CAHIERS GUTenberg

UN GUIDE POUR LUATEX

Manuel PÉGOURIÉ-GONNARD

Cahiers GUTenberg, nº 54-55 (2010), p. 13-35.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2010___54-55_13_0>

UN GUIDE POUR LUALATEX

Manuel PÉGOURIÉ-GONNARD

RÉSUMÉ. — Ce document est une carte ou un guide touristique pour le nouveau monde de LuaLaTeX[1]. Le public visé par ce document s'étend des curieux n'ayant jamais pratiqué (avec une connaissance de base de LaTeX) aux développeurs d'extensions. Ce guide est destiné à aiguiller le lecteur vers toutes les sources pertinentes sur le sujet, il regroupe les informations qui sont assez dispersées et contient quelques bases d'utilisation de LuaLaTeX.

ABSTRACT. — This document is a map, or touristic guide, for the new world of LuaLaTeX. The intended audience ranges from complete newcomers (with a working knowledge of conventional LaTeX) to package developers. This guide is intended to be comprehensive in the following sense: it contains pointers to all relevant sources, gathers information that is otherwise scattered, and adds introductory material.

NOTE. — Traduction par Maxime Chupin du texte de Manuel Pégourié-Gonnard intitulé *A guide to LuaLaTeX* qui se trouve sur CTAN : `info/luatex/lualatex-doc/lualatex-doc.pdf`.

1. INTRODUCTION

1.1. MAIS C'EST QUOI AU JUSTE LUALATEX ?

Pour répondre à cette question, nous devons mentionner un détail sur *le monde de TeX* que vous pourriez ignorer, à savoir la différence entre un *moteur* et un *format*. Un moteur est un programme informatique alors qu'un format est un ensemble de macros exécuté par un moteur, habituellement préchargé lorsque le moteur est appelé avec un nom particulier.

En fait, un format ressemble plus ou moins à une classe de document ou une extension, sauf que celle-ci est associée à un nom de

1. Bien que se focalisant sur LuaLaTeX, ce document contient aussi des informations utiles et intéressantes sur LuaTeX couplé au format Plain.

commande particulier. Par exemple, on peut imaginer une commande `latex-article` qui réalise la même tâche que la commande habituelle `latex` mis à part le fait que vous n'ayez plus besoin de préciser `\documentclass{article}` au début de votre document. De façon similaire, dans les distributions courantes de TEX, la commande `pdflatex` réalise exactement la même tâche que `pdftex` sauf que vous n'avez pas besoin de mettre l'instruction pour charger LATEX au début de votre fichier source. Ceci est pratique et légèrement plus efficace.

Les formats sont commodes puisqu'ils mettent en œuvre de puissantes macros en utilisant les outils basiques que fournit le moteur. Cependant, les développeurs de formats sont quelques fois limités par l'ensemble des outils du moteur. Ainsi on voit naître, régulièrement, de nouveaux moteurs, apportant de nouvelles fonctionnalités, qui permettent la mise en place de formats (ou extensions) de plus en plus plus évolués. Les moteurs les plus connus du moment sont pdfTEX, XETEX et LuaTEX, et bien entendu l'original TEX mais qui n'est plus guère utilisé.

Pour compliquer un peu plus encore, rappelons que le moteur original TEX ne produit que des fichiers DVI alors que ses successeurs peuvent aussi produire des fichiers PDF. Chaque commande de votre système correspond à un moteur particulier couplé avec un format particulier ainsi qu'un mode de sortie particulier… Le tableau suivant synthétise les diverses possibilités, les lignes correspondent aux formats, les colonnes aux moteurs. Dans chaque cellule, la première ligne correspond, pour le moteur et le format considéré, à la commande pour le mode DVI, la seconde ligne, elle, correspond à la commande pour le mode PDF.

	TEX	pdfTEX	XETEX	LuaTEX
Plain	`tex`	`etex`	(aucune)	`dviluatex`
	(aucune)	`pdftex`	`xetex`	`luatex`
LATEX	(aucune)	`latex`	(aucune)	`dvilualatex`
	(aucune)	`pdflatex`	`xelatex`	`lualatex`

Nous pouvons donc maintenant répondre à la question : LuaLATEX est le moteur LuaTEX couplé avec le format LATEX. Cette réponse n'est évidemment pas satisfaisante si vous ne savez pas ce que sont LuaTEX et LATEX…

Comme vous le savez probablement, LaTeX est l'environnement de travail dans lequel les documents commencent par `\documentclass`, les extensions sont chargées par `\usepackage`, les fontes sont sélectionnées de façon élégante (ainsi vous pouvez passer en gras tout en préservant l'italique) et les pages sont construites avec des algorithmes complexes permettant d'obtenir des en-têtes, des pieds de page, des notes de pied de page, des notes dans la marge, l'utilisation de matériel flottant, etc. Ceci ne change pas avec LuaLaTeX, cependant de nouvelles extensions, plus puissantes, sont utilisables et font que certains aspects de ce système fonctionnent bien mieux.

En somme, qu'est-ce que LuaTeX ? Pour faire court : le moteur le plus en vue à l'heure actuelle ! En développant un peu plus, on peut dire que c'est le successeur désigné de pdfTeX, il inclut donc l'ensemble de ses fonctionnalités de base : la génération directe de fichiers PDF avec un support pour les fonctions PDF avancées ainsi que des améliorations des algorithmes typographiques de TeX sur la micro-typographie. Il inclut bien sûr de nouvelles fonctionnalités dont les principales sont :

1. Le support natif d'Unicode, le standard actuel pour la classification et le codage des caractères, supportant tous les caractères dans le monde, de l'anglais au chinois traditionnel en passant par l'arabe, et incluant une grande partie des symboles mathématiques (ou spécialisés).

2. L'inclusion de Lua comme langage de script embarqué (voir la section 1.3 pour plus de détails).

3. Beaucoup de bibliothèques Lua très pratiques, entre autres :

   — `fontloader`, permettant le support des formats de fontes comme TrueType et OpenType ;

   — `font`, permettant une manipulation avancée des fontes à l'intérieur du document ;

   — `mplib`, une version embarquée du programme MetaPost ;

   — `callback`, donnant accès à certaines parties du moteur TeX qui étaient auparavant inaccessibles au programmeur ;

   — des bibliothèques utiles pour la manipulation d'images, de fichiers PDF, etc.

Certaines de ces fonctionnalités, comme le support d'Unicode, ont un impact direct sur tous les documents, alors que d'autres vont simplement apporter des outils que les auteurs d'extensions pourront utiliser

pour vous fournir des macros plus puissantes, plus efficaces, et bien d'autres améliorations.

1.2. Passer de LaTeX à LuaLaTeX

Comme expliqué dans la section précédente, à quelques petites différences près, LuaLaTeX s'utilise comme LaTeX, mais des outils et extensions plus puissants sont accessibles. Ici, nous allons présenter le minimum nécessaire à la production d'un document avec LuaLaTeX. La suite du document présentera les extensions utilisables plus en détails.

Il y a seulement trois différences majeures dans l'utilisation de LuaLaTeX par rapport à LaTeX :

1. Ne chargez pas inputenc, il suffit de coder votre source en UTF-8.
2. Ne chargez pas fontenc ni textcomp, mais fontspec.
3. N'utilisez pas d'extensions pour changer la police du document, mais utilisez à la place les commandes de l'extension fontspec.

Il vous suffit donc de vous familiariser un peu avec fontspec… Ceci est assez facile. Voici les fondamentaux : on sélectionne la fonte principale à empattement (*serif*) avec `\setmainfont`, la fonte sans empattement (*sans serif*) avec `\setsansfont` et la fonte à chasse fixe avec `\setmonofont`. L'argument de ces trois commandes est le nom de la fonte (celui lisible par un être humain), par exemple `Latin Modern Roman` à la place de `ec-lmr10`. Vous voudrez probablement continuer à utiliser les substitutions usuelles de TeX (ligatures, par exemple `---` pour un tiret cadratin), pour cela, il suffit de rajouter `\defaultfontfeatures{Ligatures=TeX}` avant les commandes de choix de fonte.

La bonne nouvelle c'est que vous avez directement accès à toutes les fontes de votre système d'exploitation (en plus de celles de votre distribution TeX) incluant les fontes TrueType et OpenType avec leurs fonctionnalités les plus avancées. Cela veut dire que, désormais, il est facile d'installer n'importe quelle fonte moderne que vous pouvez télécharger ou acquérir auprès d'un éditeur et l'utiliser avec LuaLaTeX en profitant pleinement de tout son potentiel.

Maintenant, une moins bonne nouvelle : il n'est pas toujours facile d'obtenir la liste de toutes les fontes utilisables. Sous Windows avec TeX Live, l'outil en ligne de commande `fc-list` les liste toutes, mais ce n'est pas très convivial (essayez, vous verrez que ce n'est pas facile à lire).

Sous Mac OS X, l'application *Fontbook* liste les fontes de votre système mais pas celles de votre distribution. Idem sous Linux avec `fc-list`. Encore une plus mauvaise nouvelle : il n'est pas facile d'avoir accès à vos « vieilles » fontes de cette façon. Heureusement, tous les jours, plus de fontes deviennent accessibles dans des formats modernes (tous les mois ou plutôt tous les ans si on ne comptabilise que les *bonnes* fontes).

En passant, notons que la plupart des informations de cette section s'appliquent à XeLaTeX, c'est-à-dire, le format LaTeX avec le moteur XeTeX. En effet, XeTeX partage deux fonctionnalités essentielles avec LuaLaTeX : le support natif d'Unicode et le support des formats modernes de fontes (il n'a pas les autres fonctionnalités de LuaTeX, mais d'un autre côté, actuellement, il est plus stable). Bien que l'implémentation de la gestion des fontes soit très différente, fontspec offre une interface unifiée pour XeLaTeX et LuaLaTeX.

En définitive, pour bénéficier des nouvelles avancées de LuaTeX, seulement quelques aspects du vieux monde qu'est pdfTeX sont à abandonner. En particulier, vous ne pourrez plus utiliser les fontes qui ne sont pas accessibles dans un format moderne. Vous n'aurez plus non plus la liberté de coder vos sources comme bon vous semble, cependant UTF-8 est, à tout point de vue, bien supérieur, ceci n'est donc pas un problème. L'extension luainputenc fournit différents compromis qui vous permettrons de retrouver ces fonctionnalités [1], cela sans doute au prix d'une perte du support réel d'Unicode.

C'est tout ce que vous avez besoin de savoir pour commencer à produire vos documents avec LuaLaTeX. Je vous recommande cependant de jeter un œil au manuel de fontspec et de vous faire la main en essayant de compiler un petit document en utilisant des fontes originales. À partir de là, vous pouvez parcourir le reste de ce document comme bon vous semble. La section 5 liste toutes les autres différences entre l'habituel LaTeX et LuaLaTeX qui me sont connues.

1. Alors que le nom de cette extension suggère qu'elle traite uniquement de le codage d'entrée, il s'avère que les détails de l'implémentation de le codage des fontes pour LaTeX rendent cette extension nécessaire pour la gestion des anciens formats de fontes avec LuaLaTeX.

1.3. Lua dans TEX, une amorce

Lua est un langage très agréable à utiliser, souvent apprécié pour sa compacité, il est évidemment moins surprenant et plus facile à apprendre que TEX en tant que langage de programmation. La référence principale est le très bon *Programming in Lua* dont la première édition est gratuitement disponible en ligne [2]. Pour démarrer rapidement, je vous recommande de lire les chapitres 1 à 5 de la première partie et de jeter un rapide coup d'œil à la partie 3. Notons au passage que toutes les bibliothèques mentionnées dans la partie 3 sont incluses dans LuaTEX, cependant la bibliothèque `os` est restreinte pour des questions de sécurité.

La suite de cet ouvrage vous intéressera différemment en fonction de votre expérience en programmation. La suite de la partie 1 et la partie 2 présentent les fonctionnalités les plus utiles du langage. La partie 4 est, par contre, inutile dans un contexte de base d'utilisation de LuaLaTEX, à part, bien sûr, si vous êtes développeur et que vous voulez bidouiller LuaTEX lui même. Enfin, le manuel de référence de Lua est accessible en ligne [3] et ce, dans de nombreuses langues (ce manuel contient un index plus qu'utile).

Maintenant, intéressons-nous de plus près à Lua dans LuaTEX. La façon principale d'exécuter du code Lua à partir d'un fichier source TEX est d'utiliser la commande `\directlua`, laquelle prend en argument du code Lua. À l'inverse, vous pouvez transférer des informations de Lua à TEX avec `tex.sprint` [4]. Par exemple, le code suivant,

```
voici une approximation
        $\pi =\direclua{tex.sprint(math.pi)}$
```

produit « voici une approximation $\pi = 3.1415926535898$ » dans votre document. Voyez comme il est facile de mélanger TEX et Lua !

En fait, il y a quand même quelques pièges. Intéressons nous tout d'abord au sens Lua vers TEX, c'est le plus simple puisqu'il s'agit principalement de Lua. Si vous regardez dans le manuel de LuaTEX, vous allez

2. Voir `http://www.lua.org/pil/`.
3. Voir `http://www.lua.org/manual/`.
4. Ce nom doit probablement signifier « string print » (écrire une chaîne de caractères) et non pas « courir très vite mais pas longtemps ».

constater qu'il existe une autre fonction au nom plus simple, `tex.print`, qui permet de transmettre l'information de Lua vers TEX. Cette fonction insère virtuellement une ligne complète dans le source TEX qui est composée de l'argument de celle-ci. Au cas où vous ne le sauriez pas, TEX fait des choses un peu vicieuses [5] avec les lignes complètes. Précisément, il ignore les espaces en début et fin de ligne et ajoute un caractère de fin de ligne. La plupart du temps, vous ne voudrez pas que cela arrive, je vous recommande donc l'usage de `tex.sprint` qui ajoute virtuellement son argument dans la ligne courante. Le résultat est alors plus prévisible.

Si vous êtes assez TEXnicien pour connaître le fonctionnement des catcodes, vous serez heureux d'apprendre que `tex.print` et ses variantes vous donnent à peu près un contrôle total sur les catcodes utilisés pour le développement de l'argument en lexèmes puisque vous pouvez spécifier en premier argument de la fonction une *table de catcodes*. Il vous sera probablement nécessaire d'en apprendre plus sur celles-ci (actuellement la section 2.7.6 du manuel de LuaTEX) pour savourer tout la puissance de la chose. Si vous ne connaissez rien aux catcodes, passez ce paragraphe [6].

Allons voir du côté de `\directlua` maintenant. Pour avoir une idée de comment cela fonctionne, imaginez que c'est en quelques sorte une commande `\write`, mais qui écrit dans un fichier virtuel qui est immédiatement envoyé à l'interpréteur Lua. Du côté de Lua, la conséquence de ce traitement est que chaque argument de la commande `\directlua` a sa propre portée : les variables locales d'un appel par `\directlua` ne sont pas visibles par un autre (ceci est plutôt sain, mais toujours bon à savoir).

Maintenant, le principal piège est que, avant d'être envoyé à l'interpréteur Lua, l'argument est tout d'abord traduit en lexèmes par TEX, ensuite il est entièrement développé et le résultat est transformé en simple chaîne de caractères. La lecture par TEX a plusieurs conséquences. En particulier, les fins de lignes sont transformées en espaces, et donc l'interpréteur Lua ne voit qu'une (longue) ligne en entrée. Comme Lua est

5. Oui, ce sont habituellement des actions très utiles, mais dans ce cas, elles seront sans doute surprenantes, c'est pour cela que je les qualifie de vicieuses

6. Oups, trop tard, déjà lu.

un langage où la mise en forme du code est libre, cela importe peu, sauf si vous souhaitez utiliser des commentaires. Considérons le code suivant.

```
\directlua{a_function()
-- un commentaire
another_function()}
```

Celui-ci ne fera sans doute pas ce que vous espérez, `another_function()` sera perçu par l'interpréteur comme une partie du commentaire puisqu'il ne s'agit pour lui que d'une seule ligne.

La lecture de l'argument par TEX fait fusionner les espaces successifs en un seul, et aussi supprime les commentaires TEX (tout ce qui commence par %). Donc, voici une version correcte, mais surprenante, de l'exemple précédent.

```
\directlua{a_function()
% un commentaire
another_function()}
```

Il convient également de noter que puisque l'argument de `\directlua` est fondamentalement à l'intérieur d'un `\write`, il faut être vigilant car TEX est alors dans un contexte de développement seul (l'argument sera développé mais pas exécuté). Si vous ne savez pas ce que cela signifie, laissez-moi simplement vous dire, sans avoir à développer plus sur le sujet, que les questions de développement sont, en grande partie, ce qui rend la programmation en TEX si difficile.

Je suis désolé si les trois derniers paragraphes vous ont paru un peu TEXniques mais je pense qu'il vous fallait passer par là. Pour vous récompenser d'être resté avec moi, voici une astuce de débogage. Mettez le code suivant dans le préambule de votre document :

```
\newwrite\luadebug
\immediate\openout\luadebug luadebug.lua
\AtEndDocument{\immediate\closeout\luadebug}
\newcommand\directluadebug{\immediate\write\luadebug}
```

Ensuite, lors de votre apprentissage de LuaTEX, lorsque vous aurez passé longtemps à comprendre pourquoi tel ou tel appel à `\directlua` ne fonctionne pas comme vous l'espériez, remplacez l'instance en question par `\directluadebug` ainsi définie, compilez comme d'habitude

et jetez un coup d'œil dans le fichier `luadebug.lua` produit. Il contient exactement ce que l'interpréteur Lua a réellement lu.

L'extension luacode fournit des commandes et des environnements qui aident plus ou moins à résoudre quelques-uns de ces problèmes. Cependant, à part pour des morceaux de code triviaux, il est plus sage d'utiliser un fichier externe ne contenant que du code Lua définissant des fonctions, pour ensuite le charger et utiliser les fonctions ainsi définies. Par exemple :

```
\directlua{dofile("my-lua-functions.lua")}
\newcommand*{\greatmacro}[2]{%
  \directlua{
    my_great_function("\luatexluaescapestring{#1}", #2)}
}
```

Cet exemple suppose que `my_great_function` est défini dans le fichier `my-lua-functions.lua` et qu'elle prend comme argument une chaîne de caractères et un nombre. Notez que l'on utilise prudemment la primitive `\luatexluaescapestring` sur le premier argument (chaîne de caractères) pour protéger les caractères backslash ou double quote qui pourraient induire en erreur le parseur Lua [7].

C'est fini en ce qui concerne Lua à partir de TEX. Si vous vous demandez pourquoi la commande `\luatexluaescapestring` a un nom si long et si bizarre, la section suivante peut vous intéresser.

1.4. D'autres choses que vous devez savoir

Au risque d'enfoncer une porte ouverte, le manuel de LuaTEX, `luatexref-t.pdf`, est une source d'informations très importante sur LuaTEX et il vous faudra sans doute le consulter sur quelques points (attention, c'est assez aride et technique).

Il est important de savoir que le nom des nouvelles primitives de LuaTEX que vous pourrez lire dans le manuel ne sont pas les noms que vous pourrez utilisez avec LuaLATEX. Pour éviter les conflits avec des macros existantes, toutes les primitives ont été affublées du préfixe `\luatex` sauf si elles commençaient déjà par cela. Ainsi, `\luaescapestring`

7. Si vous avez déjà utilisé SQL ce concept n'est pas nouveau pour vous.

devient `\luatexluaescapestring` alors que `\luatexversion` reste `\luatexversion`. La raison en est détaillée dans la section 4.

Au fait, je n'ai pas mentionné que LuaTEX est actuellement en test et que la version 1.0 est attendue pour 2012. Pour en savoir plus sur la planification de son avancée, allez voir sur le site LuaTEX.[8] Les versions beta stables sont régulièrement publiées et sont incluses dans TEX Live depuis 2008 et dans MikTEX depuis la version 2.9.

Le support de LuaTEX dans LATEX est évidemment très récent, ceci signifie qu'il y a probablement tout un tas de bugs (plus ou moins importants), et que les choses peuvent très vite évoluer. Il vous faut donc garder votre distribution TEX le plus à jour possible[9] et éviter, pour quelques temps encore, d'utiliser LuaLATEX pour des documents cruciaux.

2. Pratiques et extensions essentielles

Cette section présente les extensions que vous devez toujours charger si vous êtes utilisateur, ou que vous devez absolument connaître si vous êtes développeur.

2.1. Pour l'utilisateur

Fontspec

Moteur : XeTEX, LuaTEX. Formats : LATEX.

Auteurs : Will Robertson.

Emplacement sur CTAN : `macros/latex/contrib/fontspec/`.

Url de développement : `https://github.com/wspr/fontspec/`.

Superbe interface pour la gestion des fontes, très bien intégrée dans le système de sélection de fontes de LATEX. Extension déjà présentée dans la section précédente.

2.2. Pour le développeur

2.2.1. *Conventions de nommage*

Du côté de TEX, les séquences de contrôle commençant par `\luatex` sont réservées pour les primitives. Il est fortement recommandé de *ne*

8. Voir `http://luatex.org/`.

9. Pour TEX Live, vous pouvez utiliser le dépôt complémentaire tlcontrib, voir `http://www.gutenberg.eu.org/?TLContrib`.

pas définir de telles séquences de contrôle pour éviter tout conflit avec les versions futures de LuaTeX. Si vous voulez souligner le fait qu'une macro est spécifique à LuaTeX, nous vous recommandons d'utiliser à la place le préfixe `\lua` (sans être suivi de `tex`). L'utilisation de `\luatex@` convient pour les macros internes puisque les noms de primitives ne contiennent jamais `@`, cependant cela peut rendre les choses un peu confuses. De plus, vous utilisez déjà un préfixe unique pour toutes les macros internes de vos extensions, n'est-ce pas ?

Du côté de Lua, il est préférable de laisser l'espace de noms aussi propre que possible. C'est-à-dire que vous devez utiliser une table `mypackage` et mettre vos objets et fonctions publics dedans. Vous pourrez, pour cela, utiliser la fonction `module()` [10] de Lua. D'autres stratégies pour la gestion des modules en Lua sont exposées dans le chapitre 15 de *Programming in Lua* [11]. Aussi, l'utilisation de `local` pour vos variables et fonctions internes ne peut-être qu'une bonne idée. Enfin, toujours dans le but d'éviter des éventuels conflits avec les futures versions de LuaTeX, il est recommandé de ne pas modifier les espaces de noms des bibliothèques incluses dans LuaTeX.

2.2.2. *Détection du moteur et du mode*

Plusieurs extensions permettent de détecter le moteur utilisé pour composer le document.

Ifluatex
MOTEUR : tous. FORMATS : LaTeX, Plain.
AUTEURS : Heiko Oberdiek.
EMPLACEMENT SUR CTAN : `macros/latex/contrib/oberdiek/`.
Fournit `\ifluatex` et vérifie que `\luatexversion` est bien utilisable.

Iftex
MOTEUR : tous. FORMATS : LaTeX, Plain.
AUTEURS : Vafa Khalighi.
EMPLACEMENT SUR CTAN : `macros/latex/contrib/iftex/`.
URL DE DÉVELOPPEMENT : `http://bitbucket.org/vafa/iftex`.

10. Voir `http://www.lua.org/manual/5.1/manual.html#pdf-module`.
11. Voir `http://www.lua.org/pil/15.html`.

Fournit `\ifPDFTeX`, `\ifXeTeX`, `\ifLuaTeX` et les commandes `\Require` correspondantes.

Expl3

MOTEUR : tous. FORMATS : LaTeX.

AUTEURS : The LaTeX3 Project.

EMPLACEMENT SUR CTAN : `macros/latex/contrib/expl3/`.

URL DE DÉVELOPPEMENT : `http://www.latex-project.org/code.html`.

Parmi beaucoup d'autres choses, fournit `\luatex_if_engine:TF`, `\xetex_if_engine:TF` et leurs variantes.

Ifpdf

MOTEUR : tous. FORMATS : LaTeX, Plain.

AUTEURS : Heiko Oberdiek.

EMPLACEMENT SUR CTAN : `macros/latex/contrib/oberdiek/`.

Fournit le test `\ifpdf`. LuaTeX, comme pdfTeX, et peut produire soit du PDF, soit du DVI ; le dernier format de sortie n'est que peu utile avec LuaTeX puisqu'il ne support aucune fontionnalité avancée telle qu'Unicode et les formats de fontes modernes. Le test `\ifpdf` est vrai si et seulement si vous lancez pdfTeX ou LuaTeX dans le mode PDF (notons au passage que celui-ci n'inclut pas XeTeX pour lequel le support PDF est différent).

2.2.3. *Les ressources de base*

Luatexbase

MOTEUR : LuaTeX. FORMATS : LaTeX, Plain.

AUTEURS : Élie Roux & Manuel Pégourié-Gonnard.

EMPLACEMENT SUR CTAN : `macros/luatex/generic/luatexbase/`.

URL DE DÉVELOPPEMENT : `https://github.com/mpg/luatexbase`.

Les formats Plain et LaTeX fournissent des macros pour gérer les ressources de base de TeX comme par exemple les registres de compteurs ou de boîtes. LuaTeX introduit de nouvelles ressources qui doivent être partagées par les différentes extensions. luatexbase fournit les outils de bases pour y parvenir : les outils conventionnels de TeX, les tables de catcodes, les attributs, les *callbacks*, le chargement et l'identification des modules Lua. Cette extension fournit aussi les outils pour gérer quelques problèmes de compatibilité avec les versions précédentes de LuaTeX.

Attention. — Cette extension est actuellement en conflit avec l'extension luatex puisque qu'elles apportent principalement les mêmes fonctionnalités. Les auteurs respectifs sont bien au courant de la situation et ont prévu de fusionner ces deux extensions dans un futur proche, bien que la date ne soit pas encore tout à fait fixée.

Luatex
Moteur : LuaTeX. Formats : LaTeX, Plain.
Auteurs : Heiko Oberdiek.
Emplacement sur CTAN : `macros/latex/contrib/oberdiek/`.
Voir la description de luatexbase ci-dessus. Cette extension fournit les même fonctionnalités de base, excepté pour la gestion des *callbacks* et l'identification des modules Lua.

Lualibs
Moteur : LuaTeX. Formats : Lua.
Auteurs : Élie Roux.
Emplacement sur CTAN : `macros/luatex/generic/lualibs/`.
Url de développement : `https://github.com/mpg/lualibs`.
Collection de bibliothèques Lua et d'ajouts aux bibliothèques standards ; la plupart étant dérivées de celles de ConTeXt. Si vous avez besoin d'une fonction de base que Lua ne contient pas, vérifiez ce que propose cette extension avant de vous lancer dans votre propre implémentation.

2.2.4. *Gestion interne des fontes*
Toutes les extensions de cette section sont chargées par fontspec pour gérer les problèmes de bas niveau de fonte et de codage. Un utilisateur classique ne devrait avoir qu'à utiliser fontspec, mais un développeur aura peut-être à les connaître.

Luaotfload
Moteur : LuaTeX. Formats : LaTeX, Plain.
Auteurs : Élie Roux & Khaled Hosny.
Emplacement sur CTAN : `macros/luatex/generic/luaotfload/`.
Url de développement : `https://github.com/khaledhosny/luaotfload`.
Chargement bas niveau des fontes OpenType, adaptation d'un sous-ensemble générique de ConTeXt. Schématiquement, cette extension utilise la bibliothèque Lua `fontloader` et des *callbacks* appropriés pour

implémenter une syntaxe pour la primitive `\font`, celle-ci étant très similaire à celle de XeTeX. Elle gère aussi une base de données de fontes pour un accès transparent aux fontes du système d'exploitation ainsi qu'aux fontes de la distribution TeX aussi bien par nom de famille de fontes que par nom de fichier, ainsi qu'un cache des données des fontes pour un chargement plus rapide.

Euenc
MOTEUR : XeTeX, LuaTeX. FORMATS : LaTeX.
AUTEURS : Will Robertson, Élie Roux & Khaled Hosny.
EMPLACEMENT SUR CTAN : `macros/latex/contrib/euenc/`.
URL DE DÉVELOPPEMENT : `https://github.com/wspr/euenc`.

Implémente le codage Unicode `EUx` pour le système fontenc de LaTeX. Actuellement, XeLaTeX utilise `EU1` et LuaLaTeX `EU2`. Cette extension inclut les définitions (fichiers `fd`) pour Latin Modern, la fonte chargée par défaut par fontspec.

Pour être précis, euenc ne fait que déclarer le codage, mais ne fournit pas les définitions pour les macros LICR : celles-ci s'obtiennent en chargeant xunicode avec `\UTFencname` défini comme `EU1` ou `EU2`, ce que fait fontspec. Les codages en eux-mêmes sont donc identiques, mais il est utile d'avoir deux noms distincts pour que des fichiers `fd` différents puissent être utilisés en fonction du moteur (ce qui est effectivement le cas pour Latin Modern).

3. AUTRES EXTENSIONS

Les extensions suivantes sont listées dans un ordre plutôt aléatoire.

3.1. POUR L'UTILISATEUR

Luatextra
MOTEUR : LuaTeX. FORMATS : LaTeX.
AUTEURS : Élie Roux & Manuel Pégourié-Gonnard.
EMPLACEMENT SUR CTAN : `macros/luatex/generic/luatextra/`.
URL DE DÉVELOPPEMENT : `https://github.com/mpg/luatextra`.

Charge les extensions usuelles pour LuaTeX, actuellement fontspec, luacode, metalogo (macros pour obtenir les différents logos, en particulier `\luatex` et `\LuaLaTeX`), luatexbase, lualibs, fixltx2e (correctifs et améliorations du noyau de LaTeX).

Luacode

MOTEUR : LuaTEX. FORMATS : LATEX.

AUTEURS : Manuel Pégourié-Gonnard.

EMPLACEMENT SUR CTAN : `macros/luatex/latex/luacode/`.

URL DE DÉVELOPPEMENT : `https://github.com/mpg/luacode`.

Fournit des commandes et macros qui aident à l'inclusion de code Lua dans le source TEX, notamment en ce qui concerne les caractères spéciaux.

Luainputenc

MOTEUR : LuaTEX, XETEX, pdfTEX. FORMATS : LATEX.

AUTEURS : Élie Roux & Manuel Pégourié-Gonnard.

EMPLACEMENT SUR CTAN : `macros/luatex/latex/luainputenc/`.

URL DE DÉVELOPPEMENT : `https://github.com/mpg/luainputenc`.

Aide à la compilation des documents utilisant de vieux codages (soit dans le source soit avec les fontes). Déjà présentée dans l'introduction, cette extension s'utilise aussi bien avec les trois moteurs, elle charge `xetex-inputenc` avec XETEX et `inputenc` habituelle avec pdfTEX.

Luamplib

MOTEUR : LuaTEX. FORMATS : LATEX, Plain.

AUTEURS : Hans Hagen, Taco Hoewater & Élie Roux.

EMPLACEMENT SUR CTAN : `macros/luatex/generic/luamplib/`.

URL DE DÉVELOPPEMENT : `https://github.com/mpg/luamplib`.

Fournit une interface pour la bibliothèque Lua `mplib` qui embarque metapost dans LuaTEX.

Luacolor

MOTEUR : LuaTEX. FORMATS : LATEX.

AUTEURS : Heiko Oberdiek.

EMPLACEMENT SUR CTAN : `macros/latex/contrib/oberdiek/`.

Change l'implémentation bas niveau des couleurs en utilisant les attributs de LuaTEX. Cela permet une implémentation plus robuste et corrige certains bogues tels que le mauvais alignement lorsque `\color` se trouve au début d'une `\vbox` [12].

12. On trouvera plus de détails dans l'article « Attributs et couleurs » du présent numéro des *Cahiers*.

Luadirections

MOTEUR : LuaTEX. FORMATS : LATEX, Plain, ConTEXt.

AUTEURS : Khaled Hosny.

URL DE DÉVELOPPEMENT : `https://github.com/khaledhosny/luadirections`.

Interface de haut niveau pour le support de l'écriture dans les différentes directions de LuaTEX. Actuellement, cette extension n'est pas présente sur CTAN.

3.2. POUR LE DÉVELOPPEUR

Pdftexcmds

MOTEUR : LuaTEX, pdfTEX, XETEX. FORMATS : LATEX, Plain.

AUTEURS : Heiko Oberdiek.

EMPLACEMENT SUR CTAN : `macros/latex/contrib/oberdiek/`.

Bien que LuaTEX soit principalement un sur-ensemble de pdfTEX, quelques primitives ont été supprimées (celles qui sont, en quelques sortes, remplacées par Lua) ou renommées. Cette extension leur attribue des noms cohérents pour l'ensemble des moteurs, y compris XETEX qui offre depuis peu quelques-unes de ces primitives, par exemple `\strcmp`.

Magicnum

MOTEUR : LuaTEX, pdfTEX, XETEX. FORMATS : LATEX, Plain.

AUTEURS : Heiko Oberdiek.

EMPLACEMENT SUR CTAN : `macros/latex/contrib/oberdiek/`.

Fournit un accès hiérarchique aux « nombres magiques » que sont les catcodes, les types de groupes, etc. utilisés par le moteur TEX et ses successeurs. Avec LuaTEX, une implémentation plus efficace est utilisée avec une interface Lua.

Lua-alt-getopt

MOTEUR : `texlua`. FORMATS : command-line.

AUTEURS : Aleksey Cheusov.

EMPLACEMENT SUR CTAN : `support/lua/lua-alt-getopt`.

URL DE DÉVELOPPEMENT : `http://luaforge.net/project/lua_altgetopt`.

Il s'agit d'un parseur d'option de lignes de commandes, il est globalement compatible avec les getopt POSIX et GNU. Il s'utilise avec les scripts

Lua en ligne de commande tels que `mkluatexfontdb` fourni par luaotfload.

4. Les formats `luatex` et `lualatex`

Cette section est destinée aux développeurs et aux utilisateurs curieux uniquement ; les utilisateurs classiques peuvent donc sereinement passer leur route. Les informations qui suivent sont valables pour la distribution TeXLive 2010, et, pour la plupart, pour MikTeX 2.9, bien qu'en réalité, je n'ai pas vérifié. Les versions précédentes de TeXLive sont assez différentes et les formats fournis sont bien moins complets.

Les noms de primitives

Comme mentionné dans la section 1.4, les noms des primitives spécifiques à LuaTeX ne sont pas les mêmes dans le format `lualatex` que dans le manuel de LuaTeX. Dans le format `luatex` – c'est-à-dire LuaTeX avec le format Plain – les primitives sont accessibles avec leurs noms naturels ainsi qu'avec le nom préfixé, cela pour faciliter le développement d'extensions génériques.

Les raisons de ce changement de noms sont les suivantes [13] :

1. Toutes les macros des extensions actuelles fonctionnent bien avec pdf(e)TeX, donc ces primitives sont laissées inchangées.

2. Les autres primitives de LuaTeX qui n'existent pas dans TeX82 peuvent être incompatibles avec les macros existantes dans les extensions, particulièrement quand elles utilisent des noms très « naturels » tels que `\outputbox`, `\mathstyle` etc. Cette probabilité de conflits de noms est gênante ; plus les documents LaTeX déjà existants peuvent être compilés avec LuaTeX, mieux c'est.

3. L'équipe LuaTeX ne souhaite pas appliquer une politique de préfixage systématique, mais, heureusement, les développeurs nous fournissent un outil qui permet d'en appliquer. Nous avons choisi de l'utiliser. Dans un premier temps, nous avions désactivé les primitives spécifiques à LuaTeX, maintenant nous trouvons qu'il est préférable des les

13. Ces raisons sont recopiées à partir du fichier `lualatexiniconfig.tex` qui implémente ce changement de noms.

activer systématiquement avec un préfixe dans le but d'éviter que les extensions de macros (ou les macros des utilisateurs) les activent avec des préfixes variés et incohérents (en particulier avec le préfixe vide).

4. Le préfixe `luatex` a été choisi puisqu'il est déjà utilisé comme préfixe pour certaines primitives, telles que `\luatexversion`. De cette façon, ces primitives ne sont pas affublées d'un double préfixe (pour plus de détails, allez voir `tex.enableprimitives` dans le manuel de LuaTeX).

5. La primitive `\directlua` est fournie avec son nom naturel (facilitant la détection de LuaTeX) et avec son nom préfixé `\luatexdirectlua`. Il en est de même pour `\luatexlatelua`.

6. Quelques remarques :

— Il y a un inconvénient évident d'une telle règle de préfixes. Les noms utilisés par LaTeX ou par les développeurs de macros génériques ne vont pas correspondre aux noms utilisés dans le manuel. Nous pensons que cela est compensé par le gain de la compatibilité descendante.

— Toutes le primitives traitant de la gestion Unicode des maths commencent pas `\U`. Sans doute un jour elles correspondront avec les noms des primitives de XeTeX, donc ce préfixe ne sera plus nécessaire. Cependant, nous essayons de rendre les règles de préfixes aussi simples que possible, de sorte que le point précédent ne s'aggrave pas plus.

— Les noms de quelques primitives ainsi obtenues peuvent sembler bizarre, notamment ceux qui contiennent déjà le nom d'un moteur, par exemple `\luatexOmegaVersion`. Cependant, puisque LuaTeX n'est pas un remplaçant pour Omega/Aleph, nous pensons qu'il n'est pas judicieux de fournir `\OmegaVersion`.

— Peut-être qu'un jour, nous préférerons fournir toutes les primitives sans préfixes. Si cela arrive, il sera facile d'ajouter les primitives non préfixées dans le format, tout en gardant les noms préfixés pour la compatibilité. Cela n'aurait pas fonctionné dans l'autre sens, c'est-à-dire, que la prise de conscience tardive du fait que nous n'aurions pas dû fournir les primitives non préfixées aurait empêché le fonctionnement des extensions spécifiques à LuaTeX déjà écrites.

`\jobname`

Le noyau de LaTeX (ainsi qu'un grand nombre d'extensions) utilise des constructions telles que `\input\jobname.aux` pour réaliser des choses diverses. Quand `\jobname` contient des espaces, ceci ne produit pas le résultat escompté puisque l'argument de `\input` prend fin au premier espace. Pour traiter ce problème, pdfTeX met entre quotes `\jobname` automatiquement, cependant LuaTeX ne le fait pas pour plusieurs raisons. Une solution presque complète est incluse dans les formats LuaTeX basés sur LaTeX (mais pas ceux basés sur Plain).

Cependant, cela ne fonctionne pas si LuaTeX est invoqué de la façon suivante : `lualatex '\input name'`, plutôt que la façon plus usuelle `\lualatex name`. Pour contourner cette difficulté, il suffit de spécifier explicitement le *jobname*, par exemple `lualatex -jobname=name '\input name'`. Ou alors, et ce n'est pas plus mal, vous pouvez ne pas utiliser d'espace dans vos noms de fichiers TeX.

Pour plus de détails, veuillez consulter ce vieux fil de discussion [14] ainsi que celui-ci [15], plus récent, sur la liste de diffusion LuaTeX. Pour l'implémentation de la solution au problème, nous renvoyons au fichier `lualatexquotejobname.tex`.

Césures

LuaTeX permet un chargement dynamique des motifs de coupure de mots. Ceci n'est actuellement pas supporté par l'extension babel mais quelques fichiers ont été modifiés pour permettre un chargement semi-dynamique et pour obtenir un meilleur temps de chargement du format. Ceci n'est qu'un changement d'implémentation, rien ne change au niveau de l'utilisation. Un nouveau schéma de césure est aussi utilisé pour les formats basés sur Plain.

La documentation et les détails de mise en œuvre sont disponibles dans `luatex-hyphen.pdf`. Les sources font partie du projet *texhyphen* [16].

14. Voir `http://www.ntg.nl/pipermail/dev-luatex/2009-April/002549.html`.
15. Voir `http://tug.org/pipermail/luatex/2010-August/001986.html`.
16. Voir `http://tug.org/tex-hyphen/`.

Codes

Le moteur en lui-même, ne règle pas les `\catcodes`, `\lccodes`, etc. pour les caractères non ASCII. En particulier, avoir les bons `\lccodes` est essentiel pour l'algorithme de coupures de mots. Les formats pour LuaTeX appellent maintenant `luatex-unicode-letters.tex`, une version modifiée de `unicode-letters.tex` utilisé par XeTeX, qui prend soin d'assigner ces valeurs en accord avec le standard Unicode.

Ceci a été ajouté après que la distribution TeX Live 2010 fut sortie, donc il est fortement conseillé de mettre à jour votre installation si vous voulez bénéficier de coupures de mots correctes pour le texte non ASCII.

5. Ce qui fonctionne très bien, un peu, pas du tout (pour l'instant)

5.1. Ce qui fontionne

Unicode

Le LaTeX traditionnel offre un certain support de l'UTF-8 dans les fichiers d'entrée. Cependant, au niveau le plus bas, les caractères non ASCII ne sont pas atomiques, ils sont décomposés en plusieurs pièces élémentaires (connues sous le nom de *lexèmes* par les TeXniciens). Ainsi, quelques extensions qui analysent le texte caractère par caractère ou qui font d'autres opérations atomiques sur les caractères (comme changer leurs catcodes par exemple) ont quelques difficultés avec LaTeX traditionnel. Par exemple, vous ne pouvez pas utiliser des caractères non ASCII dans les environnements *verbatim* avec fancyvbr, etc.

La bonne nouvelle, c'est qu'avec LuaLaTeX quelques-unes des fonctionnalités de ces extensions vont être opérationnelles sur n'importe quel caractère Unicode et cela sans avoir à modifier l'extension. La mauvaise nouvelle, c'est que ce n'est pas toujours vrai. Allez voir la section suivante pour plus de détails.

5.2. Ce qui fonctionne partiellement

Microtype

L'extension microtype n'est que peu fonctionnelle avec LuaTeX, plus précisément, avec la version 2.4 2010/01/10, les phénomènes de

ponctuation pendante[17] et de *jeu sur la chasse*[18] fonctionnent et sont activées par défaut en mode PDF. Cependant l'ajustement des approches, les espacements et les alignements ne sont pas supportés (voir table 1 de la section 3.1 du fichier `microtype.pdf`).

D'un autre côté, luaotfload, chargée par fontspec, met en place un grand nombre de fonctionnalités microtypographiques. Le seul problème est donc qu'il n'existe pas d'interface unifiée.

Xunicode

La principale utilité de l'extension xunicode est de s'assurer du bon fonctionnement, dans le contexte d'Unicode, des séquences habituelles en TEX pour les caractères non ASCII, tels que `\'e` par exemple. Cette extension devrait normalement fonctionner avec LuaTEX, mais son utilisation n'a été prévue que pour XeTEX. Cependant, fontspec utilise une astuce pour la charger dans tous les cas. Ainsi, pour résumer, vous ne pouvez pas charger cette extension[19] avec LuaTEX, mais vous n'avez pas besoin de le faire puisque fontspec s'en charge.

Codages

Comme mentionné dans la section précédente, quelques petites choses, qui sont problématiques avec l'utilisation d'UTF-8, fonctionnent normalement avec LATEX conventionnel, mais pas toujours. Par exemple, avec l'extension listings et LuaLATEX, vous ne pouvez utiliser que les caractères en dessous de 256 (c'est-à-dire les caractères de l'ensemble latin-1) dans vos *listings* (mais, bien entendu, toute l'étendue des caractères Unicode reste utilisable dans le reste de votre document).

Métriques

Ce point n'est pas exactement sur ce qui fonctionne ou non, mais plutôt, sur ce qui ne fonctionne pas de la même manière qu'avec pdfTEX

17. En anglais *protrusion*; c'est le fait que certains caractères de ponctuation peuvent déborder légèrement dans la marge pour donner une meilleure impression visuelle d'alignement.

18. En anglais *expansion*; c'est le fait que les glyphes peuvent être légèrement déformés pour que, selon le cas, ils chassent plus ou moins dans le but de mieux remplir une ligne.

19. Sous peine d'obtenir un doux message : *this package currently works only with XeTEX. [N.D.T.]*

ou XeTeX. En effet, vous pourrez constater quelques différences dans la disposition et la coupure des mots de vos textes lorsque le document est compilé avec LuaLaTeX.

Cela peut-être dû aux variations entre deux versions de la même fonte utilisées par les divers moteurs, à l'ajustement fait sur les algorithmes de coupures de mots, de ligatures ou d'espacements automatiques (par exemple, le premier mot d'un paragraphe, comme les mots contenant plusieurs fontes, peuvent maintenant être coupés), ou alors cela peut provenir des changements dans les motifs de césures de mots (les motifs utilisés par pdfTeX sont fondamentalement gelés, mais LuaTeX et XeTeX utilisent de nouvelles versions pour quelques langages).

Si vous observez une différence majeure entre les résultats des compilations avec LuaLaTeX et pdfLaTeX avec les mêmes fontes, il n'est pas improbable que cela provienne d'un bogue dans LuaTeX[20] ou dans la fonte utilisée. Comme d'habitude, vérifiez que votre distribution est à jour avant de rapporter un tel problème.

Babel

Fonctionne globalement bien pour la plupart des langues latines. Pour les autres, cela peut être fluctuant. Même pour les langues latines, des problèmes liés aux codages peuvent se présenter. Par exemple, en français, les commandes `\og` et `\fg` ne donnaient pas de beaux guillemets avec frenchb 2.3d, mais fonctionnent parfaitement avec la version 2.5b (pas encore sur CTAN, donc à télécharger sur la page de l'auteur[21]).

Une extension, plus moderne mais moins complète, pour le support d'écriture dans différentes langues existe pour XeLaTeX mais n'est pas encore portée sur LuaLaTeX, il s'agit de polyglossia.

5.3. Ce qui ne fonctionne pas (encore)

Vieux codages

Les extensions manipulant les codages d'entrée (fichier source) ou de sortie (fontes) sont fortement susceptibles de ne plus fonctionner avec

20. Par exemple, LuaTeX 0.60 avait un bogue qui empêchait toute coupure de mots après une ligature `---` jusqu'à la fin du paragraphe.

21. Voir `http://daniel.flipo.free.fr/frenchb/index.html`.

LuaTEX : les plus connues sont inputenc, fontenc et textcomp, et probablement la plupart des paquets de fontes comme mathptmx ou fourier. Heureusement, l'utilisation d'Unicode est un bien meilleur moyen de régler les problèmes de codages que les vieilles extensions essayaient de résoudre. Par conséquent, vous ne devrez plus utiliser ce type d'extensions. Cependant, comme tout n'est pas encore porté dans le monde d'Unicode, vous aurez sans doute un ensemble de choix possibles plus restreint (ou juste différent) pour quelque temps encore (ceci est spécialement vrai pour les fontes).

Soul

L'extension soul utilise une astuce diabolique avec une fonte à chasse fixe pour compter les caractères. Cependant, à cause des différences dans la manipulation des fontes (et peut-être aussi d'un bogue dans LuaTEX), cela ne fonctionne plus avec les caractères au dessus de 256 (c'est-à-dire en dehors de l'ensemble latin-1).

Espaces

Les espaces dans les noms de fichiers ne sont pas, en général, réellement bien supportés dans le monde de TEX. Cela ne s'arrange pas vraiment avec LuaTEX. Aussi, pour des raisons techniques, les choses peuvent être pires si vous avez des espaces dans le nom de votre fichier maître TEX *et* que vous n'invoquez pas LuaTEX de façon classique. Si vous l'appelez normalement, tout devrait fonctionner, et sinon, je ne vous dirai pas à quoi ressemble l'invocation non usuelle qui empêche le bon fonctionnement. En fait si, vous pouvez consulter le point sur le jobname dans la section 4 pour la solution de contournement et les détails techniques. Ou, mieux encore, n'utilisez pas d'espaces dans vos noms de fichiers TEX.

◖ Manuel Pégourié-Gonnard
Institut de mathématiques de Jussieu
mpg@elzevir.fr
◖ Maxime Chupin
mc@melusine.eu.org