À TRAVERS LE CYBERESPACE DÉCHAÎNÉ : À PROPOS DE LA SOMME DE DEUX NOMBRES

Le 22 septembre 2022, sur la liste `gut@ens.fr`, Arnaud Gazagnes a initié un fil intéressant. Il a posé une question à laquelle on se confronte régulièrement dès que l'on se met à programmer quelques macros : comment produire une macro qui additionne deux nombres ? En réalité, la question était :

> Je voudrais savoir comment je pourrais mettre en place une macro, que j'appelle ici `\annee`, qui me permettrait à la compilation de `\annee{2022}{8}` de voir afficher 2030. (Une macro que je pourrais utiliser dans les exercices de suites numériques, par exemple.)

La question reste vague, et suivant le contexte précis, les solutions peuvent être très différentes. Les réponses apportées sur la liste [51] couvrent un certain nombre d'utilisations et illustrent la richesse du monde de TEX. Je vais donc ici reformuler la problématique suivant les solutions proposées.

Somme de deux entiers

Si l'on ne cherche qu'à sommer deux entiers, alors on peut « simplement » utiliser les commandes de bases de LATEX et passer par des compteurs [52].

Exemple 12

code

```
\newcounter{tempCompteur}%
\newcommand\sommeEntiers[2]{%
  \setcounter{tempCompteur}{#1}%
  \addtocounter{tempCompteur}{#2}%
  \thetempCompteur%
}
\sommeEntiers{2008}{22}\par
\sommeEntiers{4}{5}
```

résultat

2030
9

Bastien Dumont a aussi proposé une solution en utilisant des primitives de TEX, cette fois-ci purement développable (mais toujours dans les entiers).

Exemple 13

code

```
\newcommand\sommeEntiers[2]{\number\numexpr #1+#2}
\sommeEntiers{200}{300}
```

résultat

500

51. Par Denis Bitouzé, François Pétiard, Thierry Bouche et Bastien Dumont.
52. On pourrait faire aussi cela à partir des primitives de TEX, ce qu'a proposé Thierry Bouche.

Somme de deux nombres décimaux

La question qui abordait la notion de *suites* mathématiques pouvait laisser entendre que la macro d'addition devait agir sur les nombres décimaux. François Pétiard a fournit une solution LaTeX en utilisant package xfp [53]. On a alors accès à la commande `\fpeval` qui permet de faire ce qui est demandé.

Exemple 14

code

```
% \usepackage{xfp} % Inutile avec une version récente de LaTeX
\newcommand{\sommeDecimaux}[2]{\fpeval{#1+#2}}
\sommeDecimaux{20.5}{0.666666}
```

résultat

21.166666

Avec le paradigme de programmation LaTeX3

On peut aussi directement utiliser la programmation LaTeX3, dite expl3, pour définir cette commande. Denis Bitouzé a proposé une solution pour la somme de deux entiers. J'ai repris sa solution, et j'en ai fait une version qui peut gérer la somme de deux entiers ou de deux décimaux.

Pour activer ce paradigme de programmation, on doit recourir aux commandes `\ExplSyntaxOn` et `\ExplSyntaxOff` pour en délimiter la région. Elles changent le traitement classique des caractères (les fameux *catcodes*).

Donnons le code complet pour le décortiquer un peu, sans rentrer dans tous les détails car cela nécessiterait un long développement.

Exemple 15

code

```
\ExplSyntaxOn
\cs_new_protected:Nn \__somme_decimaux:nn
{
  \fp_eval:n {#1 + #2}
}
\cs_new_protected:Nn \__somme_entiers:nn
{
  \int_eval:n {#1 + #2}
}
\NewDocumentCommand{\sommeExpl}{ o m m }
{
  \str_if_eq:nnTF{#1}{int}{
     \__somme_entiers:nn {#2} {#3}
  }
  {
    \__somme_decimaux:nn {#2} {#3}
  }
}
\ExplSyntaxOff
```

53. Ce package fournit une interface haut niveau (c'est-à-dire dans le document de production, disons) à l'unité LaTeX3 gérant les nombres à virgule flottante. Ce package a été ajouté au format LaTeX sorti le 1er juin 2022.

code (suite)

```
\sommeExpl{0.66666}{21.7}\par
\sommeExpl[int]{4}{98}
```

résultat

22.36666
102

Il y a plusieurs couches à LaTeX3 : la première, pour la programmation, qui diffère beaucoup de nos habitudes de LaTeX 2$_\varepsilon$, et une autre, orientée cette fois utilisateurs et utilisatrices dont le contenu est le package xparse (inclus désormais dans le format). Pour avoir la documentation centrale de LaTeX3 il suffit d'exécuter dans son terminal la commande suivante :

```
$ texdoc interface3
```

Il y a quelques conventions de programmation de LaTeX3 mais il ne m'est pas possible de rentrer dans ces détails ici [54].

Il y a donc deux commandes internes créées par la commande `\cs_new_protected`. Côté programmation, avec LaTeX3, on n'utilise plus le `@`, mais les symboles `_` et `:` qui permettent de structurer les choses. Les deux commandes créées `\__somme_decimaux` et `\__somme_entiers` [55] prennent donc deux arguments (`nn`), et appellent simplement une autre commande interne au format LaTeX3, soit `\fp_eval` pour la somme de nombre décimaux (pour *floating point*), soit `\int_eval` pour la somme des nombres entiers (pour *integer*). Ces commandes permettent donc d'évaluer les expressions mathématiques qui suivent et de les afficher (pour `\fp_eval`, l'affichage se fait sous forme de nombre décimal).

Ces deux commandes internes (privées) sont alors appelées par une commande plus haut-niveau, côté utilisateur, `\NewDocumentCommand` qui permet de définir des commandes (la documentation est à consulter via le package xparse). On définit ici `\sommeExpl` qui prend un argument optionnel entre crochets, et deux arguments obligatoires (*mandatory*) ce qui s'écrit avec le `{o m m}`.

Ensuite, on teste si l'argument optionnel est égal à la chaîne de caractères `int` grâce à la commande `\str_if_eq` qui compare ses deux premiers arguments et qui exécute le troisième si la comparaison donne *vrai* (`T` pour *true*) ou le quatrième si la comparaison donne *faux* (`F` pour *false*).

Ici, finalement `\fp_eval` fonctionne tout à fait pour les entiers. Le test et l'utilisation de `\int_eval` sont donc inutiles mais cela permet de montrer un peu plus les outils que fournit LaTeX3. Finissons cette section par une modification [56] illustrant la relative facilité avec laquelle on peut créer des commandes étoilées avec LaTeX3.

Exemple 16

code

```
\ExplSyntaxOn
\cs_new_protected:Nn \__somme:nn
{
  \fp_eval:n {#1 + #2}
```

54. Cette question pourra faire l'objet d'un article ultérieur : n'hésitez pas à vous manifester si vous voulez écrire à ce sujet :`secretariat@gutenberg-asso.fr`.
55. Une convention de programmation en LaTeX3 est de commencer les commandes privées par `__`.
56. …suggérée par Denis Bitouzé…

code (suite)

```
}
\NewDocumentCommand{\somme}{ s m m }
{
  \IfBooleanTF{#1}{
     $#2+#3=\__somme:nn {#2} {#3}$
  }
  {
    \__somme:nn {#2} {#3}
  }
}
\ExplSyntaxOff
\somme{0.66666}{21.7}
\somme*{4}{98}
```

résultat

22.36666 4 + 98 = 102

L'argument `s` correspond à l'étoile optionnelle en fin de nom de commande et qui est un booléen *vrai* si l'étoile est présente, et *faux* sinon. Ici, l'ajout de l'étoile permet de composer l'addition qui fournit le résultat.

Conclusion

Les échanges sur la liste `gut@ens.fr` sont souvent passionnants. Je remercie ici toutes les personnes qui prennent le temps de répondre aux problèmes soulevés. Cette entraide est une force de nos logiciels.

Notre équipe au conseil d'administration de l'association a pour projet de migrer cette liste sur notre serveur, de la rendre plus moderne, et de permettre de rendre les archives consultables plus facilement. Vaste projet, mais nous avançons petit à petit. Un autre outil qui me semble également très utile pour l'entraide est le site `https://texnique.fr`. Dans tous les cas, n'hésitez pas à les utiliser pour poser vos questions !

Maxime Chupin