Un guide touristique de LuaLaTeX*

Manuel Pégourié-Gonnard <mpg@elzevir.fr>

5 mai 2013

Ce document se veut un guide touristique du nouveau monde offert par LuaLaTeX. [1] Le public visé va des nouveaux venus (ayant une connaissance pratique du LaTeX conventionnel) aux développeurs de packages. Ce guide se veut exhaustif dans le sens suivant : il contient des pointeurs vers toutes les sources pertinentes, rassemble des informations qui sont autrement dispersées et ajoute des éléments d'introduction.

Vos commentaires et suggestions d'améliorations sont les bienvenus. Ce document est un travail en cours ; merci pour votre bienveillance et votre patience.

1 Introduction 2
- 1.1 Qu'est-ce que LuaLaTeX ? . . . 2
- 1.2 Passage de LaTeX à LuaLaTeX . . . 3
- 1.3 Une introduction à Lua-dans-TeX . . . 4
- 1.4 Autres choses à savoir . . . 7

2 Packages et pratiques essentiels 7
- 2.1 Niveau utilisateur . . . 7
- 2.2 Niveau développeur . . . 8

3 Autres packages 10
- 3.1 Niveau utilisateur . . . 10
- 3.2 Niveau développeur . . . 11

4 Les formats `luatex` et `lualatex` 12

5 Ce qui marche, ce qui marchotte et ce qui ne marche pas (encore) 13
- 5.1 Ça marche . . . 13
- 5.2 Ça marche partiellement . . . 14
- 5.3 Ça ne marche pas (encore) . . . 15

*Traduit en français par Jérémy Just <jeremy@jejust.fr>.

1. Bien que centré sur LuaLaTeX, il inclut également des informations utiles sur LuaTeX quand il est utilisé avec le format Plain.

1 Introduction

1.1 Qu'est-ce que LuaLaTeX ?

Pour répondre à cette question, nous devons préciser un détail sur le monde TeX que vous pouvez habituellement négliger : la différence entre un *moteur* et un *format*. Un moteur est un programme informatique réel, tandis qu'un format est un ensemble de macros exécutées par un moteur, et généralement préchargé lorsque le moteur est invoqué sous un nom spécifique.

En fait, un format est plus ou moins comme une classe de document ou un paquetage, sauf qu'il est associé à un nom de commande particulier. Imaginez qu'il existe une commande `latex-article` qui ferait la même chose que `latex`, sauf que vous n'auriez pas besoin de dire `\documentclass{article}` au début de votre fichier. De même, dans les distributions actuelles, la commande `pdflatex` est la même que la commande `pdftex`, sauf que vous n'avez pas besoin de mettre les instructions pour charger LaTeX au début de votre fichier source. C'est pratique, et légèrement plus efficace aussi.

Les formats sont une belle invention car ils permettent d'implémenter des commandes puissantes, en utilisant les outils de base fournis par le moteur. Cependant, la puissance du format reste limitée par l'ensemble des outils du moteur, c'est pourquoi les gens ont commencé à développer des moteurs plus puissants afin que d'autres personnes puissent mettre en œuvre des formats (ou des packages) encore plus puissants. Les moteurs les plus connus actuellement (à l'exception du TeX original) sont pdfTeX, XeTeX et LuaTeX.

Pour compliquer encore le tableau, le moteur TeX original ne produisait que des fichiers DVI, alors que ses successeurs peuvent (aussi) produire des fichiers PDF. Chaque commande de votre système correspond à un moteur particulier avec un format particulier et un mode de sortie particulier. Le tableau suivant résume cela : les lignes indiquent le format, les colonnes le moteur, et dans chaque case, la première ligne est la commande pour ce moteur avec ce format en mode DVI, et la seconde en mode PDF.

	TeX	pdfTeX	XeTeX	LuaTeX
Plain	`tex`	`etex`	(aucun)	`dviluatex`
	(aucun)	`pdftex`	`xetex`	`luatex`
LaTeX	(aucun)	`latex`	(aucun)	`dvilualatex`
	(aucun)	`pdflatex`	`xelatex`	`lualatex`

Nous pouvons maintenant répondre à la question posée plus haut : LuaLaTeX est le moteur LuaTeX avec le format LaTeX. Cette réponse n'est pas très satisfaisante si vous ne savez pas ce qu'est LuaTeX (et peut-être LaTeX).

Commençons par ce que vous savez sans doute déjà : dans le monde TeX au sens large, LaTeX est le cadre général dans lequel les documents commencent par `\documentclass`, les packages sont chargés par `\usepackage`, les polices sont sélectionnées de manière intelligente (de sorte que vous puissiez passer en gras tout en préservant l'italique), les pages sont construites à l'aide d'algorithmes compliqués comprenant la prise en charge des en-têtes, des pieds de page, des notes de bas de page, des notes de marge, des flottants, etc. Tout cela ne change pas avec LuaLaTeX, mais de nouveaux packages plus puissants sont disponibles pour améliorer le fonctionnement de certaines parties du système.

Alors, qu'est-ce que LuaTEX ? Version courte : le moteur TEX le plus populaire du moment ! Version longue : c'est le successeur désigné de pdfTeX et il inclut toutes ses fonctionnalités principales : génération directe de fichiers PDF avec support des fonctionnalités PDF avancées et améliorations micro-typographiques des algorithmes typographiques TEX. Les principales nouveautés de LuaTEX sont :

1. Support natif d'Unicode, la norme moderne de classement et d'encodage des caractères, supportant tous les caractères du monde, de l'anglais au chinois traditionnel en passant par l'arabe, y compris de nombreux symboles mathématiques (ou symboles spécifiques d'autres domaines).
2. Intégration de Lua comme langage de script embarqué (voir section 1.3 pour plus de détails).
3. Une multitude de merveilleuses bibliothèques Lua, notamment :
   — `fontloader`, prenant en charge les formats de polices modernes tels que TrueType et OpenType ;
   — `font`, permettant une manipulation avancée des polices à partir du document ;
   — `mplib`, une version embarquée du programme graphique MetaPost ;
   — `callback`, qui permet d'accéder à des parties du moteur TEX qui étaient auparavant inaccessibles au programmeur ;
   — des bibliothèques utilitaires pour la manipulation d'images, de fichiers PDF, etc.

Certaines de ces fonctionnalités, comme la prise en charge d'Unicode, ont un impact direct sur tous les documents, tandis que d'autres fournissent simplement des outils que les auteurs de packages utiliseront pour vous fournir des commandes plus puissantes et autres améliorations.

1.2 Passage de LATEX à LuaLATEX

Comme l'explique la section précédente, LuaLATEX est en grande partie comme LATEX, avec quelques différences, et des packages et outils plus puissants disponibles. Nous présentons ici le minimum absolu que vous devez savoir pour produire un document avec LuaLATEX, tandis que le reste du document fournit plus de détails sur les packages disponibles.

Il n'y a que trois différences :

1. Ne chargez pas inputenc, encodez simplement votre source en UTF-8.
2. Ne chargez pas fontenc ni textcomp, mais chargez fontspec à la place.
3. babel fonctionne avec LuaLATEX mais vous pouvez charger polyglossia à la place.
4. N'utilisez pas de package qui change les polices, mais utilisez les commandes de fontspec à la place.

Ainsi, vous n'avez qu'à vous familiariser avec fontspec, ce qui est facile : sélectionnez la police principale (avec empattement) avec `\setmainfont`, la police sans empattement avec `\setsansfont` et la police à chasse fixe (style machine à écrire) avec `\setmonofont`. L'argument de ces commandes est le petit nom de la police, lisible par un humain, par exemple `Latin Modern Roman` (et non `ec-lmr10`). Pour que les substitutions TEX habituelles fonctionnent (comme `---` pour un tiret cadratin), vous pouvez utiliser `\defaultfontfeatures{Ligatures=TeX}` avant ces commandes.

La bonne nouvelle est que vous pouvez accéder directement à n'importe quelle police de votre système d'exploitation (en plus de celles de votre distribution TEX), y compris les polices TrueType et OpenType, et avoir accès à leurs fonctionnalités les plus avancées. Cela signifie qu'il est désormais facile d'installer n'importe quelle police moderne que vous pouvez télécharger ou acheter auprès d'un éditeur, de les utiliser avec LuaLATEX et de bénéficier de tout leur potentiel.

Passons maintenant aux mauvaises nouvelles : il n'est pas toujours facile d'obtenir une liste de toutes les polices disponibles. Sous Windows avec TEX Live, l'outil de ligne de commande `fc-list` les liste toutes, mais n'est pas très convivial. Sous Mac OS X, l'application *Fontbook* liste les polices de votre système, mais pas celles de votre distribution TEX. Même chose avec `fc-list` sous Linux. Autre mauvaise nouvelle : il n'est pas facile d'accéder à vos anciennes polices de cette manière. Heureusement, progressivement (et rapidement), de plus en plus de polices sont disponibles dans des formats modernes.

Soit dit en passant, le contenu de cette section jusqu'à présent vaut aussi pour XELATEX, c'est-à-dire LATEX sur XETEX. En effet, XETEX partage deux des caractéristiques essentielles de LuaTEX : l'Unicode natif et le support des formats de polices modernes (en revance, il n'a pas les autres caractéristiques de LuaTEX ; mais actuellement, il est considéré comme plus stable). Bien que leurs implémentations concernant les polices de caractères soient très différentes, fontspec parvient à offrir une interface de police pratiquement unifiée pour XELATEX et LuaLATEX.

Ainsi, pour bénéficier des nouvelles fonctionnalités de LuaTEX, vous devez renoncer à un peu de l'ancien monde, à savoir les polices qui ne sont pas disponibles dans un format moderne (ainsi qu'à la liberté d'encoder votre source comme bon vous semble, mais UTF-8 est tellement supérieur aux autres encodages que vous ne perdez quasiment rien au change). Le package luainputenc fournit des solutions de transition qui vous permettent de retrouver certains anciens comportemenbts [2], peut-être au prix de la perte du support réel d'Unicode.

En gros, c'est tout ce que vous devez savoir pour commencer à produire des documents avec LuaLATEX. Je vous recommande de jeter un coup d'oeil au manuel de fontspec et d'essayer de compiler vous-même un petit document en utilisant des polices amusantes. Vous pourrez ensuite parcourir le reste de ce document comme bon vous semble. La section 5 liste toutes les autres différences que je connais entre LATEX conventionnel et LuaLATEX.

1.3 Une introduction à Lua-dans-TEX

Lua est un petit langage, plutôt bien pensé, bien moins surprenant que TEX en tant que langage de programmation, et beaucoup plus facile à apprendre que lui. La référence essentielle est l'excellent livre *Programming in Lua*, dont la première édition est disponible gratuitement en ligne (en anglais). Pour commencer rapidement, je vous recommande de lire les chapitres 1 à 5 et de jeter un coup d'œil à la partie 3. Notez que toutes les bibliothèques mentionnées dans le chapitre 3 sont incluses dans LuaTEX, mais que la bibliothèque os est restreinte pour des raisons de sécurité.

En fonction de vos connaissances en matière de programmation, vous serez peut-être directement intéressé par le reste de la partie 1 et la partie 2, qui présentent des fonctionnalités

2. Bien que son nom suggère qu'il ne s'occupe que des encodages d'entrée, l'implémentation de l'encodage des polices en LATEX implique que ce package est nécessaire (et fonctionne) également pour utiliser les anciennes polices.

plus avancées du langage, mais la partie 4 est inutile dans un contexte de LuaTEX, à moins bien sûr que vous ne vouliez modifier LuaTEX lui-même. Enfin, le manuel de référence de Lua est disponible en ligne et est accompagné d'un index très pratique.

Passons maintenant à l'utilisation de Lua *dans* LuaTEX. La principale façon d'exécuter du code Lua à partir de TEX est la commande `\directlua`, qui prend du code Lua arbitraire comme argument. Inversement, vous pouvez passer des informations de Lua à TEX avec la commande `tex.sprint`[3]. Par exemple,

```
approximation standard de $\pi = \directlua{tex.sprint(math.pi)}$
```

affiche « approximation standard de $\pi = 3.1415926535898$ » dans votre document. Vous voyez comme il est facile d'entremêler du TEX et du Lua ?

En fait, il y a quelques pièges. Regardons d'abord le passage de Lua vers TEX, c'est le plus simple (puisqu'il s'agit davantage de Lua que de TEX). Si vous regardez le manuel de LuaTEX, vous verrez qu'il existe une autre fonction avec un nom plus simple, `tex.print`, pour faire transiter des informations dans ce sens. Elle fonctionne en insérant virtuellement une ligne complète dans votre source TEX, dont le contenu est son argument. Au cas où vous ne le sauriez pas, TEX fait beaucoup de choses néfastes[4] avec les lignes complètes de la source : ignorer les espaces en début et en fin de ligne et ajouter un caractère de fin de ligne. La plupart du temps, vous ne voulez pas que cela se produise, donc je recommande d'utiliser `tex.sprint` qui insère simplement son argument dans la ligne courante, et donne un résultat plus prévisible.

Si vous êtes suffisamment bon TEXnicien pour connaître les catcodes, vous serez heureux d'apprendre que `tex.print` et ses variantes vous donnent un contrôle presque total sur les catcodes utilisés pour tokeniser l'argument, puisque vous pouvez spécifier une table de catcodes comme premier argument. Les tables de catcodes sont présentées à la section 2.7.6 dans le manuel de LuaTEX (dans la version actuelle), vous avez sans doute intérêt à y jeter un œil. Si vous ne connaissez pas les catcodes, passez ce paragraphe.[5]

Regardons maintenant `\directlua`. Pour vous faire une idée de son fonctionnement, imaginez qu'il s'agit d'une commande `\write`, mais qu'elle écrit uniquement dans un fichier virtuel et s'arrange pour que ce fichier soit immédiatement transmis à l'interpréteur Lua. Du côté de Lua, la conséquence est que chaque argument d'une commande `\directlua` a sa propre portée : les variables définies localement dans un argument ne seront pas visibles par le suivant (ce qui est plutôt sain, mais toujours bon à savoir).

Maintenant, le problème majeur est qu'avant d'être transmis à l'interpréteur Lua, l'argument est d'abord lu et tokénisé par TEX, puis entièrement développé et transformé en une chaîne de caractères ordinaire. La lecture par TEX a plusieurs conséquences. L'une d'entre elles est que les fins de lignes sont transformées en espaces, de sorte que l'interprète Lua ne voit qu'une (longue) ligne d'entrée. Comme Lua est un langage à la forme libre, cela n'a généralement pas d'importance, sauf si vous utilisez des commentaires :

3. Dans ce nom, « sprint » signifie « *string print* » (« imprimer une chaîne »), et non « aller très vite » !
4. D'accord, ce sont généralement des actions utiles et bien intentionnées, mais dans ce cas présent, elles sont inattendues, donc je les appelle "néfastes".
5. Arf, trop tard, vous l'avez déjà lu !…

```
\directlua{une_fonction()
  -- un commentaire
  une_autre_fonction()}
```

ne fera pas ce que vous attendez probablement : `une_autre_fonction()` sera considéré comme faisant partie du commentaire (tout est mis sur une seule ligne, ne l'oubliez pas).

Une autre conséquence de la lecture par TEX est que les espaces successives sont fusionnées en une unique espace, et que les commentaires TEX sont éliminés. Voici donc une version correcte de l'exemple précédent, de façon surprenante :

```
\directlua{une_fonction()
  % un commentaire
  une_autre_fonction()}
```

Il convient également de noter que, puisque l'argument se trouve, de fait, à l'intérieur d'un `\write`, il se trouve dans un contexte purement d'expansion. Si vous ne savez pas ce que cela signifie, laissez-moi seulement vous dire que les problèmes d'expansion sont ce qui rend la programmation TEX si difficile et qu'il vaut mieux éviter de développer davantage cette question aujourd'hui.

Je vous prie de m'excuser si les trois derniers paragraphes ont été un peu TEXniques mais j'ai préféré vous prévenir de ces pièges. Pour vous récompenser d'être resté avec moi, voici une astuce de débogage. Collez le code suivant au début de votre document :

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

Ensuite, lorsque vous aurez du mal à comprendre pourquoi un appel particulier à `\directlua` ne fait pas ce que vous attendez, remplacez cette occurrence de la commande par `\directluadebug`, compilez comme d'habitude et regardez dans le fichier `luadebug.lua` ce que l'interpréteur Lua a réellement lu.

Le package luacode fournit des commandes et des environnements qui aident de différentes façons à résoudre certains de ces problèmes. Cependant, dès que le code Lua utilisé n'est plus trivial, il est plus sage d'utiliser un fichier externe contenant uniquement du code Lua définissant des fonctions, puis de le charger et d'appeler ses fonctions depuis le document LuaTEX. Par exemple :

```
\directlua{dofile("mes-functions-lua.lua")}
\newcommand*{\macrogeniale}[2]{%
  \directlua{ma_fonction_geniale("\luatexluaescapestring{#1}", #2)}}
```

L'exemple suppose que `ma_fonction_geniale` est définie dans `mes-fonctions-lua.lua` et prend une chaîne de caractères et un nombre comme arguments. Remarquez que nous prenons soin d'utiliser la primitive `\luatexluaescapestring` sur la chaîne de caractères passée en

argument, afin de protéger tout anti-slash ou guillemets doubles qu'elle pourrait contenir et qui pourraient perturber l'analyseur syntaxique de Lua. [6]

C'est tout pour ce qui concerne Lua dans TEX. Maintenant, si vous vous demandez pourquoi `\luatexluaescapestring` a un nom aussi ridiculement long, lisez la section suivante.

1.4 Autres choses à savoir

Avant toute chose, mentionnons que le manuel de LuaTEX, `luatexref-t.pdf`, est une excellente source d'informations sur LuaTEX et vous voudrez probablement le consulter à un moment ou à un autre (bien qu'il soit un peu aride et technique).

Il est important de noter que les noms des nouvelles primitives de LuaTEX tels que vous les lisez dans le manuel ne sont pas les noms réels que vous pourrez utiliser dans LuaLATEX. En effet, pour éviter des conflits avec les noms de macros existants, toutes les nouvelles primitives ont été préfixées par `\luatex`, à moins qu'elles ne commencent déjà par ce terme. Ainsi, `\luaescapestring` devient `\luatexluaescapetring`, tandis que `\luatexversion` reste `\luatexversion`. Le raisonnement est détaillé dans la section 4.

Oh, et au fait, ai-je mentionné que LuaTEX est en version bêta et que la version 1.0 est attendue pour le printemps 2014 ? Vous pourrez en apprendre davantage en lisant la feuille de route présentée sur le site web de LuaTEX. Des versions bêta stables sont publiées régulièrement et sont incluses dans TEX Live depuis 2008, et dans MikTEX depuis 2.9.

Le support de LuaTEX dans LATEX est tout nouveau, ce qui signifie qu'il peut être bourré de bugs et que les choses peuvent encore changer à tout moment. Il est donc important de garder votre distribution TEX bien à jour pour avoir les corrections de bugs [7] et éviter d'utiliser LuaLATEX pour des documents critiques, au moins pendant un certain temps (pour ne pas rencontrer un nouveau bug au mauvais moment).

De façon générale, ce guide documente les choses telles qu'elles sont au moment précis où il est écrit (ou mis à jour), sans tenir compte des changements éventuellement déjà prévus. Nous espérons que vous mettrez à jour votre distribution dans son ensemble afin d'avoir toujours des versions qui se correspondent entre ce guide et les packages, formats et moteurs qu'il décrit.

2 Packages et pratiques essentiels

Cette section présente les packages que vous voudrez sans doute toujours charger en tant qu'utilisateur, ou que vous devez absolument connaître en tant que développeur.

2.1 Niveau utilisateur

fontspec
Moteurs: XeTEX, LuaTEX. **Formats:** LATEX.
Auteurs: Will Robertson.
Sur le CTAN: `macros/latex/contrib/fontspec/`.

6. Si vous avez déjà utilisé SQL, le concept de protection (ou échappement) des chaînes de caractères devrait vous être familier.

7. For TEX Live, consider using the complementary tlcontrib repository.

URL source: https://github.com/wspr/fontspec/.
Interface conviviale pour la gestion des polices, bien intégrée dans le modèle de sélection des polices de LaTeX. Déjà présenté dans la section précédente.

polyglossia **Moteurs:** XeTeX, LuaTeX. **Formats:** LaTeX.
Auteurs: François Charette & Arthur Reutenauer.
Sur le CTAN: macros/latex/contrib/polyglossia/.
URL source: https://github.com/reutenauer/polyglossia/.
Un remplacement simple et moderne de Babel, travaillant main dans la main avec fontspec.

2.2 Niveau développeur

2.2.1 Conventions de nommage

Du côté de TeX, les séquences de contrôle commençant par `\luatex` sont réservées aux primitives. Il est fortement recommandé de *ne pas* définir de telles séquences de contrôle, afin d'éviter les conflits de noms avec les futures versions de LuaTeX. Si vous souhaitez souligner qu'une macro est spécifique à LuaTeX, nous vous recommandons d'utiliser le préfixe `\lua` (sans le `tex` suivant). Il est possible d'utiliser le préfixe `\luatex@` pour les macros internes, puisque les noms des primitives ne contiennent jamais `@`, mais cela peut prêter à confusion. Dans tous les cas, il est conseillé d'utiliser un préfixe unique pour les macros internes de tous vos packages, donc ces conventions de nommage ne devrait pas vous poser de problème.

En ce qui concerne Lua, veuillez garder l'espace de noms global aussi propre que possible. En d'autres termes, utilisez une table `mypackage` et placez toutes vos fonctions et objets publics dans cette table. Vous devez aussi éviter d'utiliser la fonction `module()` de Lua, qui est obsolète. D'autres stratégies pour la gestion des modules Lua sont discutées dans le chapitre 15 de *Programming in Lua*, et des exemples sont donnés dans `luatexbase-modutils.pdf`. De plus, c'est une saine habitude d'utiliser `local` pour vos variables et fonctions internes. Enfin, pour éviter tout conflit avec les futures versions de LuaTeX, il est nécessaire d'éviter de modifier les espaces de noms des bibliothèques par défaut de LuaTeX.

2.2.2 Détection du moteur et du mode

Plusieurs packages permettent d'identifier le moteur qui traite actuellement le document.

ifluatex **Moteurs:** tous. **Formats:** LaTeX, Plain.
Auteurs: Heiko Oberdiek.
Sur le CTAN: macros/latex/contrib/oberdiek/.
Fournit `\ifluatex` et s'assure que `\luatexversion` est disponible.

iftex **Moteurs:** tous. **Formats:** LaTeX, Plain.
Auteurs: Vafa Khalighi.
Sur le CTAN: macros/latex/contrib/iftex/.
URL source: http://bitbucket.org/vafa/iftex.
Fournit les commandes `\ifPDFTeX`, `\ifXeTeX`, `\ifLuaTeX` et les commandes `\Require` correspondantes.

expl3 **Moteurs:** tous. **Formats:** LaTeX.
Auteurs: The LaTeX3 Project.
Sur le CTAN: macros/latex/contrib/expl3/.
URL source: http://www.latex-project.org/code.html.
Fournit, entre *beaucoup* d'autres choses, \luatex_if_engine:TF, \xetex_if_engine:TF et leurs variantes.

ifpdf **Moteurs:** tous. **Formats:** LaTeX, Plain.
Auteurs: Heiko Oberdiek.
Sur le CTAN: macros/latex/contrib/oberdiek/.
Fournit le commutateur \ifpdf. LuaTeX, comme pdfTeX, peut produire une sortie PDF ou DVI; cette dernière n'est pas très utile avec LuaTeX car elle ne supporte aucune fonctionnalité avancée telle que l'Unicode et les formats de police modernes. Le commutateur \ifpdf est vrai si et seulement si vous exécutez pdfTeX-ou-LuaTeX en mode PDF (notez que cela n'inclut pas XeTeX, dont le support du format PDF est différent).

2.2.3 Ressources de base

luatexbase **Moteurs:** LuaTeX. **Formats:** LaTeX, Plain.
Auteurs: Élie Roux, Manuel Pégourié-Gonnard & Philipp Gesang.
Sur le CTAN: macros/luatex/generic/luatexbase/.
URL source: https://github.com/lualatex/luatexbase.
Les formats Plain et LaTeX fournissent des macros pour gérer les ressources de base de TeX, comme les compteurs ou les registres de boîtes. LuaTeX introduit de nouvelles ressources qui doivent être partagées intelligemment par les packages. Ce package fournit les outils de base pour gérer: les ressources TeX conventionnelles étendues, les tables de catcodes, les attributs, les callbacks, le chargement et l'identification des modules Lua. Il fournit également des outils de base pour gérer quelques problèmes de compatibilité avec les anciennes versions de LuaTeX.
Attention:. Ce package est actuellement en conflit avec le package luatex, puisqu'ils font quasiment la même chose. Les auteurs des deux packages sont bien conscients de cette situation et prévoient de les fusionner d'une manière ou d'une autre dans un avenir proche, bien que le calendrier ne soit pas encore fixé.

luatex **Moteurs:** LuaTeX. **Formats:** LaTeX, Plain.
Auteurs: Heiko Oberdiek.
Sur le CTAN: macros/latex/contrib/oberdiek/.
Voir la description de luatexbase ci-dessus. Ce package fournit les mêmes fonctionnalités de base, à l'exception de la gestion des callbacks et de l'identification des modules Lua.

lualibs **Moteurs:** LuaTeX. **Formats:** Lua.
Auteurs: Élie Roux & Philipp Gesang.
Sur le CTAN: macros/luatex/generic/lualibs/.
URL source: https://github.com/lualatex/lualibs.
Collection de bibliothèques Lua et de compléments aux bibliothèques standards; principalement dérivé des bibliothèques ConTeXt. Si vous avez besoin d'une fonction de base que Lua ne fournit pas, consultez ce package avant de vous lancer dans votre propre développement.

2.2.4 Gestion interne des polices de caractères

Ces packages sont chargés par fontspec pour gérer certaines polices de bas niveau et les problèmes d'encodage. Un utilisateur normal ne devrait utiliser que fontspec, mais un développeur peut avoir besoin de les connaître.

luaotfload **Moteurs:** LuaTeX. **Formats:** LaTeX, Plain.
Auteurs: Élie Roux, Khaled Hosny & Philipp Gesang.
Sur le CTAN: macros/luatex/generic/luaotfload/.
URL source: https://github.com/lualatex/luaotfload.
Chargement de bas niveau des polices OpenType, adapté du code générique de ConTeXt. En gros, il utilise la bibliothèque Lua fontloader et les callbacks correspondants pour implémenter une syntaxe pour la primitive \font très similaire à celle de XeTeX et donner accès aux propriétés correspondantes des polices. Il gère également une base de données de polices pour un accès transparent aux polices du système et de la distribution TeX, soit par nom de famille, soit par nom de fichier, ainsi qu'un système de cache pour un chargement plus rapide des polices.

euenc **Moteurs:** XeTeX, LuaTeX. **Formats:** LaTeX.
Auteurs: Will Robertson, Élie Roux & Khaled Hosny.
Sur le CTAN: macros/latex/contrib/euenc/.
URL source: https://github.com/wspr/euenc.
Implémente les encodages de polices Unicode EUx pour le système fontenc de LaTeX. Actuellement, XeLaTeX utilise EU1 et LuaLaTeX utilise EU2. Inclut les définitions (fichiers fd) pour Latin Modern, la police chargée par défaut par fontspec.

Pour être précis, euenc déclare simplement l'encodage, mais ne fournit pas de définitions pour les macros LICR; ceci est fait en chargeant xunicode avec \UTFencname défini à EU1 ou EU2, ce que fait fontspec. Les encodages réels sont les mêmes, mais il est utile d'avoir des noms distincts pour que différents fichiers fd puissent être utilisés selon le moteur (ce qui est en fait le cas avec Latin Modern).

3 Autres packages

Notez que les packages sont listés sans ordre particulier.

3.1 Niveau utilisateur

luatextra **Moteurs:** LuaTeX. **Formats:** LaTeX.
Auteurs: Élie Roux & Manuel Pégourié-Gonnard.
Sur le CTAN: macros/luatex/latex/luatextra/.
URL source: https://github.com/lualatex/luatextra.
Charge les packages habituels, actuellement fontspec, luacode, metalogo (commandes pour les logos, y compris \LuaTeX et \LuaLaTeX), luatexbase, lualibs, fixltx2e (corrections et améliorations pour le noyau LaTeX).

luacode **Moteurs:** LuaTeX. **Formats:** LaTeX.
Auteurs: Manuel Pégourié-Gonnard.
Sur le CTAN: macros/luatex/latex/luacode/.
URL source: https://github.com/lualatex/luacode.
Fournit des commandes et des macros qui aident à inclure du code Lua dans un source TeX, en particulier pour les caractères spéciaux.

luainputenc **Moteurs:** LuaTeX, XeTeX, pdfTeX. **Formats:** LaTeX.
Auteurs: Élie Roux & Manuel Pégourié-Gonnard.
Sur le CTAN: macros/luatex/latex/luainputenc/.
URL source: https://github.com/lualatex/luainputenc.
Aide à la compilation de documents utilisant des encodages anciens (soit dans le source, soit avec les polices). Déjà présenté dans l'introduction. Sous XeTeX, charge simplement xetex-inputenc; sous pdfTeX, charge l'inputenc standard.

luamplib **Moteurs:** LuaTeX. **Formats:** LaTeX, Plain.
Auteurs: Hans Hagen, Taco Hoewater & Philipp Gesang.
Sur le CTAN: macros/luatex/generic/luamplib/.
URL source: https://github.com/lualatex/luamplib.
Fournit une interface conviviale pour la bibliothèque Lua `mplib`, qui intègre MetaPost dans LuaTeX.

luacolor **Moteurs:** LuaTeX. **Formats:** LaTeX.
Auteurs: Heiko Oberdiek.
Sur le CTAN: macros/latex/contrib/oberdiek/.
Change l'implémentation de bas niveau des couleurs pour utiliser les attributs LuaTeX à la place des *whatsits* ("éléments extraordinaires"). Cela rend l'implémentation plus robuste et corrige des bugs bizarres, notamment un mauvais alignement lorsque `\color` se trouve au début d'un `\vbox`.

3.2 Niveau développeur

pdftexcmds **Moteurs:** LuaTeX, pdfTeX, XeTeX. **Formats:** LaTeX, Plain.
Auteurs: Heiko Oberdiek.
Sur le CTAN: macros/latex/contrib/oberdiek/.
Bien que LuaTeX soit principalement un sur-ensemble de pdfTeX, quelques primitives utilitaires ont été supprimées (celles qui sont en quelque sorte remplacées par Lua) ou renommées. Ce package les fournit avec des noms cohérents entre les différents moteurs, y compris XeTeX qui a récemment implémenté certaines de ces primitives, comme `\strcmp`.

magicnum **Moteurs:** LuaTeX, pdfTeX, XeTeX. **Formats:** LaTeX, Plain.
Auteurs: Heiko Oberdiek.
Sur le CTAN: macros/latex/contrib/oberdiek/.
Fournit un accès hiérarchique aux "nombres magiques" tels que les *catcodes*, les types de groupes, etc., utilisés en interne par TeX et ses héritiers. Sous LuaTeX, une implémentation plus efficace est utilisée et une interface Lua est fournie.

lua-alt-getopt **Moteurs:** `texlua`. **Formats:** ligne de commandes.
Auteurs: Aleksey Cheusov.
Sur le CTAN: `support/lua/lua-alt-getopt`.
URL source: `https://github.com/LuaDist/alt-getopt`.
Parseur d'options de ligne de commande, principalement compatible avec POSIX et GNU getopt, à utiliser dans les scripts Lua en ligne de commande tels que `mkluatexfontdb` ou luaotfload.

4 Les formats `luatex` et `lualatex`

Cette section s'adresse aux développeurs et aux utilisateurs curieux; les utilisateurs normaux peuvent la sauter. Les informations suivantes s'appliquent à TeX Live 2010, et très probablement à MikTeX 2.9 aussi, bien que je n'aie pas vérifié. Les versions antérieures de TeX Live avaient des fonctionnalités légèrement différentes et moins complètes.

Noms des 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 disponibles avec leur nom naturel, mais aussi avec le nom préfixé, afin de faciliter le développement de packages génériques.

Le raisonnement, copié-collé du fichier `lualatexiniconfig.tex` qui implémente ceci pour le format `lualatex`, est le suivant:

1. Tous les packages de macros actuels fonctionnent sans problème avec pdf(e)TeX, donc ces primitives sont conservées telles quelles.
2. D'autres primitives (ne datant pas de TeX82) dans LuaTeX peuvent provoquer des conflits de noms avec des macros existantes dans des packages, en particulier lorsqu'elles utilisent des noms très "naturels" tels que `\outputbox`, `\mathstyle`, etc. La probabilité d'un tel conflit de noms est importante et la situation est assez inconfortable, car plus les documents LaTeX existants fonctionneront dans leur état actuel avec LuaTeX, mieux ce sera.
3. L'équipe LuaTeX ne souhaite pas appliquer une politique de préfixage systématique, mais elle a aimablement fourni un outil permettant d'appliquer des préfixes. Nous avons donc choisi de l'utiliser. Auparavant, nous avions même désactivé les primitives supplémentaires, mais maintenant nous pensons qu'il est préférable de les activer avec un préfixe systématique, afin d'éviter que chaque package (ou chaque utilisateur) les active avec des préfixes variés et incohérents (y compris avec le préfixe vide).
4. Le préfixe `luatex` a été choisi car il est déjà utilisé comme préfixe pour certaines primitives, comme `\luatexversion`: de cette façon, ces primitives ne se retrouvent pas avec un double préfixe (pour plus de détails, voir `tex.enableprimitives` dans le manuel de LuaTeX).
5. La primitive `\directlua` est fournie à la fois avec son nom naturel (permettant une détection facile de LuaTeX) et une version préfixée, `\luatexdirectlua` (par souci de cohérence avec `\luatexlatelua`).
6. Remarques diverses:
   — L'inconvénient évident d'une telle politique de préfixage est que les noms utilisés par LaTeX ou l'outil générique d'écriture de macros ne correspondront pas aux noms utilisés dans le manuel. Nous espérons que cet inconvénient est compensé par le gain en compatibilité ascendante.
   — Toutes les primitives s'occupant de la gymnastique Unicode commencent déjà par `\U`, et leurs noms correspondront peut-être un jour aux noms des primitives XeTeX; donc peut-être que le préfixage n'était ni nécessaire ni souhaitable pour elles. Cependant, nous avons essayé de

rendre la règle de préfixage aussi simple que possible, afin de ne pas aggraver les inconvénients évoqués ci-dessus.
- Peut-être qu'un jour nous penserons qu'il est préférable de fournir toutes les primitives sans préfixe. Si cela se produit, il sera facile d'ajouter les primitives non préfixées dans le format tout en conservant les noms préfixés pour la compatibilité. Ce serait beaucoup plus problématique dans l'autre sens: si l'on réalisait tardivement qu'il vaudrait mieux ne pas fournir de primitives sans préfixe, cela casserait tous les packages spécifiques à LuaTEX qui auront été écrits.

\jobname Le noyau LaTeX (et de nombreux packages) utilise des constructions comme `\input\jobname.aux` dans divers buts. Lorsque `\jobname` contient des espaces, cela ne fait pas ce qu'il faut, puisque l'argument de `\input` se termine au premier espace. Pour contourner ce problème, pdfTEX met automatiquement des guillemets autour de `\jobname` lorsque cela est nécessaire, mais LuaTEX ne le fait pas, pour une raison inconnue. Une solution de contournement presque idéale est incluse dans les formats LuaTEX basés sur LaTeX (par opposition à Plain TEX).

Malheureusement, cette solution ne fonctionne pas, cependant, si LuaTEX est invoqué avec la syntaxe `lualatex '\input name'`, par opposition au plus habituel `lualatex name`. Pour contourner cette limitation de la solution de contournement incluse dans le format, spécifiez un nom de travail explicitement, comme dans `lualatex jobname=name '\input name'`. Ou encore mieux: n'utilisez pas d'espaces dans les noms de vos fichiers TEX.

Pour plus de détails, voir cet ancien fil de discussion et ce fil plus récent sur les listes de diffusion LuaTEX, ainsi que le fichier `lualatexquotejobname.tex` pour l'implémentation de la solution de contournement.

motifs de césure LuaTEX permet le chargement dynamique des motifs de césure. Le support pour ceci dans babel et polyglossia est apparu seulement sur TEX Live 2013, mais devrait bien fonctionner depuis.

La documentation et les détails d'implémentation sont inclus dans `luatex-hyphen.pdf`. Les sources font partie du projet *texhyphen*.

codes Le moteur lui-même ne définit pas les `\catcode`s, `\lccode`s, etc. pour les caractères non-ASCII. Des `\lccode`s corrects, en particulier, sont essentiels pour que la césure fonctionne. Les formats pour LuaTEX incluent maintenant `luatex-unicode-letters.tex`, une version modifiée de `unicode-letters.tex` de la distribution XeTEX, qui se charge de régler ces valeurs en conformité avec la norme Unicode.

Ceci a été ajouté après la sortie de TEX Live 2010, il est donc fortement conseillé de mettre à jour votre installation si vous voulez bénéficier d'une césure correcte pour les textes non-ASCII.

5 Ce qui marche, ce qui marchotte et ce qui ne marche pas (encore)

5.1 Ça marche

Unicode Le LaTeX conventionnel offre un certain niveau de support pour UTF-8 dans les fichiers d'entrée. Cependant, à bas niveau, les caractères non-ASCII ne sont pas atomiques dans ce cas: ils sont constitués de plusieurs morceaux élémentaires (connus sous le nom de *tokens* par les TEXniciens). Par conséquent, certains packages qui analysent le texte caractère par caractère ou effectuent d'autres opérations atomiques sur les caractères (comme la modification de leurs

catcodes) ont souvent des problèmes avec UTF-8 en LaTeX conventionnel. Par exemple, vous ne pouvez pas utiliser n'importe quel caractère non-ASCII pour du verbatim court avec fancyvrb, etc.

La bonne nouvelle est qu'avec LuaLaTeX, certaines des fonctionnalités de ces packages commencent à fonctionner sur des caractères Unicode arbitraires sans avoir besoin de modifier le package. La mauvaise nouvelle est que ce n'est pas toujours vrai. Voir la section suivante pour plus de détails.

5.2 Ça marche partiellement

microtype Le package microtype n'est pas entièrement compatible avec LuaTeX: plus précisément, à partir de la version 2.5 (2013/03/13), la protrusion et l'expansion sont disponibles et activées par défaut en mode PDF, mais le crénage, l'espacement et le réglage de l'approche ne sont pas supportés (voir le tableau 1 dans la section 3.1 de `microtype.pdf`).

D'un autre côté, luaotfload, chargé par fontspec, supporte un grand nombre de fonctionnalités microtypographiques. Le seul problème est donc l'absence d'une interface unifiée.

xunicode La principale fonctionnalité du package xunicode est de s'assurer que les séquences de contrôle habituelles pour les caractères non-ASCII (comme `\'e`) font ce qu'il faut dans un contexte Unicode. Il pourrait *probablement* fonctionner avec LuaTeX, mais ça n'est officiellement vrai que pour XeTeX. Cependant, fontspec utilise une astuce pour le charger de toutes façons. Donc, vous ne pouvez pas le charger explicitement, mais vous n'en avez pas besoin, puisque fontspec s'en est déjà occupé.

encodings Comme mentionné dans la section précédente, certaines choses qui posaient problème avec UTF-8 en LaTeX conventionnel fonctionnent maintenant naturellement, mais pas toujours. Par exemple, avec le package listings de LuaLaTeX, vous ne pouvez utiliser que les caractères inférieurs à 256 (c'est-à-dire les caractères du jeu Latin-1), dans vos listings (mais bien sûr, toute la gamme Unicode est toujours disponible dans le reste de votre document).

métriques Ça, ça fonctionne en soi, mais ça ne fonctionne pas exactement de la même manière que pdfTeX ou XeTeX: vous pourrez observer des différences mineures dans la mise en page et les coupures de mots de votre texte.

Elles peuvent être dues à des variations entre deux versions de la même police utilisée par les différents moteurs, à des ajustements apportés aux algorithmes de césure, de ligature ou de crénage (par exemple, le premier mot d'un paragraphe, ainsi que les mots contenant des polices différentes, peuvent désormais être coupés en fin de ligne), ou à des différences dans les motifs de césure utilisés (les motifs utilisés par pdfTeX sont fondamentalement figés, mais LuaTeX et XeTeX utilisent des versions plus récentes pour certaines langues) pour cette langue.

Si vous observez une différence majeure entre pdfLaTeX et LuaLaTeX avec les mêmes polices, il n'est pas du tout improbable qu'un bug dans LuaTeX [8] ou dans la police soit impliqué. Comme d'habitude, assurez-vous que votre distribution est à jour avant de signaler un tel problème.

8. Par exemple, LuaTeX 0.60 avait un bug qui empêchait toute césure après une ligature – jusqu'à la fin du paragraphe.

babel Fonctionne pour l'essentiel sans problème pour les langues latines. Pour les autres langues, les résultats peuvent varier. Même pour les langues latines, des problèmes liés à l'encodage peuvent survenir.

polyglossia Un package plus moderne que babel, mais moins complet, pour le support multilingue, polyglossia, est également disponible et devrait être préféré, bien qu'il ne supporte pas encore toutes les fonctionnalités de babel.

5.3 Ça ne marche pas (encore)

anciens encodages Les packages jouant avec les encodages d'entrée (fichiers sources) ou de sortie (polices) sont très susceptibles de ne pas fonctionner avec LuaTEX. Cela inclut inputenc, fontenc, textcomp, et probablement la plupart des packages de polices classiques tels que mathptmx ou fourier. La bonne nouvelle est qu'Unicode est un moyen plus puissant de gérer les problèmes d'encodage que les anciens packages essayaient de résoudre, donc vous n'avez probablement plus besoin de ces anciens packages. Cependant, tout n'a pas encore été porté dans le nouveau monde merveilleux d'Unicode, et il se peut que vous ayez un choix plus limité (ou simplement différent) pendant un certain temps (ceci est particulièrement vrai pour les polices).

espaces Les espaces dans les noms de fichiers ne sont pas vraiment bien supportés dans le monde TEX en général. Cela ne s'améliore pas non plus avec LuaTEX. De plus, pour des raisons délicates, les choses peuvent être pires si vous avez des espaces dans le nom de votre fichier TEX principal *et* que vous n'invoquez pas LuaTEX de la manière habituelle. Si vous l'invoquez de la manière habituelle, tout devrait fonctionner, et je ne vous dirai pas à quoi ressemble l'invocation inhabituelle. Sinon, lisez le paragraphe sur jobname dans la section 4 pour une solution de contournement et des détails techniques. Mais le mieux, c'est de ne pas utiliser d'espaces dans les noms de vos fichiers TEX.