

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 `LuaLaTeX`, 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/>

que certaines particularités de OpTeX, y compris le fait qu'il est basé exclusivement sur LuaTeX, accentuent encore ces différences.

Comparaison de deux documents

Voici un bref document LaTeX suivi par son équivalent OpTeX :

Exemple 28 – LaTeX

```

1  \documentclass[french]{article}
2  \usepackage{babel}
3  \frenchsetup{og=«, fg=»}
4  \babelfont{rm}{TeX Gyre Pagella}
5
6  \begin{document}
7
8  Ce document a été réalisé avec \LaTeX{} et soin.
9
10 \section{Titre de ma première section}
11
12 Voici la section «première section». On y trouvera:
13
14 {
15 \renewcommand{\labelenumi}{\alph{enumi})}
16 \begin{enumerate}
17   \item Une première sous-section;
18   \item Et une seconde contenant une note.
19 \end{enumerate}
20 }
21
22 \subsection{Première sous-section}
23
24 Quelques explications \textbf{très importantes}.
25
26 \subsection{Seconde sous-section}
27
28 D'autres explications\footnote{Avec une note!}.
29
30 \end{document}
```

Exemple 29 – OpTeX

```
\fontfam[Pagella]
\frlang
```

Ce document a été réalisé avec \OpTeX/ et soin.

```
\sec Titre de ma première section
```

Voici la section «~première section~». On y trouvera~:

```
\begitems \style a
```

```

* Une première sous-section\, ;
* Et une seconde contenant une note.
\enditems

\secc Première sous-section

Quelques explications {\bf très importantes}.

\secc Seconde sous-section

D'autres explications\fnote{Avec une note\, !}.

\bye

```

On remarque immédiatement que le balisage d' OpTeX est plus léger. Cela tient tout d'abord à l'absence de marquages spécifiques à \LaTeX : plutôt qu'un environnement document, on utilise la macro `\bye` de Plain \TeX ; les environnements sont délimités par des macros nommées `\beg<...>` et `\end<...>`; pour les changements de fonte ou de style de caractères, plutôt que des macros `\text<...>` à la \LaTeX , on utilise les commutateurs à la Plain \TeX , plus brefs. Plusieurs fonctionnalités spécifiques à OpTeX , dont nous voyons quelques exemples ici, allègent le balisage plus encore que dans Plain \TeX . Certaines macros, notamment les macros de sectionnement, sont définies pour prendre comme argument tout le reste de la ligne, éliminant le besoin d'utiliser des accolades⁵³. Nombre de macros sont conçues pour prendre des arguments brefs : par exemple, un seul caractère pour `\style`, ce qui dispense là encore de l'encadrer par des accolades. De manière générale, les noms de macros sont plus concis que dans \LaTeX .

On remarque également que pas une seule extension n'est chargée dans notre document OpTeX , ce qui est tout à fait fréquent. Le chargement de la fonte et la sélection de la langue sont prises en charge par des macros intégrées au format ; il en aurait été de même, par exemple, si nous avions voulu modifier les dimensions de la page et de la zone de texte. Comme le format OpTeX est beaucoup plus léger que \LaTeX et charge peu d'extensions, voire aucune, la durée de la compilation en est réduite d'autant : je vous laisse le soin de faire des essais⁵⁴. Bien sûr, le prix à payer est que l'on doit gérer soi-même certains aspects qui, dans un document \LaTeX , peuvent être pris en charge par des extensions dédiées : ici, on voit que l'on est obligé d'insérer soi-même les espaces insécables avant les signes de ponctuation, ce qui, avec \LaTeX , est du ressort de `babel-french`.

Ce dernier point nous amène à la différence d'approche entre OpTeX et \LaTeX dans la personnalisation des documents. Comme nous le voyons à la ligne 15 dans l'exemple 28, il est possible de redéfinir certaines macros de \LaTeX pour modifier la mise en forme par défaut ; cependant, au-delà de quelques réglages assez simples, les utilisateurs recourent fréquemment à des extensions qui permettent de personnaliser plus en profondeur les listes, les titres de sectionnement, l'empagagement, etc., sans toucher directement aux macros internes de \LaTeX . Dans OpTeX , c'est l'inverse : un nombre relativement limité de paramètres peuvent être réglés en

53. En cas de besoin, on peut faire précéder la macro de `\bracedparam` et mettre l'argument entre accolades.

54. Bien sûr, pour comparer ce qui est comparable, il faut utiliser `Lua\LaTeX`, puisqu' OpTeX repose sur `Lua\TeX`.

utilisant des macros de haut niveau, comme `\margins` ou `\fontfam`, mais, la plupart du temps, l'utilisateur doit redéfinir des macros internes du format. Nous en donnerons un aperçu dans la section « [Programmer avec OpTeX](#) », pages 51 et suivantes.

Écosystème : manuels, extensions, fontes, dessin, conversion

En utilisant OpTeX, on est amené à se renseigner progressivement sur les fondements et les subtilités du langage TeX. Outre le [TeXBook](#) [4], on peut consulter l'aide mémoire rédigé par Petr Olšák lui-même [5]⁵⁵. Pour se renseigner sur un point précis ou approfondir sa maîtrise du langage, [TeXLive](#) installe automatiquement *TeX by Topic* [6]⁵⁶ et, en français, l'excellent (et technique) ouvrage de Christian Tellechea [7]^{57, 58}. On consultera également la page dédiée sur la [FAQ francophone](#)⁵⁹, en particulier la liste commentée d'ouvrages à laquelle elle renvoie⁶⁰. Puisque OpTeX repose sur [LuaTeX](#), il peut être utile de consulter le manuel de ce moteur [8]⁶¹; si l'on souhaite définir une partie de macro en Lua, ce qui est parfois plus facile qu'en TeX, on peut se référer au manuel officiel du langage [9] ainsi qu'au livre très didactique et efficace écrit par l'un de ses principaux concepteurs [10]. Enfin, OpTeX dispose bien sûr de son propre manuel, fort bien conçu⁶². Outre les livres et les manuels, on peut aussi recourir à la liste de discussion d'OpTeX⁶³, au forum⁶⁴ et à *TeX - LATEX Stack Exchange*⁶⁵.

Bien qu'OpTeX puisse s'utiliser sans charger d'extensions, il inclut un système de modules⁶⁶. Ils sont pour le moment peu nombreux, non seulement parce que la base d'utilisateurs de ce format est relativement réduite comparée à celle de LATEX, mais aussi parce que les utilisateurs sont encouragés à étudier et à modifier les macros internes d'OpTeX ou à créer les leurs en fonction de leurs besoins ponctuels plutôt que d'utiliser des extensions qui fournissent des interfaces plus générales : ainsi, un équivalent de [titlesec](#) pour OpTeX a peu de chances de voir le jour tout simplement parce que les utilisateurs se satisfont de redéfinir les macros de sectionnement présentes dans le format. Les modules existants soit assurent la compatibilité avec une extension préexistante ([emoji](#), [tikz](#), [minim](#)), soit répondent à des besoins génériques ([math](#), [pdfextra](#), [mte](#) pour la micro-typographie, [vlna](#) pour transformer certaines espaces en insécables, par exemple entre une valeur et son unité de mesure). D'autres extensions conçues pour Plain TeX ou reposant uniquement sur les primitives de TeX devraient également fonctionner avec OpTeX, moyennant parfois quelques adaptations ou le chargement du format Plain TeX avec `\input plain`⁶⁷. OpTeX inclut aussi un certain nombre de macros qui chargent

55. [texdoc tex-nutshell](#)

56. [texdoc texbytopic](#)

57. [texdoc apprendre](#)

58. Le même auteur crée également la série de vidéos « [Programmer en TeX](#) », qui est en partie complémentaire de son ouvrage : <https://www.youtube.com/@texperimental/videos>

59. https://faq.gutenberg-asso.fr/1_generalites/documentation/livres/documents_sur_tex.html

60. <http://www.macrotext.net/texbooks/>

61. [texdoc luatex](#)

62. [texdoc optex](#)

63. <https://lists.sr.ht/~sorel/optex>

64. <https://github.com/olsak/OpTeX/discussions>

65. <https://tex.stackexchange.com/questions/tagged/optex>

66. Nous désignons ainsi une extension que l'on peut charger avec `\load` plutôt qu'avec `\input` et qui se conforme pour cela à certaines conventions.

67. Listes : <https://ctan.org/tex-archive/macros/generic/> et <https://ctan.org/tex-archive/macros/plain/contrib>. Exemple d'extension pour Plain TeX qui paraît

automatiquement un fichier \TeX externe sans que l'on ait besoin de déclarer de module dans le préambule, par exemple `\pstart` et `\pend` pour numérotter les lignes en marge (mais là encore, il faut les modifier soi-même si l'on veut, par exemple, afficher uniquement les numéros de ligne divisibles par cinq ou modifier la taille des numéros) ; ces macros dont l'implémentation est relativement brève, et bien d'autres, sont présentées sur la page *Op \TeX tricks*⁶⁸.

Étant basé sur Lua\TeX , Op \TeX permet d'utiliser des fontes **OTF** ou **TTF** en plus des fontes Type1 et **TFM**⁶⁹. Cependant, il ne fournit pas de mécanisme général pour charger ces fontes comparable à `fontspec` pour \LaTeX . On peut utiliser soit la primitive `\font`, soit le système de sélection de fontes d'Op \TeX , qui offre une interface plus commode. Dans ce second cas, chaque fonte doit être déclarée dans un fichier d'initialisation séparé. La $\text{\TeX}Live$ 2025 inclut les fichiers d'initialisation d'une soixantaine de fontes.

Pour enrichir le texte avec des éléments graphiques, Op \TeX fournit diverses macros pour définir des couleurs, colorer ou surligner du texte, l'encadrer dans un cercle ou un ovale, lui faire subir différentes transformations (rotation, etc.), définir arbitrairement les coordonnées d'un objet sur la page et, bien sûr, inclure des images externes. On peut également utiliser `tikz` et MetaPost (via `minim-mp`).

Enfin, pour faciliter la conversion de documents écrits pour Op \TeX vers d'autres formats de fichiers, Petr Olšák a défini le *Op \TeX Markup Language Standard* [11], c'est-à-dire l'ensemble des macros utilisées par un document typique. Pour le moment, il est possible d'exporter vers Op \TeX un document traité par `Pandoc`⁷⁰.

On trouvera quelques éléments supplémentaires sur la page principale d'Op \TeX (<http://petr.olsak.net/optex/>).

Programmer avec Op \TeX

Il est temps maintenant de voir concrètement comment personnaliser la mise en forme ou créer ses propres macros avec Op \TeX . S'il faut le plus souvent utiliser directement les primitives et les paramètres de base de (Plain) \TeX (`\hsize`, `\parskip`, etc.), Op \TeX offre néanmoins diverses fonctionnalités qui facilitent ce travail.

Particularités de la programmation avec Op \TeX

Dans l'exemple page 48, nous n'avons utilisé que des macros utilisateur d'Op \TeX . Cependant, un simple coup d'œil à n'importe quel extrait de source permet de repérer des particularités qui distinguent Op \TeX de Plain \TeX . Voici la définition de `\removespaces`, qui supprime les espaces dans le texte passé en argument (`\removespaces{voici un exemple} \rightarrow voiciunexemple`) :

compatible avec Op \TeX en tenant compte de certaines limites : <https://tex.stackexchange.com/questions/632832/line-numbers-for-optex>

68. <http://petr.olsak.net/optex/optex-tricks.html>

69. Mais voir ici pour les fontes TFM : <http://petr.olsak.net/optex/optex-tricks.html#nonunicode>

70. <https://github.com/spasa47/panoptexum>

Exemple 30 – Extrait de more-macros. opm

```
\_def\removespaces #1 {%
  \isempty{#1}%
  \iffalse #1\_ea\removespaces\_fi
}
\public \removespaces ;
```

On voit tout d'abord que les noms de macros sont préfixés par un tiret bas, y compris les primitives de \TeX comme \def . Ainsi, ces quelques lignes définissent d'abord une macro préfixée $_removespaces$, puis la macro utilisateur \removespaces , qui est déclarée par $_public$ comme égale à $_removespaces$; l'utilisateur n'est censé utiliser que la macro sans tiret bas. De cette manière, l'utilisateur peut, s'il le souhaite, redéfinir \removespaces à sa guise sans que cela n'affecte d'éventuelles utilisations de $_removespaces$ dans d'autres définitions de macros, par le format lui-même ou par des modules externes. Si jamais l'utilisateur décide de redéfinir une macro dont le nom est préfixé, il sait qu'il s'agit d'une macro interne d' $\text{Op}\text{\TeX}$ et qu'il a intérêt à vérifier à quels endroits du code source elle est utilisée. Cette distinction entre macros internes et macros utilisateur s'étend jusqu'aux primitives de \TeX : on peut donc redéfinir \def sans crainte ! À l'inverse, pour maintenir cette distinction et les garanties qui en résultent, il vaut mieux n'utiliser que des macros préfixées dans les définitions de macros que l'on est susceptible d'utiliser dans plusieurs documents. Pour les modules externes, $\text{Op}\text{\TeX}$ implémente un système un peu plus complexe d'espaces de noms que nous ne présentons pas ici (voir le manuel, [1, p. 352] ou [3, p. 125-126]).

On remarque également qu' $\text{Op}\text{\TeX}$ propose des macros qui facilitent la programmation en \TeX : ici $_ea$, un simple alias d' $_expandafter$, et $_isempty$, qui fait partie d'une série de tests comprenant aussi, par exemple, $_ismacro$ ou $_isdefined$. Nous pouvons aussi citer diverses macros de définition comme $_sdef$, qui sert notamment à définir des macros dont le nom est généré dynamiquement; ainsi,

```
\_sdef{c:\_the\_count0}\#1{Voici : #1}
```

est équivalent à :

```
\_ea\def\csname c:\_the\_count0\_endcsname\#1{Voici : #1}
```

Ces macros et d'autres sont présentées dans [3]. Comme on le voit, ces macros auxiliaires ne visent pas, comme expl3 , à créer une interface de haut niveau qui masquerait le code \TeX de bas niveau, mais à étendre \TeX sans en altérer radicalement la physionomie et à offrir des raccourcis commodes pour les procédés les plus courants. À cet égard, il est significatif qu'elles soient introduites dans le chapitre « *Technical documentation* » du manuel, où elles sont présentées avec leur implémentation.

Modifier une macro d' $\text{Op}\text{\TeX}$

Tous ces préalables étant posés, à quoi ressemble concrètement la personnalisation d'un document avec $\text{Op}\text{\TeX}$? Mettons, par exemple, que nous souhaitions que les titres de sections soient sous la forme « A) Titre de section », en grands caractères mais avec une graisse normale : comment s'y prendre ?

On se réfère d'abord au manuel d' $\text{Op}\text{\TeX}$ ⁷¹, chapitre « *User documentation* », où l'on apprend que la mise en forme des macros de sectionnement est implémentée

71. *texdoc optex*

par `_printchap`, `_printsec`, etc., et qu'il faut donc redéfinir ces macros si l'on souhaite la modifier. (Dans ce cas, on peut sans risques redéfinir ces macros internes puisqu'elles sont utilisées uniquement par les macros de sectionnement.) Un clic sur `_printsec` nous conduit au code commenté de son implémentation dans le chapitre « *Technical documentation* », où l'on trouve diverses recommandations pour réécrire correctement ces macros. Voici le code :

Exemple 31

```

1  \_def\_\_printsec#1{\_par
2    \_abovetitle{\_penalty-151}\_bigskip
3    {\_secfont \_noindent \_raggedright
4      \_printrefnum[@\_\quad]#1\_nbpar}%
5    \_insertmark{#1}%
6    \_nobreak \_belowtitle{\_medskip}%
7    \_firstnoindent
8 }
```

On repère que ce sont les lignes 3 et 4 qui nous intéressent. On veut tout d'abord augmenter la taille des caractères dans le titre, ce qui passe probablement par la redéfinition de `_secfont`. Un clic sur `_secfont` nous amène vers la définition de cette macro, qui confirme notre intuition :

Exemple 32

```

\_def\_\_secfont{%
  \_scalemain\_\_typoscale[\_magstep2/\_magstep2]\_boldify
}
```

Il n'y a qu'à copier-coller ce bout de code dans notre préambule en supprimant `_boldify` et en modifiant l'argument de `_typoscale` (après avoir cliqué sur ce nom de macro pour consulter sa documentation, bien sûr).

De même, on peut désormais copier-coller la définition de `_printsec` et, après avoir consulté la documentation des macros appelées si besoin, la modifier comme suit⁷² :

Exemple 33

```

\_let\_\_firstnoindent\_\_relax % Indenter tous les
  paragraphes.

\_def\_\_printsec#1{\_par
  \_abovetitle{\_penalty-151}\_bigskip
  {\_secfont \_noindent \_raggedright
   \_printrefnum[@) ]#1\_nbpar}%
  \_insertmark{#1}%
  \_nobreak \_belowtitle{\_medskip}%
```

72. `_nbpar`, pour « *non-breakable paragraph* », empêche que le titre soit coupé par un saut de page s'il s'étend sur plusieurs lignes. Pour ce qui nous intéresse ici, `_printrefnum[#1@#2]` est grossièrement équivalent à `#1__thesecnum#2`. `_athe` est une macro définie par OpTeX qui affiche la valeur d'un compteur sous la forme d'une lettre de l'alphabet; j'en ai pris connaissance en consultant l'implémentation de `\style A`, qui fait pour les numéros d'éléments dans les listes ce que je cherche à faire ici pour les numéros de titres de sections.

```
\_firstnoindent % On peut supprimer cette ligne.  
}  
  
\_def\_theseignum{\_uppercase\_ea{\_atthe\_secnum}}  
\_def\secfont{\_scalemain\_typoscale[\_magstep3/\_magstep3]}
```

Et voilà!

Maintenant, nous avons vu dans l'exemple 29 page 48 qu'il faut insérer manuellement les espaces insécables ou fines avant et après les signes de ponctuation. On aimerait bien que ces espaces soient insérées automatiquement, comme avec `babel-french` ou `efrench`. Pour cela, nul n'est besoin de consulter un ouvrage en trois gros volumes ou une FAQ en ligne de grande qualité, ni de comparer plusieurs extensions consacrées à la composition du français : on se référera simplement à la section « *Multilingual support* » du chapitre « *Technical documentation* » dans le manuel d'OpTeX. On y trouvera que la macro `_preplangmore` permet de spécifier du code à exécuter quand une langue est sélectionnée ; il faut indiquer dans son argument, à l'aide de `_langreset`, comment annuler les effets de ce code quand on passe à une autre langue. Voici donc une implémentation simple où l'on part du principe que l'on saisira les signes de ponctuation comme dans l'exemple 28, sans les faire précéder d'une espace⁷³ :

Exemple 34

```
\_preplangmore fr {  
    % \_adef rend actif le caractère qui suit  
    % et lui donne une définition.  
    % Dans la définition, le caractère conserve  
    % son ancienne signification.  
    \_adef!{\_, !}%  
    \_adef?{\_, ?}%  
    \_adef;{\_, ;}%  
    \_adef:{~:}%  
    \_adef«{\~, ~}%  
    \_adef»{\~, ~}%  
    \_def\langreset{  
        % Quand une autre langue est sélectionnée,  
        % on attribue à nouveau à ces caractères  
        % le code de catégorie « other »  
        % pour les rendre inactifs.  
        \_catcode`\\!=12 \_catcode`\\?=12  
        \_catcode`\\;=12 \_catcode`\\:=12  
        \_catcode`\\«=12 \_catcode`\\»=12  
    }%  
}
```

^{73.} Si l'on veut prendre en compte le cas où l'utilisateur introduirait lui-même des espaces devant les signes de ponctuation, on peut s'inspirer de l'implémentation de `babel-french` : voir les différentes occurrences de `\declare@shorthand{french}` dans `texmf-dist/tex/generic/babel-french/french.lfd`. Mais pour quoi faire, puisque nous sommes l'utilisateur ?

Au fil du temps, on peut constituer un module personnel contenant les morceaux de code comme celui-ci que l'on utilise le plus souvent en vue de le charger systématiquement au début de chaque document.

Avantages de cette approche

Cette approche, dira-t-on, consiste à réécrire soi-même ce qui a été fait par d'autres au lieu de profiter du travail accumulé de personnes souvent plus expertes que nous. Elle comporte cependant plusieurs avantages : des gains de temps possibles à plusieurs niveaux ; un apprentissage centré sur des concepts et des outils de portée générale ; une meilleure maîtrise du code utilisé.

Étudier le code d' \LaTeX pour le modifier, implémenter soi-même certaines fonctionnalités, peuvent paraître occasionner des pertes de temps que l'on éviterait en utilisant des extensions prêtées à l'emploi. Pourtant, tant que l'on reste à un niveau de complexité raisonnable, écrire sa propre implémentation peut faire gagner du temps : on s'évite de chercher des solutions existantes en ligne, dans des ouvrages ou sur des listes de discussion ; on économise le temps passé à comparer les différentes solutions trouvées, à les essayer, à lire les documentations ; on s'épargne les éventuels problèmes liés aux incompatibilités entre extensions et à leur ordre de chargement. L'implémentation à laquelle on arrive n'est pas aussi générale ni n'a autant de fonctionnalités que celle fournie par une extension existante, mais peu importe, puisqu'elle satisfait notre besoin particulier. Enfin, charger moins d'extensions et exécuter des macros moins complexes réduit le temps de compilation, parfois de manière sensible.

On répondra sans doute que le processus de recherche de solutions pour \LaTeX décrit ci-dessus ne concerne que les cas où l'on rencontre de nouveaux besoins : en général, les utilisateurs apprennent quelles extensions leur sont nécessaires et comment elles fonctionnent. Ce savoir n'a toutefois pas la même portée que celui que l'on acquiert quand on implémente ses propres solutions avec \OpTeX . Dans la plupart des cas, avec \LaTeX , on apprend à utiliser des interfaces qui fonctionnent comme des boîtes noires et que l'on peut, de ce fait, difficilement adapter à des besoins qui n'ont pas été prévus par les personnes qui les ont programmées. Ces interfaces sont en outre multiples et disjointes : il faut maîtriser le fonctionnement propre et la structure de la documentation de plusieurs grosses extensions dotées de nombreuses fonctionnalités comme `memoir` ou `beamer`, ainsi qu'une multitude de petites extensions dont beaucoup possèdent leurs particularités. Avec \OpTeX , à mesure que l'on ajoute ou modifie des fonctionnalités, on apprend comment est implémenté le format, comment utiliser les outils qu'il offre, comment adapter les comportements définis par défaut et, ce qui n'est pas le moindre bénéfice, comment fonctionne le langage \TeX . En manipulant systématiquement les composants de base de \TeX et d' \OpTeX , on développe des compétences plus générales et plus souvent réutilisées. Ces compétences sont en outre transposables à d'autres formats qu' \OpTeX : quand on repasse à \LaTeX , on gagne à savoir implémenter soi-même de nouvelles fonctionnalités, éventuellement en manipulant des primitives de \TeX , et à pouvoir lire le code d'extensions existantes pour le modifier.

Enfin, en étudiant l'implémentation du format, en la modifiant, en écrivant ses propres macros, on acquiert une meilleure connaissance de la base de code sur laquelle repose la composition de nos documents. Cette maîtrise n'est pas utile uniquement pour apporter des modifications. Elle facilite également la correction des dysfonctionnements. Il est plus facile de repérer ses erreurs quand on sait (ou peut comprendre) ce que font les macros que l'on utilise. On est moins

perdu lorsque l'on est confronté à des erreurs de bas niveau de \TeX telles que les mystérieux « *Missing number inserted* ». Les traces, que l'on obtient dans le fichier de log en assignant divers registres $\backslash\text{tracing}<\dots>$ à 1 ou plus, sont plus faciles à exploiter qu'avec \LaTeX , car elles sont moins prolixes et présentent beaucoup moins d'appels à des macros inconnues de l'utilisateur⁷⁴.

Bien sûr, pour que l'apprentissage soit utile, que les connaissances s'approfondissent et que le code écrit par l'utilisateur reste valide au fil des ans, il faut que le format soit stable dans le sens où son amélioration et l'ajout de nouvelles fonctionnalités n'entraînent pas de modifications fréquentes du code existant. C'est, depuis la version 1.0, un principe affiché par le créateur d' $\text{Op}\text{\TeX}$; il paraît généralement suivi⁷⁵.

Limites

$\text{Op}\text{\TeX}$ est utilisable — et utilisé — pour une grande variété de documents et contient les fonctionnalités que l'on attend dans la plupart des cas : listes, tableaux, références croisées, intégration de programmes de dessin, mise en forme automatique de la bibliographie à partir de fichiers $\text{Bib}\text{\TeX}$ (sans recours à un programme externe!), etc. Nous avons vu que l'approche mise en œuvre comporte de nombreux avantages et permet notamment des progrès dans la maîtrise de \TeX qui peuvent aussi s'avérer utiles quand on passe à \LaTeX ou à $\text{Con}\text{\TeXt}$. Pourquoi, tout compte fait, un habitué d' $\text{Op}\text{\TeX}$ continuerait-il à utiliser ponctuellement d'autres formats ?

Il faut reconnaître que l'écosystème d' $\text{Op}\text{\TeX}$ est encore restreint. Peu d'extensions sont officiellement supportées ; $\text{Op}\text{\TeX}$ possède son propre moteur de citations, mais seuls deux formats bibliographiques sont implémentés. Cette situation, nous l'avons dit, tient en partie à l'approche d' $\text{Op}\text{\TeX}$, mais il faut bien reconnaître que, pour certains besoins complexes (par exemple la composition d'un texte et de sa traduction en parallèle), il n'y a guère que quatre solutions : écrire soi-même un nouveau module pour $\text{Op}\text{\TeX}$; écrire une couche de compatibilité pour rendre une extension existante utilisable avec $\text{Op}\text{\TeX}$; réécrire au moins en partie cette extension pour la rendre compatible avec tous les formats \TeX ⁷⁶ ; ou, ce qui est le plus simple dans bien des cas, (re)passer temporairement à \LaTeX .

$\text{Op}\text{\TeX}$ ne permet pas de générer simplement des documents conformes aux standards d'archivage et d'accessibilité PDF/A et PDF/UA . Néanmoins, l'extension **minim-pdf**, compatible avec $\text{Op}\text{\TeX}$, fournit différentes macros qui devraient permettre d'atteindre cette conformité en les intégrant dans le corps du document (par exemple pour les textes alternatifs) et dans la définition de certaines macros (comme $\backslash\text{fnote}$).

Bastien Dumont

74. En ajoutant $\backslash\text{tracingmacros}=1$ et $\backslash\text{tracingcommands}=1$ au début des exemples 28 (\LaTeX) et 29 ($\text{Op}\text{\TeX}$) (p. 48 sq.), on obtient des fichiers de log qui mesurent respectivement 73M et 2,3M. Il semblerait que la différence soit largement due aux complexités de la couche `expl3`.

75. J'ai été confronté une fois à une modification du format qui était incompatible avec une macro que j'avais écrite auparavant ; il s'avère qu'elle était nécessaire à la correction d'un dysfonctionnement (<https://github.com/olsak/OpTeX/commit/65eca4279c0c8b1b6162fe83ea ce6a31d86f7594>). On trouvera l'historique des changements dans le fichier README d' $\text{Op}\text{\TeX}$, qui se trouve ici : <https://github.com/olsak/OpTeX/blob/master/optex/README>.

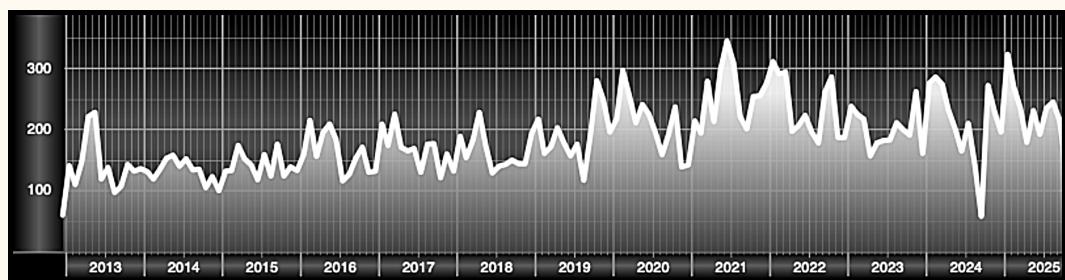
76. Exemple dans la vidéo 35 de la série « Programmer en \TeX » : <https://www.youtube.com/watch?v=QUTSEozpeFY>. Je remercie quark67 de l'avoir signalée sur la liste gut@ens.fr.

Références

- [1] Petr Olšák. « Op \TeX — a new generation of Plain \TeX ». In : *TUGboat* 41.3 (2020), p. 348-354. doi : [10.47397/tb/41-3/tb129olsak-optex](https://doi.org/10.47397/tb/41-3/tb129olsak-optex).
- [2] Petr Olšák. « Comparison of Op \TeX with other formats: \LaTeX and Con \TeX t ». In : *TUGboat* 42.1 (2021), p. 44-49. doi : [10.47397/tb/42-1/tb130olsak-fmtcmp](https://doi.org/10.47397/tb/42-1/tb130olsak-fmtcmp).
- [3] Petr Olšák. « Creating macros in Op \TeX ». In : *TUGboat* 44.1 (2023), p. 121-126. doi : [10.47397/tb/44-1/tb136olsak-optexmac](https://doi.org/10.47397/tb/44-1/tb136olsak-optexmac).
- [4] Donald E. Knuth. *The \TeX Book*. 21^e éd. Reading (Mass.) etc. : Addison-Wesley Professional, 1992. ISBN : 0-201-13448-9.
- [5] Petr Olšák. *\TeX in a Nutshell*. Version 0.10. 3 mars 2024. URL : <http://mirrors.ctan.org/info/tex-nutshell/tex-nutshell.pdf>.
- [6] Victor Eijkhout. *\TeX by Topic: A \TeX nician's Reference*. 2^e éd. Heidelberg : DANTE e.V., Lehmanns Media, 2014. ISBN : 0-201-56882-9.
- [7] Christian Tellechea. *Apprendre à programmer en \TeX* . Sept. 2014. ISBN : 978-2-9548602-0-6. URL : <http://mirrors.ctan.org/info/apprendre-a-programmer-en-tex/output/apprendre-a-programmer-en-tex.pdf>.
- [8] Lua \TeX development team. *Lua \TeX Reference Manual*. Version 1.18. Fév. 2024. URL : <http://mirrors.ctan.org/systems/doc/luatex/luatex.pdf>.
- [9] Lua.org et PUC-Rio. *Lua 5.4 Reference Manual*. 2020. URL : <https://www.lua.org/manual/5.4/>.
- [10] Roberto Ierusalimschy. *Programming in Lua*. 4^e éd. Rio de Janeiro : Lua.org, août 2016. ISBN : 978-859037986. URL : <https://www.lua.org/pil/>.
- [11] Petr Olšák. *Op \TeX Markup Language Standard*. Version 0.1. 2021. URL : <http://petr.olsak.net/ftp/olsak/optex/omls.pdf>.

@@

⌚ ET MAINTENANT, UNE BONNE VIEILLE VEILLE \TeX NOLOGIQUE !



Chers adhérents, nous veillons \TeX xnologiquement pour vous !

En effet, la présente rubrique est dédiée aux nouveautés apparues sur le CTAN que vous auriez pu manquer. Elle témoigne de la vitalité de la communauté \TeX . Nous y listerons la grande majorité des packages ou classes récemment apparus ainsi que parfois, parmi ceux « simplement » mis à jour, certains qui méritent à notre sens d'être signalés. Nous ne nous interdirons pas, le cas échéant, d'en mentionner de plus anciens, soit parce qu'ils nous semblent injustement méconnus, soit parce qu'ils sont les fruits de contributeurs francophones. Au sujet de la francophonie, nous signalons au moyen du logo ☀ les travaux de francophones.