Le sondage est à remplir avant le 15 février à l'adresse suivante :

https://framaforms.org/sondage-sur-les-utilisateurs-de-latex-1672259403

Évidemment, nos canaux de diffusion concentrent principalement des « passionnés » de TEX, et il serait aussi intéressant de sonder d'autres types d'utilisateurs et d'utilisatrices, mais ce sont tout de même des informations intéressantes, toutes partielles qu'elles sont. Plus la population sondée est grande, mieux c'est, donc n'hésitez pas à partager et diffuser ce sondage.

Une fois ces informations récoltées, nous en rendrons compte, en essayant d'analyser tout cela, ainsi que l'évolution par rapport aux sondages précédents.

Maxime Chupin

PASSER À LA DÉFINITION DE COMMANDES DE LATEX3

LATEX3 est un projet ancien puisque c'est en 1999 que Frank Mittlebach et Chris Rowley détaillaient les motivations, les réalisations et le futur du *LATEX3 Project* [9].

Le projet a bien évolué et, depuis un certain temps déjà, l'idée de produire un nouveau format LATEX3 existant en parallèle de LATEX 2ε a été abandonnée : les membres du *LATEX Project* [10] ont décidé d'améliorer et de moderniser LATEX en intégrant graduellement dans son *noyau* les nouveautés, tout en veillant au maintien de la compatibilité descendante pour les anciens documents.

C'est ainsi que désormais, en 2022, nous avons accès aux mécanismes de LATEX3 sans rien charger de plus.

Cet article [11] se limitera à montrer quelques commandes, par le passé fournies par le package xparse mais désormais nativement présentes dans le format LATEX, permettant de définir des commandes [12]. Il faudra se référer à la documentation usrguide pour davantage d'informations.

En réalité, nous nous limiterons à illustrer le passage de la classique commande `\newcommand` à `\NewDocumentCommand`, bien plus puissante. On sait en effet qu'une macro créée avec `\newcommand` a au plus 9 arguments obligatoires, dont au plus un (le premier) optionnel. Toutes les autres possibilités nécessitent, en LATEX 2ε, pas mal de programmation ou des appels à d'autres packages. La commande `\NewDocumentCommand` permet de créer *facilement* des macros avec un mélange de différents types d'arguments.

Enfin, les macros créées avec `\newcommand` ne sont pas *robustes*. Sans rentrer dans les détails, disons que c'est la raison pour laquelle ces macros doivent dans certaines situations être précédées de la primitive `\protect`. En revanche, les macros créées avec `\NewDocumentCommand` sont par défaut robustes, ce qui augmente en général leur fiabilité.

9. Voir https://www.latex-project.org/help/documentation/ltx3info.pdf.
10. https://www.latex-project.org/latex3/
11. Très fortement inspiré de celui, « *From* `\newcommand` *to* `\NewDocumentCommand` », publié en 2010 par Joseph Wright sur son blog : https://www.texdev.net/2010/05/23/from-newcommand-to-newdocumentcommand/.
12. Que nous appellerons ici aussi « macros », même si ces termes ne recouvrent pas tout à fait les mêmes objets (cf. https://tex.stackexchange.com/q/468508/18401).

Description de la commande \NewDocumentCommand

Pour créer des macros « au niveau utilisateur [13] », on dispose donc (entre autres) de :

1. la commande LaTeX2ε \newcommand dont la syntaxe est la suivante :

   ```
   \newcommand{\⟨nom⟩}[⟨narg⟩]{⟨définition de la commande⟩}
   ```

   Nous ne rentrerons pas dans le détail des explications de cette commande très classique ;

2. la commande LaTeX3 \NewDocumentCommand qui offre un nouveau paradigme de programmation :

   ```
   \NewDocumentCommand{\⟨nom⟩}{⟨spéc. arg.⟩}{⟨définition de la commande⟩}
   ```

Dans les deux cas, les arguments successifs de la macro créée sont, dans la ⟨*définition de la commande*⟩, indiqués sous la forme #1, #2, etc.

Contrairement à \newcommand, \NewDocumentCommand ne doit pas seulement connaître le nombre d'arguments de la macro créée, mais aussi la *nature* de chacun. C'est dans son argument obligatoire ⟨*spéc. arg*⟩ de spécification des arguments que ces natures seront spécifiées et, comme il est nécessaire de bien en comprendre le principe, nous allons y consacrer quelques lignes.

La forme générique de ⟨*spéc. arg*⟩ est une liste de lettres où chacune déclare un type d'argument. Nous n'allons ici décrire que les plus communs et nous renvoyons vers la documentation usrguide pour avoir l'ensemble des types d'arguments.

m déclare un argument obligatoire *standard* qui peut-être une simple unité lexicale (*token*) ou un ensemble d'unités lexicales encapsulées entre accolades. C'est le type classique d'un argument TeX normal.

o déclare un argument optionnel à fournir entre crochets dont la valeur sera `-NoValue-` s'il n'est pas fourni lors de l'utilisation. C'est l'équivalent de l'argument optionnel classique de LaTeX.

O{⟨*défaut*⟩**}** déclare un argument optionnel, comme la spécification o, mais affecte la valeur ⟨*défaut*⟩ s'il n'est pas fourni lors de l'utilisation. C'est donc l'argument optionnel avec valeur par défaut.

s permet de déclarer une variante étoilée de la commande. L'argument aura pour valeur \BooleanTrue si la commande est appelée dans sa version étoilée, et \BooleanFalse sinon.

On constate donc que, contrairement au cas de \newcommand, on peut avoir autant d'arguments optionnels que l'on souhaite, et que tous peuvent avoir une valeur par défaut.

13. C'est-à-dire destinées à être utilisées dans les documents de tout le monde, et pas uniquement pour le développement de packages ou de classes.

Par l'exemple

Illustrons par des exemples les quelques éléments introduits jusque-là.

Commande sans argument

Le cas le plus simple est celui d'une macro sans argument, typiquement une substitution de texte. À la sauce LaTeX2ε, on écrit :

Exemple 5

```
\newcommand\insertion{du texte à insérer}
```

Avec LaTeX3, on fait comme expliqué précédemment, et comme la macro que l'on définit n'a pas d'argument, la liste de spécification des arguments est vide. Cela donne :

Exemple 6

```
\NewDocumentCommand\insertion{}{du texte à insérer}
```

Un ou plusieurs arguments obligatoires

Regardons comment traiter en pratique les cas où l'on souhaite avoir un ou plusieurs arguments obligatoires.

En LaTeX2ε, on peut créer les macros suivantes, respectivement à 1 et 2 arguments :

Exemple 7

```
\newcommand\unargument[1]{Mon argument est #1}
\newcommand\deuxarguments[2]{Mes deux arguments sont
   #1 et #2}
```

Pour créer des macros équivalentes avec LaTeX3, on va utiliser l'argument de spécification des arguments pour indiquer combien d'arguments obligatoires on souhaite.

Exemple 8

code

```
\NewDocumentCommand\UnArgument{m}{Mon argument est #1}
\NewDocumentCommand\DeuxArguments{m m}{Mes deux
  arguments sont #1 et #2}
\DeuxArguments{mon premier}{mon second}.
```

résultat

Mes deux arguments sont mon premier et mon second.

Il est d'usage de séparer les spécifications d'arguments par des espaces, mais ce n'est pas obligatoire.

Pour l'instant, les exemples ci-dessous montrent que tout ceci est très similaire au paradigme LaTeX2ε, et c'est tant mieux. Nous allons voir que la puissance de cette nouvelle commande va se révéler lorsque les choses se compliquent.

Un ou plusieurs arguments optionnels

Avec la syntaxe de LaTeX2ε, on peut avoir un seul argument optionnel qui se trouve entre crochets en première position. Pour cela, il faut utiliser le mécanisme de valeur par défaut du premier argument de la commande `\newcommand`.

Exemple 9

code

```
\newcommand\unargopt[1][]{Mon argument est peut-être
   #1.}
\unargopt\par\unargopt[ceci]
```

résultat

Mon argument est peut-être .
Mon argument est peut-être ceci.

Avec LaTeX3, on va utiliser la spécification d'argument o. Ainsi, on écrira :

Exemple 10

```
\NewDocumentCommand\UnArgOpt{o}{Mon argument est
  peut-être #1.}
```

Jusqu'ici, rien d'extraordinaire, mais LaTeX3 fournit des mécanismes de test pour savoir si l'argument optionnel a été donné. Alors qu'avec LaTeX2ε, on peut avoir besoin de packages supplémentaires (comme ifthenelse et ifmtarg), nous disposons ici de la commande `\IfNoValueTF` qui nous permet de faire des choses différentes suivant que l'argument optionnel a été renseigné ou non. Par exemple :

Exemple 11

code

```
\NewDocumentCommand\UnArgOpt{o}{%
  Commande appelée
  \IfNoValueTF{#1}{%true
    \emph{sans} argument.%
  }%
  {%false
    \emph{avec} l'argument \og{}#1\fg{}.%
  }
}
\UnArgOpt\par\UnArgOpt[essai]
```

résultat

Commande appelée *sans* argument.
Commande appelée *avec* l'argument « essai ».

Avec ce nouveau mécanisme, il devient même très simple de créer une macro à deux arguments optionnels, par exemple se plaçant avant et après un argument obligatoire. Pour ce faire, on spécifiera les arguments à l'aide de la suite `o m o`.

Exemple 12

code

```
\NewDocumentCommand\ExempleOMO{o m o}{%
          L'argument \emph{obligatoire}    est \emph
   {#2}.%
  \IfNoValueF{#1}{%
    \par L'argument \emph{optionnel} \no1 est \emph
   {#1}.%
  }%
  \IfNoValueF{#3}{%
    \par L'argument \emph{optionnel} \no2 est \emph
   {#3}.%
  }
}
\ExempleOMO[chat]{chien}[lapin]
```

résultat

L'argument *obligatoire* est *chien*.
L'argument *optionnel* n° 1 est *chat*.
L'argument *optionnel* n° 2 est *lapin*.

Arguments optionnels avec valeur par défaut

Avec `\newcommand`, il est certes possible de déclarer un argument optionnel pourvu d'une valeur par défaut (qui peut être vide) mais on est limité à un unique tel argument. Avec LaTeX3, on dispose de la spécification `O{`⟨*défaut*⟩`}` pouvant être employée plusieurs fois, comme l'illustre l'exemple suivant.

Exemple 13

code

```
\NewDocumentCommand\Animal{O{le chien} O{beau}}{%
  Mon animal préféré est #1 car il est #2.%
}
\Animal\par\Animal[la baleine][gros]
```

résultat

Mon animal préféré est le chien car il est beau.
Mon animal préféré est la baleine car il est gros.

Commandes étoilées

Avec LaTeX en général, une macro peut avoir une version spéciale, portant le même `\`⟨*nom*⟩, mais *étoilée* : `\`⟨*nom*⟩`*`. Créer ces versions spéciales avec `\newcommand` est quelque peu délicat alors qu'avec `\NewDocumentCommand` et la spécification d'argument `s`, cela devient très simple. Nous avons vu que la spécification `s` renvoie un booléen et, à l'instar de `\IfNoValueTF` vu précédemment, on dispose de `\IfBooleanTF`.

Exemple 14

code

```
\NewDocumentCommand\DeuxVersions{s m}{%
  Version%
  \IfBooleanTF{#1}{%true
    étoilée
  }{%false
    classique
  }
   qui utilise #2.%
}
\DeuxVersions*{étoile}\par\DeuxVersions{soleil}
```

résultat

Versionétoilée qui utilise étoile.
Versionclassique qui utilise soleil.

Pour aller plus loin

Dans ce petit article, nous n'avons fait qu'effleurer les possibilités offertes par LaTeX3 et, pour les nombreuses spécifications d'arguments disponibles, nous vous invitons à aller parcourir la documentation offerte par le package usrguide.

Il s'agissait de présenter la commande `\NewDocumentCommand` par quelques exemples de base et de donner à imaginer ses potentialités, notamment concernant les simplifications qu'elle permet, et donc la complexification à moindre frais des commandes ainsi créées.

Notons qu'il existe la commande équivalente `\NewDocumentEnvironment` pour créer de nouveaux environnements, des commandes permettant de redéfinir des macros existantes, de nombreux autres *tests* (`\If ...`), et bien plus encore.

Pour finir cette mise en bouche, un petit exemple illustrant un spécificateur d'argument optionnel permettant de spécifier ce par quoi cet argument doit être encadré.

Exemple 15

code

```
\NewDocumentCommand
   \OptBarre{d<> o m}{%
  Nos arguments sont #3%
  \IfValueT{#1}{, #1}%
  \IfValueT{#2}{, #2}%
  .%
}
\OptBarre{chien}\par
\OptBarre<chat>{chien}
   \par
\OptBarre<chat>[lapin]{
   chien}
```

résultat

Nos arguments sont chien.
Nos arguments sont chien, chat.
Nos arguments sont chien, chat, lapin.