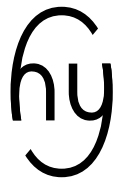
Diverses choses que je sais…
sur la programmation avec LaTeX

version 2

Le TeXnicien de surface

U.S.T.L. & GUTenberg

Dunkerque 2011



Captatio benevolentia

Il s'agit de programmation modérément facile.

Captatio benevolentia

Il s'agit de programmation modérément facile.

Remerciements à Josselin Noirel, Jean-Côme Charpentier, Manuel Pégourié-Gonnard et à beaucoup d'autres intervenant sur `fr.comp.text.tex`.

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2_ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*;`

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;
- `\renewcommand*`;

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;
- `\renewcommand*`;
- `\providecommand`;

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;
- `\renewcommand*`;
- `\providecommand`;
- `\DeclareRobustCommand`;

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2_ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;
- `\renewcommand*`;
- `\providecommand`;
- `\DeclareRobustCommand`;
- `\CheckCommand`;

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;
- `\renewcommand*`;
- `\providecommand`;
- `\DeclareRobustCommand`;
- `\CheckCommand`;
- `\newenvironment`;

Outils classiques, révision

Ce sont les commandes fournies par le noyau LaTeX 2ε.

- `\newcommand` :
  `\newcommand\truc[3][a]{#3, #1 et #2};`
- `\newcommand*`;
- `\renewcommand`;
- `\renewcommand*`;
- `\providecommand`;
- `\DeclareRobustCommand`;
- `\CheckCommand`;
- `\newenvironment`;
- `\renewenvironment`.

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx

Outils LaTeX

Les classiques

Avec xargs

Suffixes

Nom d'une macro

Bibliographie

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx

Avec xargs de Manuel Pégourié-Gonnard

`\newcommandx`, `\newcommandx*`, `\providecommandx`, `\providecommandx*`

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx,

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

```
\newcommandx*\coord[3][1=1, 3=n]{%
  (#2_{#1},\ldots,#2_{#3})}
```

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

```
\newcommandx*\coord[3][1=1, 3=n]{%
  (#2_{#1},\ldots,#2_{#3})}
```

Utilisation :
\coord{x} qui donne $(x_1, \ldots, x_n)$,

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

```
\newcommandx*\coord[3][1=1, 3=n]{%
  (#2_{#1},\ldots,#2_{#3})}
```

Utilisation :
\coord{x} qui donne $(x_1, \ldots, x_n)$,
\coord[3]{x} qui donne $(x_3, \ldots, x_n)$,

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

```
\newcommandx*\coord[3][1=1, 3=n]{%
  (#2_{#1},\ldots,#2_{#3})}
```

Utilisation :
\coord{x} qui donne $(x_1,\ldots,x_n)$,
\coord[3]{x} qui donne $(x_3,\ldots,x_n)$,
\coord{x}[p] qui donne $(x_1,\ldots,x_p)$,

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

```
\newcommandx*\coord[3][1=1, 3=n]{%
  (#2_{#1},\ldots,#2_{#3})}
```

Utilisation :

\coord{x} qui donne $(x_1,\ldots,x_n)$,
\coord[3]{x} qui donne $(x_3,\ldots,x_n)$,
\coord{x}[p] qui donne $(x_1,\ldots,x_p)$,
\coord[0]{x}[p] qui donne $(x_0,\ldots,x_p)$.

Avec xargs de Manuel Pégourié-Gonnard

\newcommandx, \newcommandx*, \providecommandx, \providecommandx*, \renewcommandx, \renewcommandx*, \newenvironmentx, \renewenvironmentx, \CheckCommandx, \DeclareRobustCommandx.

xargs tire parti de xkeyval

```
\newcommandx*\coord[3][1=1, 3=n]{%
  (#2_{#1},\ldots,#2_{#3})}
```

Utilisation :
\coord{x} qui donne $(x_1,\ldots,x_n)$,
\coord[3]{x} qui donne $(x_3,\ldots,x_n)$,
\coord{x}[p] qui donne $(x_1,\ldots,x_p)$,
\coord[0]{x}[p] qui donne $(x_0,\ldots,x_p)$.

☞ Le nombre d'arguments est toujours limité à 9.

Suffixes avec `suffix` de David Kastrup

Pour pouvoir écrire des macros comme `\R*` et `\R+`, on peut utiliser `suffix`.

Suffixes avec `suffix` de David Kastrup

Pour pouvoir écrire des macros comme `\R*` et `\R+`, on peut utiliser `suffix`.

```
\newcommandx\R[2][1=1,2=X]{%
  \EnsembleDeNbr{R}{#1}{#2}{\ReculParent}}

\WithSuffix\newcommand\R*{\R[*]}
```

Suffixes, suite

```
\def\TraiterLigne{\TraiterLigne@x}

\def\TraiterLigne@x#1{%
  \traiterequation{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne@x[#1]#2{%
  \traiterequation{#2}%
  \traiterindication{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne(#1){%
  \AjouterSaut{#1}%
  \TraiterLigne@x}
```

Suffixes, suite

```
\def\TraiterLigne{\TraiterLigne@x}

\def\TraiterLigne@x#1{%
  \traiterequation{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne@x[#1]#2{%
  \traiterequation{#2}%
  \traiterindication{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne(#1){%
  \AjouterSaut{#1}%
  \TraiterLigne@x}
```

On peut alors avoir `\TraiterLigne{...}`

Suffixes, suite

```
\def\TraiterLigne{\TraiterLigne@x}

\def\TraiterLigne@x#1{%
  \traiterequation{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne@x[#1]#2{%
  \traiterequation{#2}%
  \traiterindication{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne(#1){%
  \AjouterSaut{#1}%
  \TraiterLigne@x}
```

On peut alors avoir `\TraiterLigne{...}`, `\TraiterLigne[...]{...}`

Suffixes, suite

```
\def\TraiterLigne{\TraiterLigne@x}

\def\TraiterLigne@x#1{%
  \traiterequation{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne@x[#1]#2{%
  \traiterequation{#2}%
  \traiterindication{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne(#1){%
  \AjouterSaut{#1}%
  \TraiterLigne@x}
```

On peut alors avoir `\TraiterLigne{...}`, `\TraiterLigne[...]{...}`, `\TraiterLigne(...){...}`

Suffixes, suite

```
\def\TraiterLigne{\TraiterLigne@x}

\def\TraiterLigne@x#1{%
  \traiterequation{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne@x[#1]#2{%
  \traiterequation{#2}%
  \traiterindication{#1}%
  \gappto\LeSysteme{\endgraph}%
  \arret}

\WithSuffix\def\TraiterLigne(#1){%
  \AjouterSaut{#1}%
  \TraiterLigne@x}
```

On peut alors avoir `\TraiterLigne{...}`, `\TraiterLigne[...]{...}`, `\TraiterLigne(...){...}` ou encore `\TraiterLigne(...)[...]{...}`.

Du nom des macros

Il y a deux sortes de macros :

Du nom des macros

Il y a deux sortes de macros :

- celles du genre \& dont le nom est composé d'un seul caractère de catégorie autre que « lettres » ;

Du nom des macros

Il y a deux sortes de macros :

- celles du genre `\&` dont le nom est composé d'un seul caractère de catégorie autre que « lettres » ;
- celles du genre `\UneMacro` dont le nom est composé d'un nombre arbitraire de « lettres ».

Du nom des macros

Il y a deux sortes de macros :

- celles du genre `\&` dont le nom est composé d'un seul caractère de catégorie autre que « lettres » ;
- celles du genre `\UneMacro` dont le nom est composé d'un nombre arbitraire de « lettres ».

La catégorie — *catcode* — est un entier compris entre 0 et 15 que TEX attribue à chaque lexème qu'il lit. Les « lettres » ont 11 pour *catcode*. Elles ne contiennent que les caractères alphabétiques de l'ASCII : `a` à `z` et `A` à `Z`.

Du nom des macros

Il y a deux sortes de macros :

- celles du genre `\&` dont le nom est composé d'un seul caractère de catégorie autre que « lettres » ;
- celles du genre `\UneMacro` dont le nom est composé d'un nombre arbitraire de « lettres ».

La catégorie — *catcode* — est un entier compris entre 0 et 15 que TEX attribue à chaque lexème qu'il lit. Les « lettres » ont 11 pour *catcode*. Elles ne contiennent que les caractères alphabétiques de l'ASCII : `a` à `z` et `A` à `Z`.
Une fois attribué, le *catcode* est figé.

Du nom des macros

Il y a deux sortes de macros :

- celles du genre `\&` dont le nom est composé d'un seul caractère de catégorie autre que « lettres » ;
- celles du genre `\UneMacro` dont le nom est composé d'un nombre arbitraire de « lettres ».

La catégorie — *catcode* — est un entier compris entre 0 et 15 que TEX attribue à chaque lexème qu'il lit. Les « lettres » ont 11 pour *catcode*. Elles ne contiennent que les caractères alphabétiques de l'ASCII : `a` à `z` et `A` à `Z`.
Une fois attribué, le *catcode* est figé.

Lors de la lecture du source, TEX consomme les espaces qui suivent une macro du deuxième type et conserve le premier espace qui suit une macro du premier type comme dans `\&␣aa` : « & aa ».

Pur TEX

Utilisation :

```
\csname Une Macro\endcsname
```

Pur TeX

Utilisation :

```
\csname Une Macro\endcsname
```

Définition :

```
\expandafter\def\csname Une Macro\endcsname{%
 ...}
```

Pur TEX

Utilisation :

```
\csname Une Macro\endcsname
```

Définition :

```
\expandafter\def\csname Une Macro\endcsname{%
 ...}
```

La macro \expandafter agit de la manière suivante : le source

\expandafter $L_1\,L_2$,

où L_1 et L_2 sont deux lexèmes, est développé en

$L_1\,\langle$forme développée une fois de $L_2\rangle$.

Pur TeX

Utilisation :

```
\csname Une Macro\endcsname
```

Définition :

```
\expandafter\def\csname Une Macro\endcsname{%
 ...}
```

La macro \expandafter agit de la manière suivante : le source

\expandafter L_1 L_2,

où L_1 et L_2 sont deux lexèmes, est développé en

L_1 ⟨forme développée une fois de L_2⟩.

Après un certain nombre de développement

\expandafter L_1 \expandafter L_2 \expandafter L_3 L_4,

devient

L_1 L_2 L_3 ⟨forme développée une fois de L_4⟩.

Cependant que

\expandafter \expandafter \expandafter L_1 L_2,

devient

L_1 ⟨forme développée une fois du premier lexème du développement de L_2⟩.

Quelques remarques

TEX n'interdit pas de redéfinir ses primitives

Quelques remarques

TEX n'interdit pas de redéfinir ses primitives à vos risques et périls, bien entendu.
Le code `\def\csname Une Macro\endcsname{truc}` redéfinit la macro `\csname` de telle sorte qu'il faut écrire `\csname Une Macro\endcsname` pour obtenir « `truc` ».

Quelques remarques

TEX n'interdit pas de redéfinir ses primitives à vos risques et périls, bien entendu.
Le code `\def\csname Une Macro\endcsname{truc}` redéfinit la macro `\csname` de telle sorte qu'il faut écrire `\csname Une Macro\endcsname` pour obtenir « `truc` ».
Le code `\csname Une Macro\endcsname` semble donc produire ce que l'on veut mais il se pourrait que l'on se précipite vers quelques ennuis ! ; -)

Quelques remarques

TEX n'interdit pas de redéfinir ses primitives à vos risques et périls, bien entendu.
Le code `\def\csname Une Macro\endcsname{truc}` redéfinit la macro `\csname` de telle sorte qu'il faut écrire `\csname Une Macro\endcsname` pour obtenir « `truc` ».
Le code `\csname Une Macro\endcsname` semble donc produire ce que l'on veut mais il se pourrait que l'on se précipite vers quelques ennuis ! ;-)
Même si la macro `\AutreMacro` existe, le source `\csname AutreMacro\endcsname` amènera TEX a signaler une erreur : « l'utilisation de `\csname` n'est pas en accord avec sa définition ».

Exemple d'utilisation avec babel

```
\def\iflanguage#1{%
  \expandafter\ifx\csname l@#1\endcsname\relax
  \@nolanerr{#1}%
  \else
  \bbl@afterfi{\ifnum\csname l@#1\endcsname=\language
    \expandafter\@firstoftwo
    \else
    \expandafter\@secondoftwo
    \fi}%
  \fi}
```

Utilisation :

```
\@nameuse{Une Macro}
```

Programmation avec LaTeX
Le TdS
Outils LaTeX
Nom d'une macro
Pur TeX
LaTeX
etoolbox
Bibliographie

Utilisation :

```
\@nameuse{Une Macro}
```

Bien entendu il faut que `@` soit une lettre.

Programmation avec LaTeX
Le TdS
Outils LaTeX
Nom d'une macro
Pur TeX
LaTeX
etoolbox
Bibliographie

Utilisation :

```
\@nameuse{Une Macro}
```

Bien entendu il faut que `@` soit une lettre.
Une lettre au sens de TeX c.-à-d. de *catcode* 11. C'est le cas dans un fichier de style — c'est le boulot de `\usepackage` de s'en assurer — ou lorsque l'on fait précéder le code de `\makeatletter` et suivre de `\makeatother`.

Programmation avec LaTeX
Le TdS
Outils LaTeX
Nom d'une macro
Pur TeX
LaTeX
etoolbox
Bibliographie

Utilisation :

```
\@nameuse{Une Macro}
```

Bien entendu il faut que `@` soit une lettre.
Une lettre au sens de TeX c.-à-d. de *catcode* 11. C'est le cas dans un fichier de style — c'est le boulot de `\usepackage` de s'en assurer — ou lorsque l'on fait précéder le code de `\makeatletter` et suivre de `\makeatother`.
Définition :

```
\@namedef{Une Macro}{...}
```

Programmation avec LaTeX
Le TdS
Outils LaTeX
Nom d'une macro
Pur TeX
LaTeX
etoolbox
Bibliographie

Utilisation :

```
\@nameuse{Une Macro}
```

Bien entendu il faut que `@` soit une lettre.
Une lettre au sens de TeX c.-à-d. de *catcode* 11. C'est le cas dans un fichier de style — c'est le boulot de `\usepackage` de s'en assurer — ou lorsque l'on fait précéder le code de `\makeatletter` et suivre de `\makeatother`.
Définition :

```
\@namedef{Une Macro}{...}
```

En fait dans les sources de LaTeX, on lit :

```
\def\@namedef#1{\expandafter\def\csname #1\endcsname}
```

Programmation avec LaTeX
Le TdS
Outils LaTeX
Nom d'une macro
Pur TeX
LaTeX
etoolbox
Bibliographie

Exemples d'utilisation 1

1° Dans le noyau LaTeX :

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
  \undefinedpagestyle
  {\@nameuse{ps@#1}}}
```

Exemples d'utilisation 1

1° Dans le noyau LaTeX :

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
  \undefinedpagestyle
  {\@nameuse{ps@#1}}}
```

Lorsque LaTeX rencontre `\pagestyle` il cherche son argument. Si l'on a écrit — comme d'habitude — `\pagestyle{plain}` l'argument est `plain`. Il remplace alors `\pagestyle{plain}` par `\@ifundefined{ps@plain}\undefinedpagestyle{\@nameuse{ps@plain}}`

Exemples d'utilisation 1

1° Dans le noyau LaTeX :

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
  \undefinedpagestyle
  {\@nameuse{ps@#1}}}
```

Lorsque LaTeX rencontre `\pagestyle` il cherche son argument. Si l'on a écrit — comme d'habitude — `\pagestyle{plain}` l'argument est `plain`. Il remplace alors `\pagestyle{plain}` par `\@ifundefined{ps@plain}\undefinedpagestyle{\@nameuse{ps@plain}}` Il développe maintenant `\@ifundefined` &c.

Exemples d'utilisation 1

1° Dans le noyau LaTeX :

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
  \undefinedpagestyle
  {\@nameuse{ps@#1}}}
```

Lorsque LaTeX rencontre `\pagestyle` il cherche son argument. Si l'on a écrit — comme d'habitude — `\pagestyle{plain}` l'argument est `plain`. Il remplace alors `\pagestyle{plain}` par `\@ifundefined{ps@plain}\undefinedpagestyle{\@nameuse{ps@plain}}` Il développe maintenant `\@ifundefined` &c.
Le résultat est que si la macro `\ps@plain` n'est pas définie, le code de départ devient `\undefinedpagestyle` et que si elle est définie, le code se réduit à `\ps@plain`.

Exemples d'utilisation 1

1° Dans le noyau LaTeX :

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
  \undefinedpagestyle
  {\@nameuse{ps@#1}}}
```

Lorsque LaTeX rencontre `\pagestyle` il cherche son argument. Si l'on a écrit — comme d'habitude — `\pagestyle{plain}` l'argument est `plain`. Il remplace alors `\pagestyle{plain}` par `\@ifundefined{ps@plain}\undefinedpagestyle{\@nameuse{ps@plain}}` Il développe maintenant `\@ifundefined` &c.
Le résultat est que si la macro `\ps@plain` n'est pas définie, le code de départ devient `\undefinedpagestyle` et que si elle est définie, le code se réduit à `\ps@plain`.

Et TeX reprend le premier lexème — soit `\undefinedpagestyle` soit `\ps@plain` — et recommence à développer si c'est possible.

Exemples d'utilisation 2

2º Dans `xifthen` :

```
\@namedef{TE@cnttest@@<}{%
  \ifnum\@tempcnta<\@tempcntb AA\else AB\fi}
```

Exemples d'utilisation 2

2º Dans `xifthen` :

```
\@namedef{TE@cnttest@@<}{%
  \ifnum\@tempcnta<\@tempcntb AA\else AB\fi}
```

Ce code permet d'utiliser une macro dont le nom comporte le caractère < qui n'a certainement pas le *catcode* 11.

Avec etoolbox

etoolbox fournit \csdef, \csgdef, \csedef et \csxdef en remplacement de \@namedef.

Avec etoolbox

etoolbox fournit \csdef, \csgdef, \csedef et \csxdef en remplacement de \@namedef.
Il fournit \csuse pour \@nameuse.

```
\newcommandx\A@age@x[3][1=A@!*!*!,3=A@!*!*!]{%
  \ifstrequal{#1}{A@!*!*!}{%
    \csuse{A@age@x@\A@tyenag}#2\A@|}{%
    \csuse{A@ageL@x@\A@tyenag}#1\A@|#2\A@|}%
  \nottoggle{A@printage}%
  {\ifstrequal{#3}{A@!*!*!}%
    {}%
    {\csletcs{#3}{ageinyears}}}%
  {}}
```

Avec etoolbox

etoolbox fournit \csdef, \csgdef, \csedef et \csxdef en remplacement de \@namedef.
Il fournit \csuse pour \@nameuse.

```
\newcommandx\A@age@x[3][1=A@!*!*!,3=A@!*!*!]{%
  \ifstrequal{#1}{A@!*!*!}{%
    \csuse{A@age@x@\A@tyenag}#2\A@|}{%
    \csuse{A@ageL@x@\A@tyenag}#1\A@|#2\A@|}%
  \nottoggle{A@printage}%
  {\ifstrequal{#3}{A@!*!*!}%
    {}%
    {\csletcs{#3}{ageinyears}}}%
  {}}
```

\csletcs, \nottoggle et \ifstrequal sont fournis par etoolbox.

Avec etoolbox

etoolbox fournit \csdef, \csgdef, \csedef et \csxdef en remplacement de \@namedef.
Il fournit \csuse pour \@nameuse.

```
\newcommandx\A@age@x[3][1=A@!*!*!,3=A@!*!*!]{%
  \ifstrequal{#1}{A@!*!*!}{%
    \csuse{A@age@x@\A@tyenag}#2\A@|}{%
    \csuse{A@ageL@x@\A@tyenag}#1\A@|#2\A@|}%
  \nottoggle{A@printage}%
  {\ifstrequal{#3}{A@!*!*!}%
    {}%
    {\csletcs{#3}{ageinyears}}}%
  {}}
```

\csletcs, \nottoggle et \ifstrequal sont fournis par etoolbox.

etoolbox SUITE

```
\newcommand*{\csuse}[1]{%
  \ifcsname#1\endcsname
    \csname#1\expandafter\endcsname
  \fi}
```

etoolbox SUITE

```
\newcommand*{\csuse}[1]{%
  \ifcsname#1\endcsname
    \csname#1\expandafter\endcsname
  \fi}
```

```
\newrobustcmd*{\csdef}[1]{%
  \expandafter\def\csname#1\endcsname}
```

\newrobustcmd est fourni par etoolbox.

etoolbox SUITE ENCORE

Une macro TEX définie avec `\def` n'admet pas qu'un de ses arguments, si elle en accepte, contienne un saut de paragraphe. En LATEX, c'est `\newcommand*` qui a cette propriété. Pour avoir le comportement contraire il faut utiliser `\newcommand` ou, en TEX, `\long\def`.

etoolbox SUITE ENCORE

Une macro TEX définie avec `\def` n'admet pas qu'un de ses arguments, si elle en accepte, contienne un saut de paragraphe. En LATEX, c'est `\newcommand*` qui a cette propriété. Pour avoir le comportement contraire il faut utiliser `\newcommand` ou, en TEX, `\long\def`.
Une macro TEX définie avec `\def` dans un groupe n'est définie que dans ce groupe. Si l'on veut qu'elle soit définie également hors du groupe il faut employer `\global\def` ou `\gdef`.

Programmation avec LATEX
Le TdS
Outils LATEX
Nom d'une macro
Pur TEX
LATEX
etoolbox
Bibliographie

etoolbox SUITE ENCORE

Une macro TEX définie avec `\def` n'admet pas qu'un de ses arguments, si elle en accepte, contienne un saut de paragraphe. En LATEX, c'est `\newcommand*` qui a cette propriété. Pour avoir le comportement contraire il faut utiliser `\newcommand` ou, en TEX, `\long\def`.
Une macro TEX définie avec `\def` dans un groupe n'est définie que dans ce groupe. Si l'on veut qu'elle soit définie également hors du groupe il faut employer `\global\def` ou `\gdef`.
On peut obtenir que la définition de la macro soit développée lors de sa création avec `\edef`. `\xdef` en est la version `\global`.

Programmation avec LATEX
Le TdS
Outils LATEX
Nom d'une macro
Pur TEX
LATEX
etoolbox
Bibliographie

etoolbox SUITE ENCORE

Une macro TEX définie avec `\def` n'admet pas qu'un de ses arguments, si elle en accepte, contienne un saut de paragraphe. En LATEX, c'est `\newcommand*` qui a cette propriété. Pour avoir le comportement contraire il faut utiliser `\newcommand` ou, en TEX, `\long\def`.
Une macro TEX définie avec `\def` dans un groupe n'est définie que dans ce groupe. Si l'on veut qu'elle soit définie également hors du groupe il faut employer `\global\def` ou `\gdef`.
On peut obtenir que la définition de la macro soit développée lors de sa création avec `\edef`. `\xdef` en est la version `\global`.

On pourra rechercher ces commandes dans le fichier `source2e.pdf` pour des exemples d'utilisation.

Bibliographie

- Sur TEX

Bibliographie

- Sur TeX
  - *The TeXbook*, D. E. Knuth ; traduction française *Le TeXlivre* ;

Bibliographie

- Sur TeX
  - *The TeXbook*, D. E. Knuth ; traduction française *Le TeXlivre* ;
  - *TeX by topic*, V. Eijkhout, disponible chez `http://www.lulu.com/content/2555607` (version papier) et sur `https://savannah.nongnu.org/projects/texbytopic` (pdf).

Bibliographie

- Sur TeX
  - *The TeXbook*, D. E. Knuth ; traduction française *Le TeXlivre* ;
  - *TeX by topic*, V. Eijkhout, disponible chez `http://www.lulu.com/content/2555607` (version papier) et sur `https://savannah.nongnu.org/projects/texbytopic` (pdf).
- Sur LaTeX

Bibliographie

- Sur TeX
  - *The TeXbook*, D. E. Knuth ; traduction française *Le TeXlivre* ;
  - *TeX by topic*, V. Eijkhout, disponible chez `http://www.lulu.com/content/2555607` (version papier) et sur `https://savannah.nongnu.org/projects/texbytopic` (pdf).
- Sur LaTeX
  - Les sources commentées du noyau de LaTeX 2_ε sont disponibles sous le nom `source2e.pdf`.