

# Un guide pour LuaLaTeX

Par Manuel Pégourié-Gonnard - [ram-0000](#) (traducteur)

Date de publication : 5 mai 2013

Dernière mise à jour : 24 juin 2014

Ce document est une carte ou un guide touristique pour le nouveau monde **LuaLaTeX** (1). Le public visé va du débutant complet (avec une connaissance pratique de LaTeX classique) pour englober les développeurs. Ce guide est destiné à être complet dans le sens où il contient des pointeurs vers toutes les sources pertinentes, synthétise les informations éparses et ajoute une introduction.

Les commentaires et les suggestions d'amélioration sont les bienvenus. Ce document est toujours en cours d'évolution, merci pour votre compréhension et votre patience. Après votre lecture, n'hésitez pas. **Commentez**

1 - Introduction.....	3
1-1 - Qu'est-ce que LuaLaTeX au juste .....	3
1-2 - Migrer de LaTeX à LuaLaTeX.....	4
1-3 - Un début, Lua dans Tex.....	5
1-4 - Autre chose que vous devriez savoir.....	6
2 - Extensions et pratiques essentielles.....	7
2-1 - Niveau utilisateur.....	7
2-1-1 - fontspec.....	7
2-1-2 - polyglossia.....	7
2-2 - Niveau développeur.....	7
2-2-1 - Conventions de nommage.....	7
2-2-2 - Moteur et mode de détection.....	8
2-2-2-1 - ifluatex.....	8
2-2-2-2 - iftex.....	8
2-2-2-3 - expl3.....	8
2-2-2-4 - ifpdf.....	8
2-2-3 - Ressources de base.....	9
2-2-3-1 - luatexbase.....	9
2-2-3-2 - luatex.....	9
2-2-3-3 - lualibs.....	9
2-2-4 - Fontes internes.....	9
2-2-4-1 - luaotfload.....	10
2-2-4-2 - euenc.....	10
3 - Autres extensions.....	10
3-1 - Niveau utilisateur.....	10
3-1-1 - luatextra.....	10
3-1-2 - luacode.....	11
3-1-3 - luainputenc.....	11
3-1-4 - luamplib.....	11
3-1-5 - luacolor.....	11
3-2 - Niveau développeur.....	12
3-2-1 - pdftexcmds.....	12
3-2-2 - magicnum.....	12
3-2-3 - lua-alt-getopt.....	12
4 - Les formats luatex et lualatex.....	12
4-1 - Les noms des primitives.....	12
4-2 - \jobname.....	13
4-3 - La césure.....	13
4-4 - Les codes.....	14
5 - Ce qui fonctionne bien, partiellement ou pas encore.....	14
5-1 - Ce qui fonctionne bien.....	14
5-1-1 - Unicode.....	14
5-2 - Ce qui fonctionne partiellement.....	14
5-2-1 - microtype.....	14
5-2-2 - xunicode.....	14
5-2-3 - Les encodages.....	14
5-2-4 - metrics.....	15
5-2-5 - babel.....	15
5-2-6 - polyglossia.....	15
5-3 - Ce qui ne fonctionne pas (encore).....	15
5-3-1 - Les anciens encodages.....	15
5-3-2 - Les espaces.....	15
6 - Remerciements.....	16

## 1 - Introduction

### 1-1 - Qu'est-ce que LuaLaTeX au juste

Pour répondre à cette question, nous devons mentionner quelques détails sur le monde **TeX** que vous pouvez généralement ignorer : la différence entre un moteur et un format. Un moteur est un programme de l'ordinateur réel, tandis que le format est un ensemble de macros exécutées par ce moteur. Ce format est généralement préchargé lorsque le moteur est invoqué avec un nom particulier.

En fait, un format est plus ou moins comme une classe de document ou une extension, sauf qu'il est associé à un nom de commande particulière. Imaginez qu'il y ait 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 aussi beaucoup plus efficace.

Les formats sont bien parce qu'ils mettent en œuvre des commandes puissantes en utilisant les outils de base fournis par le moteur. Cependant, la puissance des formats est parfois limitée par les outils mis à disposition par le moteur. C'est ainsi que des personnes ont commencé à développer des moteurs plus puissants, permettant à d'autres personnes de mettre en œuvre des formats encore plus puissants (ou extensions). Les plus célèbres moteurs maintenant (à l'exception du **TeX** origine) sont **pdfTeX**, **XeTeX** et **LuaTeX**.

Pour compliquer un peu plus la situation, le moteur de **TeX** original produit uniquement des fichiers DVI, alors que ses successeurs peuvent (aussi) produire des fichiers PDF. Chaque commande, dans votre système, correspond à un moteur particulier avec un format et un mode de sortie particuliers. Le tableau suivant résume cela: les lignes sont indexées par le format, les colonnes par le moteur. Dans chaque cellule, la première ligne est la commande de ce moteur avec le format en mode DVI et la deuxième ligne pour le mode PDF.

	TeX	pdfTeX	XeTeX	LuaTeX
Plain	TeX Aucun	eTeX pdfTeX	Aucun xetex	dviluatex luatex
LaTeX	Aucun Aucun	LaTeX pdflatex	Aucun xelatex	dvilualatex lualatex

Nous pouvons maintenant répondre à la question du titre : **LuaLaTeX** est le moteur de **LuaTeX** avec le format **LaTeX**. Eh bien, cette réponse n'est pas très satisfaisante si vous ne savez pas ce que sont **LuaTeX** et **LaTeX**.

Comme vous le savez probablement, **LaTeX** est le cadre général dans lequel les documents commencent par `\documentclass`, les extensions sont chargées avec `\usepackage`, les polices sont sélectionnées de façon intelligente (de sorte que vous pouvez passer en gras tout en préservant l'italique), les pages sont construites avec des algorithmes compliqués y compris le support pour les en-têtes, pieds de page, les notes, les notes en marge, les notes flottantes, etc. Cela ne change pas avec **LuaLaTeX**, mais des extensions nouvelles et plus puissantes sont disponibles pour que le système fonctionne d'une meilleure façon.

Alors, qu'est-ce que **LuaTeX** ? Version courte : le meilleur moteur **TeX** du moment ! Version longue : Il est le successeur désigné de **pdfTeX** et comprend toutes ses fonctions de base : la génération directe de fichiers PDF avec le support de fonctionnalités PDF avancées et des améliorations microtypographiques par rapport aux algorithmes typographiques de **TeX**. Les principales nouvelles fonctionnalités de **LuaTeX** sont indiquées ci-dessous.

- 1 La prise en charge native de l'Unicode, la norme moderne pour la classification et l'encodage des caractères supportant tous ceux du monde, de l'anglais au chinois traditionnel en passant par l'arabe, y compris beaucoup de symboles mathématiques (et autres symboles spécialisés).
- 2 L'inclusion de **Lua** comme langage de script intégré (voir section 1.3 pour plus de détails).
- 3 Un grand nombre de bibliothèques **Lua**, avec entre autres :

- 1 **fontloader**, pour le support du format de polices modernes telles que *TrueType* et *OpenType* ;
- 2 **font**, permettant la manipulation avancée des polices de l'intérieur du document ;
- 3 **mplib**, une version intégrée du programme graphique MetaPost ;
- 4 **callback**, pour fournir des fonctions de rappel (hook ou callback) dans le moteur **TeX** qui étaient auparavant inaccessibles pour le développeur ;
- 5 des bibliothèques d'utilitaires pour la manipulation d'images, de fichiers PDF, etc.

Certaines de ces fonctionnalités, telles que le support Unicode, ont un impact direct sur tous les documents, tandis que d'autres se contentent de fournir des outils que les auteurs d'extensions vont utiliser pour vous fournir des commandes plus puissantes et d'autres améliorations.

## 1-2 - Migrer de LaTeX à LuaLaTeX

Comme expliqué dans le paragraphe précédent, **LuaLaTeX** est la plupart du temps identique à **LaTeX**, avec toutefois quelques différences, des extensions et des outils disponibles plus puissants. Nous présentons ici le strict minimum de ce que vous devez savoir pour produire un document avec **LuaLaTeX**, tandis que le reste du document fournit plus de détails sur les extensions disponibles.

Il y a seulement quatre différences.

- 1 Ne chargez pas *inputenc*, il suffit juste d'encoder votre source en UTF-8.
- 2 Ne chargez ni *fontenc*, ni *textcomp*, mais chargez *fontspec*.
- 3 *Babel* fonctionne avec **LuaLaTeX**, mais vous pouvez aussi charger *polyglossia* à sa place.
- 4 N'utilisez pas les extensions qui changent les polices, mais utilisez les commandes *fontspec* à leur place.

Donc, vous avez seulement besoin de vous familiariser avec *fontspec*, ce qui est facile : sélectionnez la police principale (serif) avec `\setmainfont`, la fonte de caractères sans empattement avec `\setsansfont` et la police à espacement fixe (machine à écrire) avec `\setmonofont`. L'argument de ces commandes est le nom de la police, par exemple **Latin Modern Roman** plutôt que **ec-lmr10**. Vous voudrez aussi probablement utiliser la commande `\defaultfontfeatures = TeX` avant ces commandes pour conserver les substitutions habituelles **TeX** (comme --- pour un tiret long).

La bonne nouvelle est que vous pouvez accéder directement à n'importe quelle police de votre système d'exploitation (en plus de celles de la distribution **TeX**), y compris des polices *TrueType* et *OpenType*, et avoir accès à leurs fonctionnalités les plus avancées. Cela signifie aussi qu'il est maintenant possible d'utiliser, avec **LuaLaTeX**, n'importe quelle police moderne que vous pourriez télécharger ou acheter chez un éditeur, et ainsi bénéficier de leur plein potentiel.

Maintenant, pour les mauvaises nouvelles : il n'est pas toujours facile d'obtenir la liste de toutes les polices disponibles. Sous Windows, avec *TeX Live*, l'outil en ligne de commande `fc-list` les énumère toutes, mais n'est pas très convivial. Sous Mac OS X, l'application *Fontbook* répertorie les polices de votre système, mais pas celles de votre distribution **TeX**. C'est pareil avec `fc-list` sur Linux. Encore pire, il n'est pas facile d'accéder à vos anciennes polices de cette façon. Heureusement, de plus en plus de fontes modernes sont disponibles tous les jours (enfin tous les mois ou toutes les années si on ne compte que les bonnes polices).

En passant, mentionnons que le contenu de cette section est aussi valable pour **XeLaTeX**, c'est-à-dire **LaTeX** sur **XeTeX**. En effet, **XeTeX** partage deux caractéristiques essentielles avec **LuaTeX** : Unicode et le support des formats de polices modernes (bien qu'il ne possède pas les autres caractéristiques de **LuaTeX**, il est maintenant plus stable). Bien que la mise en œuvre concernant les polices soit très différente, *fontspec* parvient à offrir une interface unifiée pour tous les deux, **XeLaTeX** et **LuaLaTeX**.

Ainsi, pour bénéficier des nouvelles fonctionnalités de **LuaTeX**, vous devez supprimer quelques parties, à savoir les polices qui ne sont pas disponibles dans un format moderne (et aussi la liberté d'encoder votre texte comme vous

voulez, mais UTF-8 est tellement supérieur que cela ne compte guère). L'extension *luainputenc* fournit différents compromis qui vous permettent de retrouver ces parties (2) probablement au prix de la perte du soutien Unicode.

C'est tout ce que vous avez besoin de savoir pour commencer à produire des documents avec **LuaLaTeX**. Je vous recommande de jeter un coup d'œil au manuel de *fontspec* et d'essayer de compiler un petit document en utilisant des polices amusantes. Vous pouvez ensuite survoler le reste de ce document comme vous le souhaitez. La section 5 répertorie toutes les autres différences entre **LaTeX** et **LuaLaTeX** dont je suis au courant.

### 1-3 - Un début, Lua dans Tex

**Lua** est un petit langage sympathique, évidemment moins surprenant et beaucoup plus facile à apprendre que **TeX** comme langage de programmation. La référence essentielle est le livre *Programming in Lua* qui est très facile à lire, dont la première édition est gratuitement **disponible en ligne**. Pour un démarrage rapide, je vous recommande de lire les chapitres 1 à 5 et de survoler la partie 3. Notez que toutes les bibliothèques mentionnées dans le chapitre 3 sont incluses dans **LuaTeX**, mais la bibliothèque OS est limitée pour des raisons de sécurité. En fonction de votre culture de la programmation, vous pouvez être directement intéressé par la fin de la partie 1 et la partie 2 qui présentent des fonctionnalités plus avancées de la langue. Par contre, la partie 4 est inutile dans un contexte de **LuaTeX**, à moins bien sûr que vous vouliez pirater **LuaTeX** lui-même. Enfin, le manuel de référence **Lua** est **disponible en ligne** et est livré avec un index pratique.

Maintenant, passons à **Lua** dans **LuaTeX**. Le principal moyen pour exécuter du code **Lua** dans **TeX** est d'utiliser la commande `\directlua` qui prend comme argument du code arbitraire **Lua**. Inversement, vous pouvez aussi passer des informations de **Lua** vers **TeX** avec `tex.sprint` (3). Par exemple :

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

affiche « the standard approximation  $\pi = 3.1415926535898$  » dans votre document. Vous voyez comment il est simple de mixer **Lua** et **TeX** ?

En fait, il y a quelques pièges. Regardons tout d'abord le passage de **Lua** vers **TeX**, c'est le plus simple (car c'est plus **Lua** que **TeX**). Si vous regardez le manuel de **LuaTeX**, vous verrez qu'il existe une autre fonction avec un nom plus simple, **tex.print**, pour transmettre des informations de cette façon. Il fonctionne en insérant littéralement une ligne de texte dans votre source **TeX** dont le contenu est l'argument passé. Au cas où vous ne le sauriez pas, **TeX** fait de drôles de choses (4) avec les lignes de source : il ignore les espaces au début et à la fin de la ligne et il ajoute un caractère de fin de ligne. La plupart du temps, vous ne voulez pas que cela arrive, nous vous recommandons alors d'utiliser **tex.sprint** qui insère littéralement son argument dans la ligne actuelle, il en est beaucoup plus prévisible.

Si vous êtes assez *TeXnicien* pour comprendre les *catcodes*, vous serez heureux de savoir que **tex.print** et ses variantes vous donnent presque les pleins pouvoirs sur les catcodes, puisque vous pouvez spécifier un tableau de catcodes en tant que premier argument. Vous aurez probablement envie de tout savoir sur ces tables de catcodes (paragraphe 2.7.6 dans le manuel de **LuaTeX**) avant que vous soyez pleinement heureux. Si vous ne connaissez pas les catcodes, sautez ce paragraphe (5).

Regardons `\directlua` maintenant. Pour avoir une idée sur la façon dont cela fonctionne, imaginez que c'est une commande `\write`, mais qui écrit dans un fichier virtuel immédiatement mis à disposition et envoyé à l'interpréteur **Lua**. Côté **Lua**, la conséquence est que chaque argument d'une commande `\directlua` a son propre champ d'application : les variables locales de l'un ne seront pas visibles par l'autre (c'est plutôt sain, mais toujours bon à savoir).

Maintenant, le gros problème c'est qu'avant d'être envoyé à l'interpréteur **Lua**, l'argument est d'abord lu et analysé par **TeX**, puis complètement transformé en une chaîne simple. Cette lecture par **TeX** a plusieurs conséquences. L'une d'elles est que les fins de lignes sont transformées en espaces, donc **Lua** ne voit qu'une seule (longue) ligne d'entrée. **Lua** est un langage de forme libre et cela n'a pas d'importance en général, mais il si vous utilisez commentaires suivants :

```
\directlua{a_function()  
-- a comment
```

```
another_function() }
```

Cela ne fera pas ce que vous attendez : `another_function()` sera considérée comme faisant partie du commentaire (c'est une seule ligne, rappelez-vous).

Une autre conséquence de la lecture par **TeX** est que les espaces successifs sont fusionnés en un seul espace et les commentaires **TeX** sont ignorés. Voici une version correcte de l'exemple précédent :

```
\directlua{a_function()
% a comment
another_function() }
```

Il faut aussi remarquer que, puisque l'argument est fondamentalement à l'intérieur d'un `\write`, c'est dans un contexte d'expansion uniquement. Si vous ne savez pas ce que cela signifie, laissez-moi vous dire que les questions d'expansion sont celles qui rendent la programmation **TeX** si difficile, pour éviter de s'étendre sur ce sujet.

Je suis désolé si les trois derniers paragraphes sont TeXniques, mais je pensais qu'il était nécessaire que vous le sachiez. Pour vous récompenser de rester avec moi, voici une astuce de débogage. Placez le code suivant au début de votre document :

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

Puis, quand vous aurez passé suffisamment de temps à essayer de comprendre pourquoi un appel à `\directlua` ne fait pas ce que vous attendez, remplacez cette commande par `\directlua\debug`, compilez comme d'habitude et regardez le fichier `luadebug.lua` qui contient ce qui est réellement interprété par **Lua**.

L'extension *luacode* fournit des commandes et des environnements qui aident à des degrés divers, avec certains de ces problèmes. Cependant, pour des morceaux de code triviaux de **Lua**, il est plus courant d'utiliser un fichier externe contenant uniquement du code **Lua** définissant ces fonctions, puis le charger et utiliser ses fonctions. Par exemple :

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

L'exemple suppose que `my_great_function` est défini dans `my-lua-functions.lua` et prend une chaîne de caractères et un nombre comme arguments. Notez que nous utilisons les primitives `\luatexluaescapestring` avec l'argument de la chaîne de caractères pour échapper à tout caractère backslash ou guillemet qu'il pourrait contenir et qui pourrait perturber l'analyseur **Lua** (6) .

C'est tout ce qui concerne l'insertion de code **Lua** dans **TeX**. Si vous vous demandez pourquoi `\luatexluaescapestring` a un nom long et stupide, vous pouvez lire la section suivante.

## 1-4 - Autre chose que vous devriez savoir

Juste au cas où cela ne semble pas évident, le manuel de **LuaTeX**, `luatexref-t.pdf`, est une grande source d'informations sur **LuaTeX** et vous aurez probablement besoin de le consulter à un moment donné (même s'il est un peu aride et technique). Il est important de savoir que les noms des nouvelles primitives de **LuaTeX** que vous trouverez dans le manuel ne sont pas les noms réels que vous pourrez utiliser dans **LuaLaTeX**. Pour éviter des conflits avec des noms de macros existantes, toutes les nouvelles primitives ont été préfixées avec `\luaTeX` à moins qu'elles ne commencent déjà comme cela, donc `\luaescapestring` devient `\luatexluaescapestring` alors que `\luatexversion` reste `\luatexversion`. La raison est détaillée dans la section 4.



Oh, en passant, je vous ai dit que **LuaTeX** est en version bêta et que la version 1.0 est attendue pour le printemps 2014 ? Vous pouvez en apprendre plus sur la page de la feuille de route du [site LuaTeX](#). Des versions bêta stables sont régulièrement publiées et sont incluses dans *TeX Live* depuis 2008 et depuis *MikTeX* 2.9.

Sans surprise, le soutien à **LuaTeX** pour **LaTeX** est flambant neuf, ce qui signifie qu'il peut y avoir pleins de bogues et que les choses peuvent changer à tout moment. Vous pouvez garder votre distribution **TeX** à jour (7) et également éviter d'utiliser **LuaLaTeX** dans des documents critiques, au moins pour un certain temps.

En règle générale, ce guide documente les choses telles qu'elles sont au moment où il est écrit ou mis à jour, sans garder trace des changements. J'espère que vous allez mettre à jour votre distribution dans son ensemble de sorte que vous obteniez toujours les versions de ce guide, les extensions, les formats et le moteur correspondant.

## 2 - Extensions et pratiques essentielles

Cette section présente les extensions que vous voudrez probablement toujours charger en tant qu'utilisateur, ou celles que vous devriez absolument connaître en tant que développeur.

### 2-1 - Niveau utilisateur

#### 2-1-1 - fontspec

Moteurs	XeTeX, LuaTeX
Formats	LaTeX
Auteurs	Will Robertson
Localisation CTAN	<a href="#">macros/latex/contrib/fontspec/</a>
URL de développement	<a href="https://github.com/wspr/fontspec/">https://github.com/wspr/fontspec/</a>

Interface conviviale de gestion de fontes, bien intégrée dans le schéma de sélection de police de **LaTeX**. Déjà présentée dans la section précédente.

#### 2-1-2 - polyglossia

Moteurs	XeTeX, LuaTeX
Formats	LaTeX
Auteurs	François Charette & Arthur Reutenauer
Localisation CTAN	<a href="#">macros/latex/contrib/polyglossia/</a>
URL de développement	<a href="https://github.com/reutenauer/polyglossia/">https://github.com/reutenauer/polyglossia/</a>

Un remplaçant simple et moderne de *babel*, travaillant en synergie avec *fontspec*.

### 2-2 - Niveau développeur

#### 2-2-1 - Conventions de nommage

Au niveau de **TeX**, des séquences de contrôle commençant par `\luaTeX` sont réservées pour les primitives. Il vous est fortement recommandé de ne pas définir de séquence de contrôle, cela pour éviter les conflits de noms avec les futures versions de **LuaTeX**. Si vous insistez pour indiquer qu'une macro est spécifique à **LuaTeX**, nous vous recommandons d'utiliser le préfixe `\lua` (non suivi de `tex`) à la place. Il est correct d'utiliser le préfixe `\luaTeX@` pour les macros internes puisque les noms de primitives ne contiennent jamais `@`, mais cela peut être source de confusion. De plus, vous utilisez déjà un préfixe unique pour les macros internes dans l'ensemble de vos extensions, n'est-ce pas ?

Au niveau de **Lua**, s'il vous plaît, gardez l'espace global de nommage aussi propre que possible. C'est-à-dire utilisez une table `mypackage` et mettez toutes les fonctions publiques et les objets dans ce tableau. Vous devriez aussi éviter d'utiliser la fonction obsolète `module()` de **Lua**. D'autres stratégies de gestion du module **Lua** sont discutées dans le **chapitre 15 de la programmation en Lua** et des exemples sont donnés dans `luatexbase-modutils.pdf`. En outre, il est bon et nécessaire d'utiliser `local` pour les variables et fonctions internes. Enfin, pour éviter des conflits avec les futures versions de **LuaTeX**, il est impératif d'éviter de modifier l'espace de nommage des bibliothèques par défaut de **LuaTeX**.

## 2-2-2 - Moteur et mode de détection

Différentes extensions permettent de détecter le moteur utilisé pour le traitement du document en cours.

### 2-2-2-1 - ifluatex

Moteurs	all
Formats	LaTeX, Plain
Auteurs	Heiko Oberdiek
Localisation CTAN	<a href="#">macros/latex/contrib/oberdiek/</a>

Fournit `\ifluatex` et s'assure que `\luatexversion` est disponible.

### 2-2-2-2 - iftex

Moteurs	all
Formats	LateX, Plain
Auteurs	Vafa Khalighi
Localisation CTAN	<a href="#">macros/latex/contrib/iftex/</a>

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

### 2-2-2-3 - expl3

Moteurs	all
Formats	LaTeX
Auteurs	the LaTeX3 Project
Localisation CTAN	<a href="#">pkg/expl3</a>
URL de développement	<a href="http://www.latex-project.org/code.html">http://www.latex-project.org/code.html</a>

Entre autres choses, fournit `\luatex_if_engine:TF`, `\xetex_if_engine:TF` et leurs variantes.

### 2-2-2-4 - ifpdf

Moteurs	all
Formats	LaTeX, Plain
Auteurs	Heiko Oberdiek
Localisation CTAN	<a href="#">macros/latex/contrib/oberdiek/</a>

Fournit le commutateur `\ifpdf`. **LuaTeX**, comme **pdfTeX**, peut générer une sortie PDF ou DVI. Ce dernier n'est pas très utile avec **LuaTeX**, car il ne prend en charge aucune des fonctions avancées comme Unicode et les formats de polices modernes. Le commutateur `\ifpdf` est vrai si et seulement si vous utilisez `pdfTeX-or-LuaTeX` en mode PDF (notez que cela n'inclut pas **XeTeX** dont le soutien PDF est différent).




## 2-2-3 - Ressources de base

### 2-2-3-1 - luatexbase

Moteurs	LuaTeX
Formats	LaTeX, Plain
Auteurs	Élie Roux, Manuel Pégourié-Gonnard & Philipp Gesang
Localisation CTAN	<a href="#">macros/luatex/generic/luatexbase/</a>
URL de développement	<a href="https://github.com/lualatex/luatexbase">https://github.com/lualatex/luatexbase</a>

Les formats **Plain** et **LaTeX** fournissent des macros pour gérer les ressources de base de **TeX**, comme le registre de compteurs ou de boîtes. **LuaTeX** introduit de nouvelles ressources qui peuvent être utilisées par d'autres extensions. Cette extension fournit les outils de base pour gérer : les ressources étendues classiques de **TeX**, les tables de catcode, les attributs, les fonctions de rappels, le chargement et l'identification des modules **Lua**. Il fournit également des outils de base pour gérer quelques problèmes de compatibilité avec les versions plus anciennes de **LuaTeX**.

 *Attention, cette extension est actuellement en conflit avec l'extension de **LuaTeX** puisque toutes les deux font presque les mêmes choses. Les auteurs respectifs de ces extensions sont bien conscients de cette situation et envisagent, dans un proche avenir, de fusionner les deux paquets. Cela dit, la chronologie n'est pas encore claire.*

### 2-2-3-2 - luatex

Moteurs	LuaTeX
Formats	LaTeX, Plain
Auteurs	Heiko Oberdiek
Localisation CTAN	<a href="#">macros/latex/contrib/oberdiek/</a>

Voir la description de *luatexbase* ci-dessus. Cette extension fournit les mêmes fonctionnalités de base à l'exception de la gestion des fonctions de rappel et l'identification des modules **Lua**.

### 2-2-3-3 - lualibs

Moteurs	LuaTeX
Formats	Lua
Auteurs	Élie Roux & Philipp Gesang
Localisation CTAN	<a href="#">macros/luatex/generic/lualibs/</a>
URL de développement	<a href="https://github.com/lualatex/lualibs">https://github.com/lualatex/lualibs</a>

Une collection de bibliothèques **Lua** et d'ajouts à la bibliothèque standard. La plupart provenant des bibliothèques de *ConTeXt*. Si vous avez besoin d'une fonction de base que **Lua** ne possède pas, vérifiez cette extension avant de débuter votre propre implémentation.

## 2-2-4 - Fontes internes

Ces extensions sont chargées par *fontspec* pour gérer certaines fontes de caractères de bas-niveau et les problèmes d'encodage. Un utilisateur normal ne devrait utiliser que *fontspec*, mais un développeur pourrait en avoir besoin.

## 2-2-4-1 - luaotfload

Moteurs	LuaTeX
Formats	LaTeX, Plain
Auteurs	Élie Roux, Khaled Hosny & Philipp Gesang
Localisation CTAN	<a href="#">macros/luatex/generic/luaotfload/</a>
URL de développement	<a href="https://github.com/lualatex/luaotfload">https://github.com/lualatex/luaotfload</a>

Chargement bas-niveau des fontes *OpenType*, *luaotfload* est adapté du sous-ensemble générique de *ConTeXt*. Fondamentalement, il utilise la bibliothèque **Lua fontloader** et les fonctions de rappels nécessaires pour mettre en œuvre une syntaxe pour la primitive **\font**, très similaire à celle de **XeTeX**, et met en œuvre les fonctions des fontes correspondantes. Il gère également une base de données de fontes pour un accès transparent aux polices système et **TeX** soit par nom de famille ou par nom de fichier, ainsi qu'un cache de police pour un chargement plus rapide.

## 2-2-4-2 - euenc

Moteurs	XeTeX, LuaTeX
Formats	LaTeX
Auteurs	Will Robertson, Élie Roux & Khaled Hosny
Localisation CTAN	<a href="#">macros/latex/contrib/euenc/</a>
URL de développement	<a href="https://github.com/wspr/euenc">https://github.com/wspr/euenc</a>

Met en œuvre l'encodage des fontes Unicode Eux pour le système de fontenc de **LaTeX**. Actuellement, **XeLaTeX** utilise **EU1** et LuaLaTeX utilise **EU2**. Il comprend les définitions (fichiers fd) pour la fonte *Latin Modern*, la fonte par défaut chargée par *fontspec*.

Pour être précis, *euenc* déclare simplement l'encodage, mais ne prévoit pas de définitions pour les macros LICR. Cela se fait par chargement *xunicode* avec `\UTFencname` défini à **EU1** ou **UE2**, ce que fait *fontspec*. Les encodages sont les mêmes, mais il est utile d'avoir des noms distincts afin que les différents fichiers fd puissent être utilisés selon le moteur (ce qui est actuellement fait avec *Latin Modern*).

## 3 - Autres extensions

Notez que les extensions ne sont pas énumérées dans un ordre particulier.

### 3-1 - Niveau utilisateur

#### 3-1-1 - luatextra

Moteurs	LuaTeX
Formats	LaTeX
Auteurs	Élie Roux & Manuel Pégourié-Gonnard
Localisation CTAN	<a href="#">macros/luatex/latex/luatextra/</a>
URL de développement	<a href="https://github.com/lualatex/luatextra">https://github.com/lualatex/luatextra</a>

Charge les extensions habituelles, actuellement *fontspec*, *luacode*, *metalogo* (commandes pour les logos pour `\LuaTeX` et `\LuaLaTeX`), *luatexbase*, *lualibs* et *fixltx2e* (corrections et améliorations pour le noyau de **LaTeX**).

### 3-1-2 - luacode

Moteurs	LuaTeX
Formats	LaTeX
Auteurs	Manuel Pégourié-Gonnard
Localisation CTAN	<a href="#">macros/luatex/latex/luacode/</a>
URL de développement	<a href="https://github.com/lualatex/luacode">https://github.com/lualatex/luacode</a>

Fournit des commandes et des macros qui permettent d'inclure du code **Lua** dans une source **TeX**, en particulier les caractères spéciaux.

### 3-1-3 - luainputenc

Moteurs	LuaTeX, XeTeX, pdfTeX
Formats	LaTeX
Auteurs	Élie Roux & Manuel Pégourié-Gonnard
Localisation CTAN	<a href="#">macros/luatex/latex/luainputenc/</a>
URL de développement	<a href="https://github.com/lualatex/luainputenc">https://github.com/lualatex/luainputenc</a>

Aide à la compilation de documents en s'appuyant sur les anciens encodages (soit dans la source, soit dans les polices). Déjà présenté dans l'introduction. Lors de l'exécution, **XeTeX** charge juste *xetex-inputenc*. Sous **pdfTeX**, il charge *inputenc* standard.

### 3-1-4 - luamplib

Moteurs	LuaTeX
Formats	LaTeX, Plain
Auteurs	Hans Hagen, Taco Hoewater & Philipp Gesang
Localisation CTAN	<a href="#">macros/luatex/generic/luamplib/</a>
URL de développement	<a href="https://github.com/lualatex/luamplib">https://github.com/lualatex/luamplib</a>

Fournit une interface agréable pour la bibliothèque **Lua mplib** qui embarque *MetaPost* dans **LuaTeX**.

### 3-1-5 - luacolor

Moteurs	LuaTeX
Formats	LaTeX
Auteurs	Heiko Oberdiek
Localisation CTAN	<a href="#">macros/latex/contrib/oberdiek/</a>

Modifie l'implémentation bas-niveau des couleurs pour utiliser les attributs **LuaTeX** à la place de *whatsits*. Cela rend l'application plus robuste et corrige des bogues étranges tels que le mauvais alignement quand `\color` est placé au début de `\vbox`.

## 3-2 - Niveau développeur

### 3-2-1 - pdftexcmds

Moteurs	LuaTeX, pdfTeX, XeTeX
Formats	LaTeX, Plain
Auteurs	Heiko Oberdiek
Localisation CTAN	<a href="https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/">macros/latex/contrib/oberdiek/</a>

Bien que **LuaTeX** soit surtout un sur-ensemble de **pdfTeX**, quelques primitives ont été supprimées (celles qui sont en quelque sorte remplacées par **Lua**) ou renommées. Cette extension leur donne des noms uniformes à travers les moteurs, y compris **XeTeX** qui a récemment mis en œuvre certaines de ces primitives comme `\strcmp`.

### 3-2-2 - magicnum

Moteurs	LuaTeX, pdfTeX, XeTeX
Formats	LaTeX, Plain
Auteurs	Heiko Oberdiek
Localisation CTAN	<a href="https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/">macros/latex/contrib/oberdiek/</a>

Fournit un accès hiérarchique aux « nombres magiques » comme les catcodes, les types de groupes, etc. utilisés en interne par **TeX** et ses successeurs. Sous **LuaTeX**, une mise en œuvre plus efficace est utilisée et une interface **Lua** est fournie.

### 3-2-3 - lua-alt-getopt

Moteurs	texlua
Formats	command-line
Auteurs	Aleksey Cheusov
Localisation CTAN	<a href="https://ctan.org/ctan/packages/support/lua/lua-alt-getopt">support/lua/lua-alt-getopt</a>
URL de développement	<a href="https://github.com/LuaDist/alt-getopt">https://github.com/LuaDist/alt-getopt</a>

Un analyseur de lignes de commande, principalement compatible avec *POSIX* et *GNU getopt*, pourrait être utilisé dans la ligne de commande scripts **Lua** tels que *mkluatexfontdb* de *luaotfload*.

## 4 - Les formats luatex et lualatex

Cette section est seulement faite pour les développeurs et les utilisateurs curieux. Les autres utilisateurs peuvent l'oublier en toute sécurité. Les informations suivantes s'appliquent à *TeX Live* 2010 et à *MikTeX* 2.9. Bien que je ne l'ai pas vérifié récemment, les versions antérieures de *TeX Live* avaient des implémentations légèrement différentes et moins complètes.

### 4-1 - Les noms des primitives

Comme indiqué dans la section 1.4, les noms des primitives spécifiques **LuaTeX** ne sont pas les mêmes dans le format de **LuaLaTeX** comme dans le manuel de **LuaTeX**. Dans le format de **LuaTeX** (c'est-à-dire **LuaTEX** avec le format ordinaire), les primitives sont disponibles avec leur nom naturel, mais aussi avec le nom préfixé afin de faciliter le développement de forfaits génériques.

Ci-dessous les raisons, copier-coller à partir du fichier `lualatexiniconfig.tex` qui implémente ceci pour le format **LuaLaTeX**.

- 1 Toutes les macros actuelles s'exécutent correctement sur **pdf(e)TeX** de sorte que ces primitives sont laissées intactes.
- 2 Les autres primitives non-TeX82 de **LuaTeX** peuvent causer des conflits de noms avec des macros existantes dans des extensions de macros, surtout quand elles utilisent des noms très « naturels » tels que `\outputbox`, `\mathstyle`, etc. Une telle probabilité pour les conflits de noms n'est pas souhaitable, car la plupart des documents **LaTeX** existants fonctionnent sans changement sous **LuaTeX**, ce qui est très bien.
- 3 L'équipe **LuaTeX** ne veut pas appliquer une politique de préfixation systématique, mais fournit aimablement 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 mieux d'autoriser la préfixation systématique afin d'éviter que chaque extension macro (ou utilisateur) l'autorise avec des préfixes divers et incohérents (y compris le préfixe vide).
- 4 Le préfixe **LuaTeX** a été choisi, car il est déjà utilisé pour certaines primitives telles que `\luatexversion`. De cette façon, ces primitives n'ont pas un double préfixe (pour les détails, voir *tex.enableprimitives* dans le manuel de **LuaTeX**).
- 5 La primitive `\directlua` est fournie à la fois par son nom naturel (permettant une détection facile de **LuaTeX**) et par une version `\luatexdirectlua` préfixée (par souci de cohérence avec `\luatexlualua`).
- 6 Remarques diverses :
  - 1 l'inconvénient évident d'une telle politique préfixe est que les noms utilisés par **LaTeX** ou les macros génériques ne correspondront pas aux noms utilisés dans le manuel. Nous espérons que cela est compensé par le gain de compatibilité ascendante ;
  - 2 toutes les primitives relatives à Unicode commencent déjà avec `\U` et correspondront, peut-être un jour, aux noms des primitives **XeTeX**. Alors peut-être que l'utilisation de tel préfixe n'était pas utile ou désirable pour elles. Malgré tout, nous avons essayé de créer des règles de préfixes aussi simples que possible, de sorte que le point précédent ne devienne pire ;
  - 3 peut-être qu'un jour, nous éprouverons le besoin de définir les primitives sans préfixe. Si cela arrive, il sera facile d'ajouter ces primitives sans préfixe dans les formats qui resteront compatibles avec les primitives avec préfixes. Cela n'aurait pas fonctionné dans l'autre sens, c'est-à-dire, réaliser tardivement que nous n'aurions pas dû définir les primitives sans préfixes, les rajouter reviendrait à rendre inutilisables toutes les extensions à macros déjà écrites spécifiquement pour **luaTeX**.

## 4-2 - `\jobname`

Le noyau **LaTeX** et un grand nombre d'extensions utilisent des constructions comme `\input\jobname.aux` à des fins diverses. Lorsque `\jobname` contient des espaces, cela produit des problèmes, puisque l'argument de `\input` se termine avec le premier espace. Pour contourner ce problème, pdfTeX protège automatiquement `\jobname` en cas de besoin avec des guillemets, mais **LuaTeX** ne le fait pas pour une autre raison. Une solution presque parfaite est incluse avec *LaTeX-based* (par opposition à Plain-based) dans les formats **LuaTeX**.

Cela ne fonctionne pas cependant si **LuaTeX** est invoqué comme `lualatex "input name"`, par opposition à la méthode plus habituelle `lualatex name`. Pour contourner cette limitation de la solution de contournement inclus dans le format, il faut spécifier explicitement un nom, comme dans `lualatex jobname = nom "input"`. Ou mieux encore, il suffit de ne pas utiliser d'espaces dans les noms de vos fichiers **TeX**.

Pour plus de détails, voir cette [vieille discussion](#) et [celle-ci plus récente](#) sur les listes de diffusion de **LuaTeX** et `lualatexquotejobname.tex` pour la mise en œuvre de la solution de contournement.

## 4-3 - La césure

**LuaTeX** offre un chargement dynamique des modèles de césure. Son support au sein de *babel* et *Polyglossia* apparaît seulement sur *TeX Live* 2013, mais doit bien fonctionner maintenant.

La documentation et les détails de mise en œuvre sont inclus dans `luatex-hyphen.pdf`. Les sources font partie du [projet de texhyphen](#).

## 4-4 - Les codes

Le moteur lui-même ne définit pas `\catcodes`, `\lccodes`, etc. pour les caractères non ASCII. Des corrections dans `\lccodes` en particulier sont nécessaires pour que la césure fonctionne. Les formats pour LuaTeX comprennent maintenant `luatex-unicode-letters.tex`, une version modifiée de `unicode-letters.tex` de la distribution de **XeTeX** qui s'occupe de positionner les valeurs conformément à la norme Unicode.

Cela a été ajouté après que *TeX Live 2010* soit sorti. Il est donc fortement recommandé de mettre à jour votre installation si vous voulez profiter de la césure correcte pour le texte non-ASCII.

## 5 - Ce qui fonctionne bien, partiellement ou pas encore

### 5-1 - Ce qui fonctionne bien

#### 5-1-1 - Unicode

LaTeX classique offre un certain niveau de support pour UTF-8 dans les fichiers d'entrée. Cependant, à un bas niveau, les caractères non-ASCII ne sont pas atomiques, ils se composent de plusieurs pièces élémentaires (appelées tokens pour les TeXniciens). Par conséquent, certaines extensions qui analysent un texte caractère par caractère ou effectuent d'autres opérations sur des caractères atomiques (comme le changement de leurs catcodes) ont souvent des problèmes avec UTF-8 dans **LaTeX** classique. Par exemple, vous ne pouvez pas utiliser n'importe quel caractère non-ASCII pour insérer de courts textes avec *fancyvrb*, etc.

La bonne nouvelle est qu'avec **LuaLaTeX**, quelques-unes des caractéristiques de ces extensions commencent à travailler sur des caractères Unicode arbitraires sans avoir à modifier l'extension. La mauvaise nouvelle est que cela n'est pas toujours vrai. Voir la section suivante pour plus de détails.

### 5-2 - Ce qui fonctionne partiellement

#### 5-2-1 - microtype

L'extension *microtype* a un support limité pour **LuaTeX**. Plus précisément, à partir de la version 2.5 du 13/03/2013, protrusions et expansions sont disponibles et activées par défaut en mode PDF, mais le crénage, l'espacement et le suivi ne sont pas pris en charge (voir le tableau 1 dans la section 3.1 de *microtype.pdf*).

D'autre part, *luaotfload*, chargé par *fontspec*, prend en charge un grand nombre de fonctionnalités microtypographiques. Donc, le seul problème est l'absence d'une interface unifiée.

#### 5-2-2 - xunicode

La caractéristique principale de l'extension *xunicode* est de s'assurer que les séquences de contrôles habituelles, pour les caractères non-ASCII (comme `\e`), font la bonne chose dans un contexte Unicode. Elle pourrait probablement fonctionner avec **LuaTeX**, mais travaille uniquement avec **XeTeX**. Cependant, *fontspec* utilise une astuce pour la charger de toute façon. Donc, vous ne pouvez pas charger explicitement, mais vous n'en avez pas besoin non plus, puisque *fontspec* le fait déjà.

### 5-2-3 - Les encodages

Comme indiqué dans la section ci-dessus, un certain nombre de choses qui étaient problématiques avec UTF-8 sur **LaTeX** classique fonctionnent spontanément, mais pas toujours. Par exemple, avec l'extension *listings* sur **LuaLaTeX**, vous pouvez utiliser uniquement les caractères inférieurs à 256 (c'est-à-dire caractères Latin-1) à l'intérieur des

environnements *listings* (mais bien sûr la gamme complète de l'Unicode est toujours disponible dans le reste du document).

## 5-2-4 - metrics

Cette extension ne travaille pas exactement de la même manière que **pdfTeX** ou **XeTeX**. Vous pourrez observer des différences mineures dans la mise en page et la césure de votre texte.

Elles peuvent être dues aux variations entre deux versions de la même police de caractères utilisée par les différents moteurs, les ajustements apportés à la césure, les algorithmes de ligature ou de crénage (où par exemple, le premier mot d'un paragraphe et les mots suivants dans une police différente peuvent maintenant être liés par un trait d'union), ou des différences dans les motifs de césure utilisés (les modèles utilisés par **pdfTeX** sont essentiellement gelés, mais **LuaTeX** et **XeTeX** utilisent une version plus récente pour certaines langues) pour cette langue.

Si jamais vous constatez une différence majeure entre **pdfLaTeX** et **LuaLaTeX** pour les mêmes polices, il n'est pas du tout improbable que ce soit un bogue dans **LuaTeX** (8) ou dans la police impliquée. Comme d'habitude, assurez-vous que votre distribution est à jour avant de signaler un tel problème.

## 5-2-5 - babel

Travaillant la plupart du temps sans problème pour les langues latines. Pour les autres langues, votre vitesse peut varier. Même pour les langues latines, les problèmes associés au codage peuvent se produire.

## 5-2-6 - polyglossia

Une extension plus moderne, mais moins complète pour le support multilingue, *polyglossia* est également disponible et devrait être préférée, même s'il ne prend pas encore en charge toutes les fonctionnalités de *babel*.

## 5-3 - Ce qui ne fonctionne pas (encore)

### 5-3-1 - Les anciens encodages

Les extensions d'encodages utilisant l'entrée (fichiers source) ou la sortie (polices) sont très susceptibles avec **LuaTeX**. Elles comprennent *inputenc*, *fontenc*, *textcomp*, et des paquets de polices probablement plus classiques tels que *mathptmx* ou *fourier*. La bonne nouvelle est qu'Unicode est le moyen le plus puissant pour gérer ces problèmes d'encodage que les anciennes extensions tentent de résoudre de sorte que vous ne devriez probablement pas avoir besoin de ces paquets de toute façon. Cependant, tout n'est pas encore porté sur le nouveau monde brillant de l'Unicode. Vous pourriez donc avoir un ensemble plus limité (ou simplement différent) de choix disponibles pour un certain temps (cela est particulièrement vrai pour les polices).

### 5-3-2 - Les espaces

Les espaces dans les noms de fichiers ne sont pas très bien pris en charge dans le monde de **TeX** en général. Ce n'est pas vraiment mieux avec **LuaTeX**. En outre, pour des raisons compliquées, les choses peuvent être pires si vous avez des espaces dans le nom de votre fichier principal de **TeX** et n'invoquez pas **LuaTeX** de la manière usuelle. Si vous l'appellez avec la manière habituelle, tout devrait fonctionner et je ne vous dirais pas à quoi ressemble l'invocation inhabituelle. Sinon, lisez le point sur *jobname* dans la section 4 pour une solution de contournement et les détails techniques. Mieux encore, n'utilisez pas les espaces dans les noms de vos fichiers de **TeX**.



## 6 - Remerciements

Nous remercions l'auteur, **Manuel Pégourié-Gonnard**, de nous avoir aimablement autorisé à publier son article. Nous remercions aussi **ram-0000** pour sa traduction, **-Nikopol-** pour sa relecture technique ainsi que **milkoseck** pour sa relecture orthographique.

Les commentaires et les suggestions d'amélioration sont les bienvenus, ce document est toujours en cours d'évolution, merci pour votre compréhension et patience. Après votre lecture, n'hésitez pas. **Commentez**

- 1 : Bien que se concentrant sur LuaLaTeX, il comprend des informations pertinentes sur LuaTeX avec le format ordinaire aussi.
- 2 : Comme son nom l'indique, il gère seulement l'encodage en entrée. Les détails de codages de la mise en œuvre par LaTeX des fontes impliquent que ce paquet est nécessaire (et qu'il fonctionne) pour les vieilles polices aussi.
- 3 : Le nom signifie probablement « impression de chaînes de caractères » et non pas « courir très vite sur une courte période de temps »
- 4 : D'accord, ce sont généralement des actions belles et utiles, mais dans ce cas elles sont très probablement inattendues donc je les appelle nocives.
- 5 : Ah ! Trop tard, vous l'avez déjà lu.
- 6 : Si vous avez déjà utilisé SQL, alors le concept d'échapper les chaînes de caractères n'est pas nouveau pour vous.
- 7 : Pour TeX Live, pensez à utiliser le référentiel de **tlcontrib** complémentaire.
- 8 : Par exemple, LuaTeX 0.60 avait un bogue qui empêchait toute césure après une ligature --- jusqu'à la fin du paragraphe.