

SUR LES NOUVEAUTÉS DE BABEL

L^AT_EX évolue, Babel aussi... Dans le numéro des *ltnews* de novembre 2024 [1], l'équipe L^AT_EX3 explique les raisons pour lesquelles elle portera désormais le principal de ses efforts sur le moteur LuaT_EX ; les usagers sont fermement invités à abandonner pdfT_EX et X_YT_EX au profit de LuaT_EX.

De son côté, Babel a beaucoup changé depuis sa reprise en mains par Javier Bezos en 2013 : dans les versions récentes de la documentation (`babel.pdf`) Javier oppose le modèle initial, dit aussi « classique » qui reposait sur les fichiers `<langue>.ldf` (`french.ldf` pour le français) au modèle « moderne » qui s'inspire des *locales* d'Unicode. Son intention est clairement de marginaliser à terme le modèle « classique » disponible uniquement pour quelques langues (principalement européennes). Ceci dit, rien n'empêche de mixer les deux approches, « classique » pour le français et « moderne » pour l'arabe, le devanagari, le tibétain, etc. C'est la solution préconisée par Javier Bezos : voir <https://latex3.github.io/babel/guides/locale-french.html>

Nous allons nous intéresser plus particulièrement au français et comparer les deux approches pour notre langue. Nous nous limiterons au moteur LuaT_EX pour les raisons évoquées ci-dessus.

Présentation de l'approche « moderne »

Les *locales* sont des procédures qui permettent de produire, à partir d'un même source (programme informatique ou document), des sorties en différentes langues. Les *locales* sont souvent désignées par des initiales : `fr` ou `fr_FR` pour le français de France, `fr_CA` pour le français canadien, `en_GB` pour l'anglais britannique, etc.

L'adaptation d'un document L^AT_EX à une langue, prenons le français, consiste à :

- activer les bonnes coupures de mots en fin de ligne ;
- traduire les intitulés L^AT_EX : en français `\chapter*{Introduction}` affichera « Chapitre Introduction », `\tableofcontents` affichera « Table des matières » ;
- afficher les dates en français (par exemple commande `\today`) ;
- faciliter la saisie en ajoutant automatiquement les espaces insécables de chasse adaptée devant la ponctuation haute et les guillemets (ce qui est spécifique au français).

Depuis la version 24.14 de Babel, ces quatre points sont désormais pris en compte correctement par l'approche « moderne ». Voici un exemple à compiler avec LuaL^AT_EX :

Exemple 25 – syntaxe de base

```
\documentclass[french]{article}
\usepackage[provide=]{babel}
\babelprovide[
  transforms = rm:sf:punctuation.space
]{french}
\begin{document}
\thispagestyle{empty} \today, \frenchdate{2001}{2}{1}
```

```
\textbf{\chaptername, \contentsname.}
```

Un exemple: ça va? bien! et la «suite»; (!) ?? !? ?! !!

```
\texttt{a: x!} % Aucune espace ajoutée.
\end{document}
```

L'option `provide=*` de `babel` empêche le chargement de `french.ldf` et donne accès à toutes les *locales* de `babel` (on pourra par exemple placer une commande `\selectlanguage{spanish}` dans le corps du document sans autre ajout dans le préambule). La commande `\babelprovide` active la gestion automatique de la ponctuation haute en français pour les polices à empattement (rm) et sans serif (sf), mais pas pour les polices à chasse fixe (tt).

Les réglages des espaces ajoutées sont ceux de `babel-french`. Il est possible de les ajuster; on pourra par exemple simuler l'option `ThinColonSpace` de `babel-french` grâce aux commandes :

Exemple 26 – paramétrer les espaces

```
% espace fine
\SetTransformValue{french}{colon.natural}{.5}
% ...sans élasticité
\SetTransformValue{french}{colon.minus}{0}
\SetTransformValue{french}{colon.plus}{0}
```

Voir <https://latex3.github.io/babel/news/whats-new-in-babel-24.13.html> pour d'autres exemples.

Le seul point où `babel-french` fait mieux que l'approche « moderne » est la gestion du deux-points dans les URL et les ratios avec `LuaLaTeX` : il faudra coder par exemple

Exemple 27 – parade pour les espaces parasites

```
\foreignlanguage{english}{https://mon.site.fr,
C:\textbackslash Mes~Dossiers, 4:3, 16:9}
```

pour ne pas avoir d'espace parasite devant les deux-points⁴⁶; le passage en anglais est inutile avec `babel-french` sous `LuaLaTeX`.

Conclusion

Pour des raisons historiques, `babel-french` opère de nombreux changements sur la maquette du document : présentation des listes, des notes de bas de pages etc. Ces changements étaient justifiés dans les années 1990, lors desquelles les seules classes disponibles `article`, `report` et `book` proposaient des mises en page inadaptées⁴⁷ (espaces verticaux démesurés dans les listes par exemple). Depuis, de nouvelles classes (`koma-script`, `memoir`) et de nouvelles extensions (`enumitem`, `footmisc`, `fnpct`, etc.) permettent d'obtenir des mises en pages élégantes grâce à la souplesse de leur paramétrage. Désormais, c'est à elles qu'il convient de déléguer

46. Oublier la commande `\NoAutoSpacing` propre à `babel-french` !

47. Pas seulement au français !

la totalité de la mise en page en passant l'option `StandardLayout` à `babel-french` pour cantonner celui-ci à la francisation.

L'approche « moderne » de Babel offre, au moins avec `LuaLTeX`, une alternative intéressante à l'utilisation de `babel-french`. Elle se limite strictement à la francisation du document, ce qui devrait accélérer la compilation des documents en français.

Daniel Flipo



🌀 OPT_EX : UN LIBRE HÉRITIER DE PLAIN T_EX

De nos jours, les utilisateurs de T_EX s'appuient, dans leur grande majorité, sur les formats L^AT_EX ou, plus rarement, ConT_EXt, qui leur donnent accès à de vastes bibliothèques de code destinées à leur rendre l'accès au logiciel plus facile et à leur fournir l'implémentation de nombreuses fonctionnalités qui ne figurent pas dans le format originel Plain T_EX de Donald E. Knuth. L^AT_EX et ConT_EXt fournissent tous deux des interfaces pour définir des macros, des éléments de composition comme les notes ou les références croisées et des extensions⁴⁸ qui couvrent des besoins très spécifiques. L'objectif⁴⁹, que la surcouche logicielle `expl3` vient parachever dans L^AT_EX, est que l'utilisateur n'ait pas à manipuler directement les primitives de T_EX.

D'autres projets, comme Eplain⁵⁰ ou OPmac⁵¹, ont cependant maintenu l'approche de Plain T_EX : ils définissent seulement quelques interfaces limitées à T_EX (comme la possibilité d'identifier des boîtes, des registres ou des dimensions par des noms plutôt que par des numéros) et fournissent des macros correspondant aux éléments de composition les plus fréquemment utilisés, comme les titres de sectionnement ; si le rendu ne convient pas à l'utilisateur, celui-ci peut copier les définitions des macros concernées et les modifier à sa guise. En 2020, Petr Olšák, créateur d'OPmac, a publié OpT_EX⁵², qu'il a déclaré stable en février 2021 et qui est inclus dans T_EXLive. Contrairement à Eplain et à OPmac, il ne s'agit pas d'une simple collection de macros, mais d'un nouveau format dont l'objectif est d'actualiser Plain T_EX tout en limitant l'ajout de couches d'abstraction par-dessus le langage T_EX : en somme, OpT_EX peut être caractérisé comme un Plain T_EX augmenté, modernisé, amélioré, qui vient en outre avec un petit écosystème d'extensions et de fichiers de définition de fontes. L'auteur a publié plusieurs articles dans le *TUGboat* pour promouvoir ce travail et défendre son approche de T_EX, particulièrement face à L^AT_EX [1, 2, 3]. Pour ce numéro de la *Lettre*, je tâcherai de montrer quel est l'intérêt de cette approche et comment OpT_EX la met en œuvre tout en offrant à l'utilisateur des facilités supplémentaires par rapport aux autres solutions basées sur Plain T_EX. Je comparerai OpT_EX surtout avec L^AT_EX.

Principales différences avec L^AT_EX

Une bonne partie des différences principales avec L^AT_EX (mais aussi avec ConT_EXt) vient de ce que OpT_EX reste dans l'esprit de Plain T_EX. Nous allons voir cependant

48. *Packages*.

49. Souvent même l'injonction...

50. <https://tug.org/eplain/>

51. <http://petr.olsak.net/opmac-e.html>

52. Site officiel : <http://petr.olsak.net/optex/>