliographie

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{a^{\to 2}}
```

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{a^{\to 2}}
```

Voici ma commande  $\symb$  et  $a^{\to 2}$ .

Voici ma commande  $a^{\rightarrow 2}$  et  $a^{-2}$ .

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{a^{\to 2}}
```

Mon symbole `$$\symb$`. Mais `\symb` produit une erreur.

```
! Missing $ inserted.
```

```
<inserted text>
```

\$

1.13 Mon symbole `$$\symb$`. Mais `\symb`

produit

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{ $a^{\to 2}$ }
```

```
Voici mon symbole \symb. \\  
Voici mon symbole  $a^{\to 2}$ .
```

```
Voici mon symbole  $a^{\to 2}$ .  
Voici mon symbole  $a^{\to 2}$ .
```

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{ $a^{\to 2}$ }
```

```
Voici mon symbole \symb. \\  
Voici mon symbole  $a^{\to 2}$ .
```

```
Voici mon symbole  $a^{\to 2}$ .  
Voici mon symbole  $a^{\to 2}$ .
```

```
Une erreur se produit si je tape  $\symb$  !
```

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{ $a^{\to 2}$ }
```

```
Voici mon symbole \symb. \\  
Voici mon symbole  $a^{\to 2}$ .
```

```
Voici mon symbole  $a^{\to 2}$ .  
Voici mon symbole  $a^{\to 2}$ .
```

```
Une erreur se produit si je tape  $\symb$  !
```

```
! Missing $ inserted.  
<inserted text>  
$
```

```
1.12 Une erreur se produit si je tape  $\symb$   
$ !
```

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{\ensuremath{a^{\to 2}}}
```

Mon symbole `\symb`. Et aussi `\symb`.

Mon symbole  $a^{\rightarrow 2}$ . Et aussi  $a^{\rightarrow 2}$ .

---

# Déclaration de nouvelles commandes

---

## Importance du contexte

```
\newcommand{\symb}{\ensuremath{a^{\to 2}}}
```

Mon symbole `\symb`. Et aussi `\symb`.

Mon symbole  $a^{\rightarrow 2}$ . Et aussi  $a^{\rightarrow 2}$ .

Une autre solution plus fine sera présentée ultérieurement...

---

# Déclaration de nouvelles commandes

---

## Un argument

```
\newcommand{\symbA}[1]{\ensuremath{#1^{\to 2}}}
```

Voici le résultat :  $\symbA{\pi}$  ou  $\$ \symbA{6} \$$ .

Voici le résultat :  $\pi^{\rightarrow 2}$  ou  $6^{\rightarrow 2}$ .

---

# Déclaration de nouvelles commandes

---

## Un argument

```
\newcommand{\symbA}[1]{\ensuremath{\#1^{\to 2}}}
```

Voici le résultat : `\symbA{\pi}` ou `$$\symbA{6}$$`.

Voici le résultat :  $\pi^{\rightarrow 2}$  ou  $6^{\rightarrow 2}$ .

Mais attention : `\symbA` pourrait surprendre.

---

# Déclaration de nouvelles commandes

---

## Un argument

```
\newcommand{\symbA}[1]{\ensuremath{\#1^{\to 2}}}
```

Voici le résultat : `\symbA{\pi}` ou `$$\symbA{6}$$`.

Voici le résultat :  $\pi^{\rightarrow 2}$  ou  $6^{\rightarrow 2}$ .

Mais attention : `\symbA` pourrait surprendre.

Mais attention :  $p^{\rightarrow 2}$  pourrait surprendre.

---

# Déclaration de nouvelles commandes

---

## Un argument

```
\newcommand{\symbA}[1]{\ensuremath{\#1^{\to 2}}}
```

Voici le résultat :  $\symbA{\pi}$  ou  $\$ \symbA{6} \$$ .

Voici le résultat :  $\pi^{\rightarrow 2}$  ou  $6^{\rightarrow 2}$ .

Mais attention :  $\symbA$  pourrait surprendre.

Mais attention :  $p^{\rightarrow 2}$  pourrait surprendre.

Examinons maintenant  $\symbA{\}$ .

Examinons maintenant  $\rightarrow 2$ .

## Plan

### Déclaration de nouvelles commandes

Prototype d'une  
commande

#### Première approche

Conditions et boucles

Redéclaration

`\makeatletter`

Variantes étoilées

`\newcommand` vs `\def`

Trois astuces

Deux arguments  
optionnels

Pourquoi commenter les  
fins de lignes ?

L'intérêt du `\relax`

### Déclaration de nouveaux environnements

### Fichier de style

### Classe de document

### Bibliographie

---

# Déclaration de nouvelles commandes

---

## Plusieurs arguments

---

# Déclaration de nouvelles commandes

---

## Plusieurs arguments

```
\newcommand{\np}[2]{Nom : #1, Prénom : #2.}
```

```
\np{Marc}{Oleptique}
```

```
Nom : Marc, Prénom : Oleptique.
```

---

# Déclaration de nouvelles commandes

---

## Un argument optionnel

```
\newcommand{\symbB}[1][a]{\ensuremath{\#1^{\to 2}}}
```

---

# Déclaration de nouvelles commandes

---

## Un argument optionnel

```
\newcommand{\symbB}[1][a]{\ensuremath{\#1^{\to 2}}}
```

On utilise `\symbB`, `\symbB[30]` ou `\symbB[\pi]`.  
Notons l'effet de `\symbB[]...`

On utilise  $a^{\rightarrow 2}$ ,  $30^{\rightarrow 2}$  ou  $\pi^{\rightarrow 2}$ . Notons l'effet de `\rightarrow 2...`

---

# Déclaration de nouvelles commandes

---

## Plusieurs arguments dont un optionnel

```
\newcommand{\symbD}[2][a]{\ensuremath{#1^{\to #2}}}
```

---

## Déclaration de nouvelles commandes

---

### Plusieurs arguments dont un optionnel

```
\newcommand{\symbD}[2][a]{\ensuremath{\#1^{\to \#2}}}
```

Avec l'argument optionnel : `\symbD[\tau]{3}`. `\`

Sans l'argument optionnel : `\symbD{3}`.

Avec l'argument optionnel :  $\tau^{\rightarrow 3}$ .

Sans l'argument optionnel :  $a^{\rightarrow 3}$ .

---

# Déclaration de nouvelles commandes

---

## Conditions

---

# Déclaration de nouvelles commandes

---

## Conditions

Primitive T<sub>E</sub>X pour un test de parité :

```
\ifodd <nombre>  
    <actions-si-le-nombre-est-impair>  
\else  
    <actions-si-le-nombre-est-pair>  
\fi
```

---

## Déclaration de nouvelles commandes

---

### Conditions

Primitive T<sub>E</sub>X pour un test de parité :

```
\ifodd <nombre>  
    <actions-si-le-nombre-est-impair>  
\else  
    <actions-si-le-nombre-est-pair>  
\fi
```

Par exemple :

```
\newcommand{\nbr}[1]{%  
    Nombre : #1 (\ifodd #1 impair\else pair\fi).}
```

```
\nbr{16} \nbr{7}
```

Nombre : 16 (pair). Nombre : 7 (impair).

---

## Déclaration de nouvelles commandes

---

### Conditions

Primitive T<sub>E</sub>X pour une comparaison entre nombres entiers :

```
\ifnum <nombre1> # <nombre2>  
    <actions-si-le-test-est-vrai>  
\else  
    <actions-si-le-test-est-faux>  
\fi
```

lorsque  $\# \in \{<, =, >\}$ .

---

## Déclaration de nouvelles commandes

---

### Conditions

Par exemple :

```
\newcommand{\compare}[2]{%
```

```
Les deux nombres sont #1 et #2. %
```

```
Le plus petit est : \ifnum#1<#2 #1.\else #2.\fi}
```

```
\compare{23}{17} \\
```

```
\compare{2}{4} \\
```

```
\compare{-3}{-5} \\
```

```
\compare{-3}{-1}
```

Les deux nombres sont 23 et 17. Le plus petit est : 17.

Les deux nombres sont 2 et 4. Le plus petit est : 2.

Les deux nombres sont -3 et -5. Le plus petit est : -5.

Les deux nombres sont -3 et -1. Le plus petit est : -3.

---

## Déclaration de nouvelles commandes

---

### Conditions

Primitive T<sub>E</sub>X pour une comparaison de longueurs :

```
\ifdim <longueur1> # <longueur2>  
    <actions-si-le-test-est-vrai>  
\else  
    <actions-si-le-test-est-faux>  
\fi
```

lorsque  $\# \in \{<, =, >\}$ .

Bon à noter : le test tient compte des unités (px, cm, ...) !

---

# Déclaration de nouvelles commandes

---

## Conditions

Par exemple :

```
\newcommand{\compareL}[2]{%
```

Les deux longueurs sont #1 et #2. %

```
La plus petite est : \ifdim#1<#2 #1.\else #2.\fi}
```

```
\compareL{1cm}{2cm} \\
```

```
\compareL{1cm}{12mm} \\
```

```
\compareL{1in}{72pt} \\
```

```
\compareL{1in}{73pt}
```

Les deux longueurs sont 1cm et 2cm. La plus petite est : 1cm.

Les deux longueurs sont 1cm et 12mm. La plus petite est : 1cm.

Les deux longueurs sont 1in et 72pt. La plus petite est : 72pt.

Les deux longueurs sont 1in et 73pt. La plus petite est : 1in.

---

## Déclaration de nouvelles commandes

---

### Conditions

Un `\ifdim` pour la comparaison entre nombres non entiers :

```
\newcommand{\compareR}[2]{%
```

```
Les deux réels sont #1 et #2.
```

```
Le plus petit est : \ifdim#1pt<#2pt #1.\else #2.\fi}
```

```
\compareR{25}{1} \\\
```

```
\compareR{25}{26} \\\
```

```
\compareR{25.3}{25.4} \\\
```

```
\compareR{25.8}{25.43} \\\
```

Les deux réels sont 25 et 1. Le plus petit est : 1.

Les deux réels sont 25 et 26. Le plus petit est : 25.

Les deux réels sont 25.3 et 25.4. Le plus petit est : 25.3.

Les deux réels sont 25.8 et 25.43. Le plus petit est : 25.43.

---

# Déclaration de nouvelles commandes

---

## Conditions

Primitives  $\TeX$  pour tester le mode courant :

```
\ifhmode <si-mode-horizontal>           \else <sinon> \fi  
\ifvmode <si-mode-verticale>           \else <sinon> \fi  
\ifmmode <si-mode-mathématique>       \else <sinon> \fi  
\ifinner <si-mode-encapsulé-(réduit)>   \else <sinon> \fi
```

---

# Déclaration de nouvelles commandes

---

## Conditions

```
\newcommand{\modes}{\bf
  \ifhmode
    \ifinner IH \else H \fi
  \else
    \ifvmode
      \ifinner \hbox{IV} \else \hbox{V} \fi
    \else
      \ifmmode \hbox{M} \else error \fi
    \fi
  \fi}}
```

Une formule :  $\$ \backslash modes \$$ , ensuite normal  $\backslash modes$ ,  
 $\backslash hbox{\text{encapsulé horizontal } \backslash modes}$  et  
 $\backslash vbox{\text{encapsulé vertical } \backslash modes}$ .  $\backslash par \backslash modes$

Une formule :  $M$ , ensuite normal  $H$ , encapsulé horizontal  $IH$  et  
encapsulé vertical  $IV$ .

**V**

---

# Déclaration de nouvelles commandes

---

## Conditions

```
\newcommand{\comm}[1]{%  
\ifmmode #1~\else #1 (texte)\fi}
```

```
\comm{3}      \\  
$\comm{3}$   \\  
\[\comm{3}\]
```

3 (texte)  
 $3^m$

$3^m$

---

# Déclaration de nouvelles commandes

---

## Conditions

La commande `\ensuremath` fonctionne un peu comme

```
\newcommand{\ensuremath}[1]{\ifmmode#1\else{${#1}$}\fi}
```

---

# Déclaration de nouvelles commandes

---

## Conditions

La commande `\{` fonctionne un peu comme

```
\newcommand{\{ }{\ifmmode\lbrace\else\textbraceleft\fi}
```

---

# Déclaration de nouvelles commandes

---

## Conditions L<sup>A</sup>T<sub>E</sub>X - extension 'ifthen'

Commande de test :

```
\ifthenelse{<test>}{<then>}{<else>}
```

---

# Déclaration de nouvelles commandes

---

## Conditions $\LaTeX$ - extension 'ifthen'

Commande de test :

```
\ifthenelse{<test>}{<then>}{<else>}
```

Opérateurs booléens :  $\AND$ ,  $\OR$ ,  $\NOT$ .

---

## Déclaration de nouvelles commandes

---

### Conditions $\LaTeX$ - extension 'ifthen'

Commande de test :

```
\ifthenelse{<test>}{<then>}{<else>}
```

Opérateurs booléens :  $\AND$ ,  $\OR$ ,  $\NOT$ .

Tests entre nombres :

$$nb1 < nb2 \quad nb1 = nb2 \quad nb1 > nb2$$

---

## Déclaration de nouvelles commandes

---

### Conditions $\LaTeX$ - extension 'ifthen'

Commande de test :

```
\ifthenelse{<test>}{<then>}{<else>}
```

Opérateurs booléens :  $\AND$ ,  $\OR$ ,  $\NOT$ .

Tests entre nombres :

$nb1 < nb2$     $nb1 = nb2$     $nb1 > nb2$

Tests entre longueurs :

$\lengthtest\{dim1 < dim2\}$

$\lengthtest\{dim1 = dim2\}$

$\lengthtest\{dim1 > dim2\}$

---

## Déclaration de nouvelles commandes

---

### Boucles avec l'extension 'ifthen'

```
\newcounter{cn}  
\newcommand{\puniton}[2]{  
  \setcounter{cn}{0}  
  \whiledo{\thecn < #2}{\addtocounter{cn}{1} #1\\}  
}
```

```
\puniton{Je ne parlerai plus à mon voisin.}{5}
```

```
Je ne parlerai plus à mon voisin.  
Je ne parlerai plus à mon voisin.  
Je ne parlerai plus à mon voisin.  
Je ne parlerai plus à mon voisin.  
Je ne parlerai plus à mon voisin.
```

---

# Déclaration de nouvelles commandes

---

## Boucles en T<sub>E</sub>X uniquement

```
\newcount\n
\def\puniton#1#2{%
  \n=0
  \loop\ifnum\n<#2
    \advance\n by1 #1\\
  \repeat
}
```

---

# Déclaration de nouvelles commandes

---

## Boucles en T<sub>E</sub>X uniquement

```
\newcount\n
\def\punition#1#2{%
  \n=0
  \loop\ifnum\n<#2
    \advance\n by1 #1\\
  \repeat
}
```

```
\punition{Je ne parlerai plus à mon voisin.}{5}
```

```
Je ne parlerai plus à mon voisin.
Je ne parlerai plus à mon voisin.
Je ne parlerai plus à mon voisin.
Je ne parlerai plus à mon voisin.
Je ne parlerai plus à mon voisin.
```

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\renewcommand{cmd}[args][opt]{def}
```

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\renewcommand{cmd}[args][opt]{def}
```

```
\newcommand{\wawa}[1]{\textbf{GRAS-#1}}
```

```
\renewcommand{\wawa}[2][1]{  
  \ifnum#1=1 \wawa{#2}\else\textit{ITALIQUE-#1}\fi  
}
```

```
\wawa{essai}
```

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\renewcommand{cmd}[args][opt]{def}
```

```
\newcommand{\wawa}[1]{\textbf{GRAS-#1}}
```

```
\renewcommand{\wawa}[2][1]{  
  \ifnum#1=1 \wawa{#2}\else\textit{ITALIQUE-#1}\fi  
}
```

```
\wawa{essai}
```

```
! TeX capacity exceeded, sorry [parameter stack  
size=6000].
```

```
\kernel@ifnextchar #1#2#3->
```

```
\let \reserved@d =#1\def
```

```
\reserved@a {#2}\def \re...
```

```
1.24 \wawa{essai}
```

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\newcommand{\wawa}[1]{\textbf{GRAS-#1}}
```

```
\let\oldwawa=\wawa
```

```
\renewcommand{\wawa}[2][1]{
```

```
  \ifnum#1=1 \oldwawa{#2}\else\textit{ITALIQUE-#2}\fi  
}
```

```
\wawa{essai}\
```

```
\wawa[2]{autre essai}
```

**GRAS-essai**

*ITALIQUE-autre essai*

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\let\oldtextbf=\textbf

\renewcommand{\textbf}[2][\null]{
  \ifx#1\null\oldtextbf{#2}
  \else({\bf #1}-{\it #2})
  \fi
}
```

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\let\oldtextbf=\textbf

\renewcommand{\textbf}[2][\null]{
  \ifx#1\null\oldtextbf{#2}
  \else({\bf #1}-{\it #2})
  \fi
}
```

Comportement normal : `\textbf{et voilà}.\`

Comportement ajouté : `\textbf[ici]{là}.`

Comportement normal : **et voilà** .

Comportement ajouté : (**ici-là**) .

---

# Déclaration de nouvelles commandes

---

## Surcharge de commandes

```
\let\oldtextbf=\textbf

\renewcommand{\textbf}[2][\null]{%
\ifx#1\null\oldtextbf{#2}%
\else({\bf #1}-{\it #2})%
\fi%
}
```

Comportement normal : `\textbf{et voilà}.\`  
Comportement ajouté : `\textbf[ici]{là}.`

Comportement normal : **et voilà**.  
Comportement ajouté : (**ici**-là).

---

## Déclaration de nouvelles commandes

---

`\makeatletter` et `\makeatother`

`\nom[opt]{arg}`

`\nom` identifiant de la commande (A-Z, a-z) ;

`opt` paramètre optionnel ;

`arg` argument de la commande.

---

## Déclaration de nouvelles commandes

---

`\makeatletter` et `\makeatother`

```
\nom[opt]{arg}
```

`\nom` identifiant de la commande (A-Z, a-z) ;

`opt` paramètre optionnel ;

`arg` argument de la commande.

Le caractère @ est protégé. Pourtant il est utile pour

- définir des commandes « invisibles » pour l'utilisateur ;

```
\newcommand{\commande}{...}
```

```
\makeatletter
```

```
\newcommand{\comm@nde}{...}
```

```
\makeatother
```

---

## Déclaration de nouvelles commandes

---

`\makeatletter` et `\makeatother`

`\nom[opt]{arg}`

`\nom` identifiant de la commande (A-Z, a-z) ;  
`opt` paramètre optionnel ;  
`arg` argument de la commande.

Le caractère @ est protégé. Pourtant il est utile pour

- définir des commandes « invisibles » pour l'utilisateur ;
- accéder à des commandes « système » du T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X.

`\@ifstar`

---

# Déclaration de nouvelles commandes

---

## Déclaration de variantes étoilées

```
\newcommand{\etang}[1]{étang #1 de grenouilles}  
\newcommand{\etang*}[1]{étang #1 de poissons}
```

---

## Déclaration de nouvelles commandes

---

### Déclaration de variantes étoilées

```
\newcommand{\etang}[1]{étang #1 de grenouilles}  
\newcommand{\etang*}[1]{étang #1 de poissons}
```

```
! LaTeX Error: Command \etang already defined.  
Or name \end... illegal, see p.192 of the manual.
```

See the LaTeX manual or LaTeX Companion for  
explanation. Type H <return> for immediate help.

...

```
1.16 ...command{\etang*}[1]{étang #1 de poissons}
```

---

# Déclaration de nouvelles commandes

---

## Déclaration de variantes étoilées

```
\newcommand{\Etang}[1]{étang #1 de grenouilles}
\newcommand{\ETANG}[1]{étang #1 de poissons}
\makeatletter
  \newcommand{\etang}{\@ifstar{\Etang}{\ETANG}}
\makeatother
```

```
\etang{plein} \\
\etang*{rempli}
```

```
étang plein de poissons
étang rempli de grenouilles
```

---

# Déclaration de nouvelles commandes

---

## Déclaration de variantes étoilées

```
\makeatletter
  \newcommand{\et@ngs}[1]{étang #1 de grenouilles}
  \newcommand{\et@ng}[1]{étang #1 de poissons}
  \newcommand{\etang}{\@ifstar{\et@ngs}{\et@ng}}
\makeatother
```

```
\etang{plein} \\
\etang*{rempli}
```

étang plein de poissons  
étang rempli de grenouilles

---

# Déclaration de nouvelles commandes

---

## newcommand vs def

---

<code>\newcommand</code>	<code>\def</code>
--------------------------	-------------------

---

---

## Déclaration de nouvelles commandes

---

### `newcommand` vs `def`

	<code>\newcommand</code>	<code>\def</code>
contrôle d'erreur	oui	non

```
\def\par{machin}
```

---

## Déclaration de nouvelles commandes

---

### `newcommand` vs `def`

	<code>\newcommand</code>	<code>\def</code>
contrôle d'erreur	oui	non
argument opt.	facile	plus compliqué

---

## Déclaration de nouvelles commandes

---

### `newcommand` vs `def`

	<code>\newcommand</code>	<code>\def</code>
contrôle d'erreur	oui	non
argument opt.	facile	plus compliqué
parsing	avec <code>{...}</code>	plus étendu

```
\def\test #1-#2-#3{Date : #1/#2/#3}  
\test 3-5-2005
```

```
Date : 3/5/2005
```

## Déclaration de nouvelles commandes

### newcommand vs def

	<code>\newcommand</code>	<code>\def</code>
contrôle d'erreur	oui	non
argument opt.	facile	plus compliqué
parsing	avec {...}	plus étendu

```
\def\test #1-#2-#3{Date : #1/#2/#3}
\test 3-5-2005
```

Date : 3/5/2005

```
\def\scal<#1,#2>{\ensuremath{\left<#1,\,#2\right>}}
k = \scal<\alpha,\beta>, $k=\scal<\alpha,\beta>$.
```

$\langle \alpha, \beta \rangle, k = \langle \alpha, \beta \rangle.$

---

# Déclaration de nouvelles commandes

---

## Deux arguments optionnels

```
\newcommand{\xvec}[1][0]{x_{#1},\,\ldots,\,\xvecint}  
\newcommand{\xvecint}[1][n]{x_{#1}}
```

---

# Déclaration de nouvelles commandes

---

## Deux arguments optionnels

```
\newcommand{\xvec}[1][0]{x_{#1}, \, \, \ldots, \, \, \xvecint}  
\newcommand{\xvecint}[1][n]{x_{#1}}
```

```
$_xvec$ \ \
```

```
$_xvec[1]$ \ \
```

```
$_xvec[1][m]$
```

$x_0, \dots, x_n$

$x_1, \dots, x_n$

$x_1, \dots, x_m$

---

# Déclaration de nouvelles commandes

---

## Commenter ses fins de lignes

```
\let\oldtextbf=\textbf

\renewcommand{\textbf}[2][\null]{%
\ifx#1\null\oldtextbf{#2}%
\else({\bf #1}-{\it #2})%
\fi%
}
```

Comportement normal : `\textbf{et voilà}.\`

Comportement ajouté : `\textbf[ici]{là}`.

Comportement normal : **et voilà**.

Comportement ajouté : **(ici-là)**.

---

# Déclaration de nouvelles commandes

---

## Commenter ses fins de lignes

```
\let\oldtextbf=\textbf
```

```
\renewcommand{\textbf}[2][\null]{  
\ifx#1\null\oldtextbf{#2}  
\else({\bf #1}-{\it #2})  
\fi  
}
```

Comportement normal : `\textbf{et voilà}.\`

Comportement ajouté : `\textbf[ici]{là}.`

Comportement normal : **et voilà** .

Comportement ajouté : (**ici-là**) .

---

# Déclaration de nouvelles commandes

---

## Commenter ses fins de lignes

```
\huge  
\fbox{  
  Du texte.  
}  
\quad  
\fbox{%  
  Du texte.%  
}
```

Du texte.

Du texte.

---

# Déclaration de nouvelles commandes

---

## L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

---

# Déclaration de nouvelles commandes

---

## L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

`\hskip<largeur>` est une commande qui insère un espace horizontal de largeur donnée.

---

# Déclaration de nouvelles commandes

---

## L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

```
\def\petiteespace{\hskip 3pt plus 1pt}
```

---

## Déclaration de nouvelles commandes

---

### L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

```
\def\petiteespace{\hskip 3pt plus 1pt}
```

```
A \petiteespace B
```

```
A B
```

---

## Déclaration de nouvelles commandes

---

### L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

```
\def\petiteespace{\hskip 3pt plus 1pt}
```

```
majuscule \petiteespace minuscule
```

---

## Déclaration de nouvelles commandes

---

### L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

```
\def\petiteespace{\hskip 3pt plus 1pt}
```

```
majuscule \petiteespace minuscule
```

```
! Missing number, treated as zero.
```

```
<to be read again>
```

```
c
```

```
1.17 majuscule \petiteespace minusc
```

```
ule
```

```
! Illegal unit of measure (pt inserted).
```

```
<to be read again>
```

```
c
```

```
1.17 majuscule \petiteespace minusc
```

```
le
```

---

# Déclaration de nouvelles commandes

---

## L'intérêt du `\relax`

`\relax` est une primitive T<sub>E</sub>X qui... ne fait « rien ».

```
\def\petiteespace{\hskip 3pt plus 1pt\relax}
```

```
majuscule \petiteespace minuscule
```

```
majuscule minuscule
```

---

# Déclaration de nouveaux environnements

---

---

# Déclaration de nouveaux environnements

---

Déclaration type d'un nouvel environnement en L<sup>A</sup>T<sub>E</sub>X :

```
\newenvironment{nom}[nbargs]{début}{fin}
```

---

## Déclaration de nouveaux environnements

---

Déclaration type d'un nouvel environnement en L<sup>A</sup>T<sub>E</sub>X :

```
\newenvironment{nom}[nbargs]{début}{fin}
```

`\begin{nom}` → *début*

`\end{nom}` → *fin*

---

## Déclaration de nouveaux environnements

---

Par exemple :

```
\newenvironment{itemizeit}[1]%  
{\begin{itemize}\renewcommand{\labelitemi}{#1}\it}%  
\end{itemize}}
```

```
\begin{itemizeit}{<}  
\item premier élément ;  
\item deuxième élément ;  
\item troisième élément.  
\end{itemizeit}
```

---

## Déclaration de nouveaux environnements

---

Par exemple :

```
\newenvironment{itemizeit}[1]%  
{\begin{itemize}\renewcommand{\labelitemi}{#1}\it}%  
\end{itemize}}
```

```
\begin{itemizeit}{<}  
\item premier élément ;  
\item deuxième élément ;  
\item troisième élément.  
\end{itemizeit}
```

```
< premier élément ;  
< deuxième élément ;  
< troisième élément.
```

---

## Déclaration de nouveaux environnements

---

Autre exemple :

```
\newenvironment{myitemize}[1]%  
{\textbf{#1}\begin{itemize}}%  
{\end{itemize}}
```

```
\begin{myitemize}{Titre de ma liste}  
\item premier élément ;  
\item deuxième élément ;  
\item troisième élément.  
\end{myitemize}
```

**Titre de ma liste**

- premier élément ;
- deuxième élément ;
- troisième élément.

---

## Déclaration de nouveaux environnements

---

Dernier exemple :

```
\newenvironment{myitemize}[1]%  
{\begin{itemize}}%  
{\end{itemize}\par\hfill---~\textbf{#1}~---\hfill}
```

---

## Déclaration de nouveaux environnements

---

Dernier exemple :

```
\newenvironment{myitemize}[1]%  
{\begin{itemize}}%  
{\end{itemize}\par\hfill---~\textbf{#1}~---\hfill}
```

```
\begin{myitemize}{Titre de ma liste}  
\item premier élément ;  
\item deuxième élément ;  
\item troisième élément.  
\end{myitemize}
```

```
! Illegal parameter number in definition of \endmyitemize.  
<to be read again>  
1  
1.22 ...emize}\par\hfill---~\textbf{#1}~---\hfill}
```

---

## Déclaration de nouveaux environnements

---

Dernier exemple :

```
\newenvironment{myitemize}[1]%  
{\def\mytemp{#1}\begin{itemize}}%  
\end{itemize}\par\hfill---~\textbf{\mytemp}~\hfill}
```

```
\begin{myitemize}{Titre de ma liste}  
\item premier élément ;  
\item deuxième élément ;  
\item troisième élément.  
\end{myitemize}
```

- premier élément ;
- deuxième élément ;
- troisième élément.

— Titre de ma liste —

---

# Création d'un fichier de style

---

---

# Déclaration d'un fichier de style

---

## Les débuts

Les fichiers de styles portent l'extension « `.sty` ».

Attention à l'accessibilité de ces fichiers par L<sup>A</sup>T<sub>E</sub>X !

Linux `/usr/share/texmf/tex/generic`

Windows (MikTeX) `C:\Program Files\MiKTeX\tex\generic`

Commande « `texhash` » tapée dans une console.

Nous allons créer un fichier « `monextension.sty` ».

Voici son contenu :

---

## Déclaration d'un fichier de style

---

### Fichier « monextension.sty »

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{monextension}[RL, 17/11/2007, v1.0]
% extensions à charger
\RequirePackage[utf8]{inputenc}
\RequirePackage[T1]{fontenc}
\RequirePackage{amsmath}
\RequirePackage{graphicx}
\RequirePackage[...]{babel}
% commandes personnelles
\makeatletter
\newcommand{\et@ngs}[1]{étang #1 de grenouilles}
\newcommand{\et@ng}[1]{étang #1 de poissons}
\newcommand{\etang}{\@ifstar{\et@ngs}{\et@ng}}
\makeatother
...
\endinput
```

---

# Déclaration d'un fichier de style

---

## Utilisation de « `monextension.sty` »

Le fichier « `monextension.sty` » est accessible au système  $\LaTeX$  et on tape dans notre document `.tex` :

```
\documentclass[a4paper, 11pt]{article}
\usepackage{monextension}
\begin{document}
  ...
  Mon texte. \etang{plein}
  ...
\end{document}
```

---

# Déclaration d'un fichier de style

---

## Gestion d'options

Prototype de déclaration d'options :

```
\DeclareOption{NomOption}{Commandes}  
\ProcessOptions\relax
```

---

# Déclaration d'un fichier de style

---

## Gestion d'options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{monextension}[RL, 28/11/2007, v1.0]
\RequirePackage{geometry}
```

```
\DeclareOption{maths}%
{
  \RequirePackage{amsmaths}
  \RequirePackage{amssymb}
  \RequirePackage{amsthm}
}
\DeclareOption{graphics}%
{
  \RequirePackage{graphicx}
  \RequirePackage{color}
}
\ProcessOptions\relax
...
```

```
\usepackage[maths]{monextension}
```

---

# Déclaration d'un fichier de style

---

## Passer des options à une extension

On utilise la commande

```
\PassOptionsToPackage{ListeOptions}{extension}
```

---

# Déclaration d'un fichier de style

---

## Passer des options à une extension

On utilise la commande

```
\PassOptionsToPackage{ListeOptions}{extension}
```

Par exemple :

```
\RequirePackage{babel}
```

```
\DeclareOption{fr}%  
{\PassOptionsToPackage{frenchb}{babel}}
```

```
\ProcessOptions\relax
```

```
\usepackage[fr]{monextension}
```

---

# Création d'une classe de document... basique

---

---

## Déclaration d'une classe de document... basique

---

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{monarticle}[2007/11/27 Classe d'articles]

\LoadClass[a4paper]{article}
\RequirePackage[utf8]{inputenc}
\RequirePackage[french]{babel}

\DeclareOption{auteur1}{%
\author{Marc \textsc{Oleptique}\thanks{IUFT Grenoble,
Adresse, {\tt molecptique@machin.fr}.}}}

\DeclareOption{auteur2}{%
\author{Marc \textsc{Assin}\thanks{Université de Liège,
Belgique, {\tt massin@ulg.ac.be}.}}}

\DeclareOption*{%
\PassOptionsToClass{\CurrentOption}{article}}

\ProcessOptions\relax
```

---

# Déclaration d'une classe de document... basique

---

```
\documentclass[12pt, auteur2]{monarticle}
\begin{document}
\title{Titre de l'article}
\maketitle

texte...

\end{document}
```

---

## Bibliographie (non exhaustive)

---

- 1 [T<sub>E</sub>X for the impatient \(pdf\)](#),
- 2 [The not so Short Introduction to LaTeX2e \(pdf\)](#),
- 3 The TEX book, Donald E. Knuth (Addison Wesley),
- 4 The LaTeX companion, M. Goossens, F. Mittelbach, et A. Samarin (Addison Wesley, 1994),
- 5 Source L<sup>A</sup>T<sub>E</sub>X : `latex.ltx`,
- 6 ... beaucoup d'autres...

## Plan

### Déclaration de nouvelles commandes

Prototype d'une  
commande

Première approche

Conditions et boucles

Redéclaration

`\makeatletter`

Variantes étoilées

`\newcommand` vs `\def`

Trois astuces

Deux arguments  
optionnels

Pourquoi commenter les  
fins de lignes ?

L'intérêt du `\relax`

### Déclaration de nouveaux environnements

### Fichier de style

### Classe de document

### Bibliographie

Merci !