Eine Einführung zu LuaLaTeX*

Manuel Pégourié-Gonnard <mpg@elzevir.fr>

28. Dezember 2013

Dies ist sozusagen eine Art Landkarte oder ein Reiseführer zu einer neuen Welt, der Welt von LuaLaTeX.[1] Die Einführung richtet sich sowohl an absolute LuaLaTeX-Anfänger, die aber schon einmal mit LaTeX gearbeitet haben sollten, als auch an erfahrene Paketentwickler. Sie soll „umfassend“ sein in dem Sinne, dass sie alle erheblichen Quellen nachweist. Informationen, die sonst nur verstreut vorliegen, werden hier gesammelt, und es wird auf weiteres Material verwiesen.

Rückmeldungen und Verbesserungsvorschläge sind willkommen. Es handelt sich um ein *work in progress*.

Inhaltsverzeichnis

*Deutsche Fassung von *A guide to LuaLaTeX* (Stand: 10. Mai 2013 mit Ergänzungen von Elie Roux zu TeX Live 2013 und Lua 5.2). Die Übersetzung ins Deutsche besorgte Jürgen Fenn. Der Text steht, wie das Original, unter der GNU Free Documentation Licence, GFDL, http://www.gnu.org/licenses/fdl.html. No invariant sections or cover text.

[1] Obwohl es hier nur um LuaLaTeX gehen soll, enthält die Einführung auch Angaben, die zum Arbeiten mit LuaTeX oder mit Plain TeX nützlich sind.

1 Einführung

1.1 Was ist LuaLaTeX?

Um diese Frage zu beantworten, müssen wir uns einige Einzelheiten zur TeX-Welt in Erinnerung rufen, mit denen man sich normalerweise nicht zu beschäftigen braucht, nämlich den Unterschied zwischen der *Engine* und dem *Format*. Als Engine bezeichnet man das Computerprogramm, das den Textsatz versieht, während man unter einem Format all die Makros versteht, die von der Engine ausgeführt werden. Üblicherweise werden sie geladen, wenn die Engine mit einem bestimmten Namen aufgerufen wird.

Ein Format ist mehr oder weniger wie eine Dokumentenklasse oder wie ein Paket, mit dem Unterschied, dass es mit einem bestimmten Befehlsnamen verbunden ist. Beispielsweise könnte es einen Befehl `latex-article` geben, der dasselbe tut wie der Befehl `latex`, mit der einen Ausnahme, dass man zu Beginn der Eingabedatei nicht `\documentclass{article}` angeben müsste. In den heutigen TeX-Distributionen ist es ganz ähnlich mit dem Befehl `pdflatex`. Wenn man ihn aufruft, geschieht dasselbe, wie wenn man den Befehl `pdftex` eingäbe, mit der Ausnahme, dass man LaTeX nicht mehr ausdrücklich zu Beginn seiner Eingabedatei laden muss. Das ist praktisch, weil es vieles wesentlich vereinfacht.

Das Schöne an Formaten ist, dass sie mächtige Befehle mithilfe der grundlegenden Werkzeuge definieren, die eine Engine bereitstellt. Die Mächtigkeit des Formats wird aber manchmal von derjenigen der Tools beschränkt, die eine Engine mitbringt. So kam es, dass die einen damit begannen, immer mächtigere Engines zu entwickeln, damit andere immer noch mächtigere Formate oder Pakete schreiben konnten. Die bekanntesten Engines heutzutage (außer dem Original TeX) sind pdfTeX, XeTeX und LuaTeX.

Um alles noch komplizierter zu machen, muss man sich weiterhin daran erinnern, dass die ursprüngliche TeX-Engine nur DVI-Dateien erzeugen konnte, während ihre Nachfolger auch PDF-Dateien ausgeben. Jeder Befehl auf Ihrem System entspricht nun einer bestimmten Engine mit einem bestimmten Format und einem bestimmten Ausgabemodus. Die Tabelle 1.1 auf der nächsten Seite fasst das zusammen, in den Zeilen nach Formaten, in den Spalten nach Engines geordnet. In jeder Tabellenzelle steht in der ersten Zeile der Befehl, mit dem diese Engine für das jeweilige Format im DVI-Modus aufgerufen wird, in der zweiten Zeile entsprechend für den PDF-Modus:

Und jetzt können wir die Frage, die über diesem Abschnitt steht, beantworten: LuaLaTeX ist

Tabelle 1: Aufruf verschiedener TEX-Engines nach Formaten.

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

eine LuaTEX-Engine mit einem LATEX-Format. Zugegeben, die Antwort ist nicht befriedigend, wenn Sie noch nicht wissen, was LuaTEX und LATEX sind.

Wahrscheinlich wissen Sie schon, dass LATEX das allgemeine Framework ist, in dem Dokumente mit `\documentclass` beginnen, Pakete mit `\usepackage` geladen werden, in dem man Schriften in einer cleveren Weise ausgewählt kann (nämlich so, dass man zu fett umschalten kann, während die Einstellung „kursiv“ erhalten bleibt), in dem Seiten mit einem komplizierten Algorithmus aufgebaut werden, einschließlich Überschriften, Fußzeilen, Fußnoten, Randnotizen, Gleitobjekten usw. Das meiste davon funktioniert so auch bei LuaLATEX, es kommen aber neue und noch mächtigere Pakete hinzu, mit denen Teile des ganzen Systems noch besser funktionieren.

Was also ist LuaTEX? Kurz gesagt: Die heißeste TEX-Engine derzeit! Etwas ausführlicher: Es soll einmal ein würdiger Nachfolger von pdfTEX werden und beinhaltet deshalb all dessen Kernfeatures: Direktes Erzeugen von PDF-Dateien mit fortgeschrittenen PDF-Features und mikroypografischen Verbesserungen im Vergleich zu den typografischen Algorithmen von TEX. Die wichtigesten Eigenschaften von LuaTEX sind:

1. Native Unterstützung von Unicode, dem modernen Standard zur Klassifizierung und zur Kodierung von Zeichen, der alle Symbole der Welt umfasst, von Englisch über traditionellem Chinesisch bis hin zu Arabisch, und natürlich auch viele mathematische (oder sonstige speziellen) Symbole.

2. Lua als eingebettete Skriptsprache (zu Einzelheiten siehe Abschnitt 1.3).

3. Viele wunderbare Lua-Bibliotheken, einschließlich:
   - `fontloader` für moderne Font-Formate wie TrueType und OpenType;
   - `font` für fortgeschrittene Manipulationen an Schriften innerhalb des Dokuments;
   - `mplib`, eine eingebettete Version des Grafikprogramms MetaPost;
   - `callback`, das den Zugriff auf Teile der TEX-Engine erlaubt, die früher dem Programmierer verschlossen waren;
   - Werkzeug-Bibliotheken, um Bilder, PDF-Dateien usw. zu bearbeiten.

Einige dieser Features, beispielsweise die Unicode-Unterstützung, wirken sich direkt auf alle Dokumente aus, während andere nur Werkzeuge bereitstellen, von denen Paketautoren profitieren werden, um noch mächtigere Befehle oder weitere Verbesserungen zu entwickeln.

1.2 Der Umstieg von LaTeX auf LuaLaTeX

Wie schon im vorhergehenden Abschnitt erklärt, ist LuaTeX im wesentlichen wie LaTeX, es gibt nur wenige Unterschiede. Allerdings stehen bei LuaTeX einige sehr viel mächtigere Pakete und Tools zur Verfügung, auf die man zurückgreifen kann. An dieser Stelle soll es nur um das absolut Notwendige gehen, das man unbedingt wissen muß, um Dokumente mit LuaTeX erzeugen zu können. Alles weitere zu den benötigten Paketen wird dann im Rest dieser Einleitung erklärt.

Genaugenommen gibt es nur vier Unterschiede:[2]

1. Laden Sie nicht inputenc, sondern kodieren Sie Ihre Eingabedatei von vornherein in UTF-8.
2. Laden Sie nicht fontenc oder textcomp, sondern fontspec.
3. babel funktioniert mit LuaLaTeX, aber Sie können stattdessen auch polyglossia laden.
4. Verwenden Sie keine Pakete, die die Schriftarten ändern, sondern verwenden Sie dafür die Befehle aus dem Paket fontspec.

Das einzige, womit Sie sich neu vertraut machen müssen, ist also das Paket fontspec, was aber einfach ist: Die Hauptschriftart (serif) wählt man aus mit `\setmainfont`, den serifenlosen Schnitt aktiviert man mit `\setsansfont` und die von der Schreibmaschine her bekannte Festbreitenschrift mit `\setmonofont`. Das Argument zu diesen Befehlen ist benutzerfreundlich: Es ist der Name des Fonts, also zum Beispiel `Latin Modern Roman` und nicht `ec-lmr10`. Wenn Sie vor diesen Befehlen ein `\defaultfontfeatures{Ligatures=TeX}` platzieren, bleiben die üblichen TeX-Ersetzungen verfügbar (etwa `---` für einen Geviertstrich).

Die gute Nachricht ist also: Sie können auf jeden Font, der auf Ihrem System installiert ist, direkt zugreifen, zusätzlich zu den Schriften, die bei Ihrer TeX-Distribution dabei sind, einschließlich TrueType- und OpenType-Fonts, und sie haben dabei auch Zugriff auf die fortgeschrittenen Features all dieser Schriften. Das heißt, es ist jetzt einfach geworden, irgendeinen modernen Font, den Sie herunterladen oder kaufen, zur Verwendung mit LuaTeX zu installieren und dabei alle Möglichkeiten auszuschöpfen.

Und nun die schlechte Nachricht: Es ist nicht immer einfach, eine Liste mit allen verfügbaren Fonts zu bekommen. Unter Windows und TeX Live listet sie der Befehl `fc-list` alle auf, aber die lange und unübersichtliche Liste ist nicht sehr benutzerfreundlich. Unter Mac OS X zeigt die Schriftensammlung alle auf dem System installierten Schriften an, aber nicht diejenigen aus Ihrer TeX-Distribution. Unter Linux gilt für `fc-list` das gleiche wie zuvor. Und was noch schlechter ist: Es ist nicht einfach, auf Ihre alten Fonts auf diese Weise zuzugreifen. Zum Glück gibt es mit der Zeit immer mehr moderne Formate (zwar nicht tagtäglich, aber so etwa monatlich oder jährlich, wenn man nur die hochwertigen Schriften berücksichtigt).

Übrigens gilt das Vorstehende bis hierhin auch für XeLaTeX, also LaTeX über XeTeX. Denn XeTeX hat zwei der wichtigsten Eigenschaften mit LuaTeX gemein: Den nativen Unicode-Support und die Unterstützung moderner Font-Formate. Nur die übrigen Features von LuaTeX

[2] Anmerkung des Übersetzers: Im Original ist an dieser Stelle die Rede von „drei Unterschieden“, es folgt dann aber eine Aufzählung mit vier Punkten, die es nahelegt, diese Anzahl auch in der Einleitung des Absatzes anzuführen.

fehlen in XeTeX. Andererseits ist XeTeX zumindest derzeit stabiler als LuaTeX, das sich ja noch in Entwicklung befindet. Obwohl die beiden Engines sehr unterschiedlich mit Schriften umgehen, bietet fontspec hierfür eine größtenteils einheitliche Schnittstelle an.

Um also von den neuen Möglichkeiten zu profitieren, die LuaTeX Ihnen bietet, müssen Sie nur auf wenige Dinge aus der alten Welt verzichten, nämlich auf die Fonts, die nicht in einem modernen Format verfügbar sind (außerdem auf die Freiheit, Ihre Eingabedatei in einer beliebigen Kodierung zu speichern, aber UTF-8 ist eine in vielerlei Hinsicht überlegene Kodierung, so dass das kaum ins Gewicht fällt). Das Paket luainputenc gleicht dies teilweise aus, damit kann man diese Dinge wieder zurückerlangen,[3] wahrscheinlich um den Preis, die echte Unicode-Unterstützung wiederum zu verlieren.

Das ist alles, was Sie zum Anfang wissen müssen, um Dokumente mit LuaTeX erzeugen zu können. Es empfiehlt sich, einen Blick in die Anleitung zu fontspec zu werfen und einmal ein kleines Dokument mit beliebigen Schriften zu kompilieren. Dann können Sie getrost den Rest dieser Anleitung überfliegen, wenn Sie das möchten. Abschnitt 5 listet alle anderen Unterschiede zwischen dem konventionellem LaTeX und LuaLaTeX auf, die mir bekannt sind.

1.3 Grundlegendes zu Lua in TeX

Lua ist eine schöne, kleine Programmiersprache, die ganz offenbar weniger überraschend und viel einfacher zu erlernen ist als TeX. Die knappste Einführung ist das gut lesbare Buch *Programming in Lua*, dessen erste Auflage unter `http://www.lua.org/pil/` frei online verfügbar ist. Als Einstieg empfiehlt es sich, die Kapitel 1 bis 5 zu lesen und einen kurzen Blick auf den Teil 3 zu werfen. Alle Bibliotheken, die in Kapitel 3 erwähnt werden, sind in LuaTeX enthalten, nur `os` ist aus Sicherheitsgründen eingeschränkt.

Je nach Ihrem Programmierstil, werden Sie sich unmittelbar für Teil 1 oder Teil 2 interessieren, die die mehr fortgeschrittenen Features der Sprache erklären, nur Teil 4 ist für LuaTeX ohne Belang, außer, natürlich, wenn Sie LuaTeX selbst hacken wollen. Auch das *Lua reference manual* ist unter `http://www.lua.org/manual/5.2/` online verfügbar. Es hat auch ein praktisches Stichwortverzeichnis.

Wenden wir uns also Lua und LuaTeX zu. Um Lua-Quelltext mit TeX auszuführen, gibt es den Befehl `\directlua`, dem man beliebigen Lua-Code als Argument übergeben kann. Umgekehrt können Informationen von Lua an TeX mithilfe von `tex.sprint` übergeben werden.[4] Beispielsweise ergibt

```
die übliche Annäherung $\pi = \directlua{tex.sprint(math.pi)}$
```

die Ausgabe „die übliche Annäherung $\pi = 3.1415926535898$“ im Dokument. So einfach ist es also, TeX und Lua zu mischen.

Allerdings gibt es ein paar Punkte, auf die man dabei achten muss. Betrachten wir zunächst die Richtung Lua zu TeX, denn sie ist am einfachsten (weil es mehr Lua als TeX ist). Im LuaTeX-Handbuch steht, dass es noch eine weitere Funktion mit einem einfacheren Namen

[3] Dem Namen nach geht es bei dem Paket zwar nur um die Eingabekodierungen, die Einzelheiten des Font-Encodings von LaTeX bedingen aber, dass das Paket auch mit alten Fonts funktioniert.

[4] Wahrscheinlich steht der Name für „string print“, und nicht für „laufe kurzzeitig sehr schnell”.

gibt, um Daten von Lua an TEX weiterzugeben, nämlich `tex.print`. Damit wird sozusagen eine ganze Zeile in den TEX-Quelltext eingefügt, die aus ihren Argumenten besteht. Falls noch nicht bekannt: TEX vollführt viele nervige[5] Dinge mit ganzen Zeilen im Quelltext: Leerzeichen werden zu Anfang und am Ende einer Zeile ignoriert und ein Zeilenende-Zeichen wird angehängt. Meistens will man das nicht, deshalb empfehle ich, `tex.sprint` zu verwenden, das sozusagen sein eigenes Argument in die aktuelle Zeile einfügt, was zu besser vorhersehbaren Ergebnissen führt.

Wenn Sie ein fortgeschrittener TEXniker sind, werden Ihnen Catcodes ein Begriff sein. `tex.print` und seine Varianten erlauben es, nahezu vollständige Kontrolle über die Catcodes auszuüben, die zum Kodieren von Argumenten verwendet werden, denn es ist möglich, als erstes Argument eine Catcode-Tabelle anzugeben. Dazu sollten Sie den entsprechenden Abschnitt im LuaTEX-Handbuch lesen (derzeit Abschnitt 2.7.6). Wenn Sie nicht mit Catcodes arbeiten, überspringen Sie diesen Abschnitt einfach.[6]

Wenden wir uns nun `\directlua` zu. Um einen Eindruck zu erhalten, wie diese Funktion arbeitet, stellen Sie sich einmal vor, es wäre ein `\write`-Befehl, der aber nur in eine virtuelle Datei schreibt und dann dafür sorgt, dass diese Datei direkt an den Lua-Interpreter übergeben wird. Auf der Lua-Seite heißt das, jedes Argument eines `\directlua`-Befehls hat seinen eigenen Scope: Lokale Variable des einen sind für das andere nicht sichtbar (was durchaus sinnvoll ist, sich aber nicht von selbst versteht).

Aber das allerwichtigste bei alledem ist, dass das Argument zuerst von TEX gelesen und in ein Token umgewandelt wird, bevor es expandiert und zurück in einen Plain-String verwandelt werden kann. Erst dann wird es an den Lua-Interpreter übergeben. Die Verarbeitung durch TEX hat mehrere Folgen. Eine davon ist, dass Zeilenenden in Leerzeichen umgesetzt werden, so dass der Lua-Interpreter nur eine (lange) Zeile Input sieht. Da Lua eine *free-form language* ist, macht das im allgemeinen nichts aus, außer bei Kommentaren:

```
\directlua{eine_funktion()
  -- ein Kommentar
  eineweitere_funktion()}
```

wird wahrscheinlich nicht das ergeben, was Sie erwarten würden: `eineweitere_funktion()` wird als Teil des Kommentars behandelt (denn es ist, wohlgemerkt, nur eine Zeile).

Eine weitere Folge dieser Eigenheit von TEX ist, dass aufeinanderfolgende Leerzeichen zu einem Leerzeichen zusammengefasst werden. Dabei fallen die TEX-Kommentare weg. Überraschenderweise lautet deshalb die korrekte Version des vorhergehenden Beispiels so:

```
\directlua{eine_funktion()
  % ein Kommentar
  eineweitere_funktion()}
```

[5] Gut, meistens ist das hilfreich, aber in diesem Fall erwartet man das nicht unbedingt, deshalb bezeichne ich sie als nervig.

[6] Ach, Sie haben ihn ja schon zuende gelesen.

Da das Argument innerhalb von `\write` auftaucht, wird es intern ersetzt (expandiert). Falls Sie nicht wissen, was das bedeutet: Man könnte sagen, dass die Probleme, die bei der Inline-Ersetzung auftreten, gerade das sind, was die Programmierung in TEX so schwierig macht. Mehr möchte ich dazu aber lieber nicht sagen.

Es tut mir leid, dass die letzten drei Abschnitte etwas TEXnischer waren, ich meine aber, dass Sie das alles durchaus erfahren und beachten sollten. Als Belohnung dafür, dass Sie weitergelesen haben, folgt nun ein Trick zum Debuggen. Fügen Sie doch bitte einmal die folgenden Zeilen zu Anfang in Ihr Dokument ein:

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

Wenn Sie nicht so ganz verstehen, warum ein bestimmter Aufruf zu `\directlua` nicht das tut, was Sie erwarten würden, ersetzen Sie diesen Teil des Befehls doch einfach durch `\directluadebug`, kompilieren Sie es wie gewohnt und schauen Sie in die Datei `luadebug.lua`. Dann sehen Sie, was der Lua-Interpreter tatsächlich gelesen hat.

Diese Befehle und Umgebungen werden von dem Paket luacode bereitgestellt, das bei solchen Probleme mehr oder weniger hilfreich ist. Wann immer man es aber mit etwas komplizierterem Lua-Code zu tun hat, sollte man besser eine externe Datei verwenden, die nur den Lua-Code enthält, in dem die Funktionen definiert sind, diesen dann laden und die Funktionen verwenden lassen. Ein Beispiel:

```
\directlua{dofile("meine-lua-funktionen.lua")}
\newcommand*{\tollesmakro}[2]{%
  \directlua{meine-tollen-lua-funktionen("\luatexluaescapestring{#1}", #2)}}
```

Das Beispiel geht davon aus, dass `meine-tollen-lua-funktionen` in `meine-lua-funktionen.lua` definiert sind und dass sie einen String und eine Zahl als Argument übernehmen. Die Primitive `\luatexluaescapestring` wird eingesetzt, um die Backslashes und die Doppelquotes im String-Argument zu escapen, die sonst den Lua-Parser durcheinanderbringen würden.[7]

Das ist alles, was es zu Lua in TEX zu sagen gibt. Und wenn Sie sich jetzt noch fragen, warum `\luatexluaescapestring` so einen langen und komischen Namen hat, dann sollten Sie auch den folgenden Abschnitt noch lesen.

1.4 Was man sonst noch wissen sollte

Falls es sich noch nicht von selbst versteht: Das LuaTEX-Handbuch `luatexref-t.pdf` ist natürlich ein ganz formidables Nachschlagewerk zu LuaTEX, auf das man immer wieder zurückgreifen sollte (obwohl es zugegebenermaßen manchmal etwas trocken und technisch daherkommt).

[7] Wenn Sie jemals SQL verwendet haben, werden Sie diesen Ansatz, um Strings zu escapen wahrscheinlich schon kennen.

Wichtig ist, dass die Namen der neuen Primitiven von LuaTEX, wie sie im Handbuch erscheinen, nicht die tatsächlichen Namen sind, die man in LuaLATEX verwenden kann. Um Konflikte mit anderen Makronamen zu vermeiden, werden alle neuen Primitiven mit der Präfix `\luatex` versehen, soweit sie nicht ohnehin schon damit anfangen. Aus `\luaescapestring` wird also `\luatexluaescapetring`, während es für `\luatexversion` bei `\luatexversion` bleibt. Die Logik dahinter wird in Abschnitt 4 näher erklärt.

Habe ich eigentlich schon erwähnt, dass LuaTEX weiterhin als Beta vorliegt und Version 1.0 erst für TEX Live 2016[8] erwartet wird? Mehr dazu auf der Roadmap-Seite auf LuaTEX-Website http://luatex.org/. Stabile Betas werden laufend veröffentlicht und sind in TEX Live seit 2008 und in MikTEX seit Version 2.9 enthalten.

Unter diesen Umständen ist es nicht weiter überraschend, dass die Unterstützung für LuaTEX in LATEX noch ganz frisch ist, was nichts anders bedeutet, als dass sie noch voller frischer Bugs sein wird und dass sich noch einiges in nächster Zeit verändern kann. Deshalb ist es ratsam, seine TEX-Distribution laufend zu aktualisieren[9] und LuaLATEX für wichtige Dokumente bis auf weiteres möglichst nicht zu verwenden.

Allgemein gesagt, fasst diese Einführung alles zusammen, was es derzeit zu LuaLATEX zu sagen gibt. Weitere Entwicklungen werden dabei nicht mehr berücksichtigt. Deshalb sollten Sie Ihre Distribution immer vollständig aktualisieren, um mit zueinander passenden Versionen dieser Einführung, der Pakete, der Formate und der Engine, die darin beschrieben werden, zu arbeiten.

2 Die wichtigsten Pakete und Befehle

In diesem Abschnitt werden alle Pakete vorgestellt, die ein Anwender wahrscheinlich nützlich finden wird – und die ein Entwickler unbedingt kennen sollte.

2.1 Für Anwender

fontspec
Engine(s): XeTEX, LuaTEX. **Formate:** LATEX.
Autor(en): Will Robertson.
CTAN: macros/latex/contrib/fontspec/.
Projekt-URL: https://github.com/wspr/fontspec/.
Elegantes Interface zum Zugriff auf Schriften, gut gestaltet und integriert in das LATEX-Font-Selection-Scheme. Wurde schon im vorigen Abschnitt vorgestellt.

polyglossia
Engine(s): XeTEX, LuaTEX. **Formate:** LATEX.
Autor(en): François Charette & Arthur Reutenauer.
CTAN: macros/latex/contrib/polyglossia/.
Projekt-URL: https://github.com/reutenauer/polyglossia/.
Ein einfacher und moderner Ersatz für Babel, arbeitet mit fontspec zusammen.

[8] Anmerkung des Übersetzers: An dieser Stelle wurde die Angabe „Frühling 2014“ anhand der aktuellen Roadmap aktualisiert (Stand: Dezember 2013).
[9] Für TEX Live gibt es die Ergänzung tlcontrib unter http://tlcontrib.metatex.org/.

2.2 Für Entwickler

2.2.1 Namenskonventionen

In TEX sind Escape-Sequenzen, die mit `\luatex` anfangen, für Primitiven reserviert. Es ist sehr zu empfehlen, solche Namen *nicht* in Definitionen zu verwenden, um Konflikte mit zukünftigen Versionen von LuaTEX zu vermeiden. Will man betonen, dass ein Makro zu LuaTEX gehört, so wäre mein Rat, stattdessen die Präfix `\lua` zu benutzen (ohne ein anschließendes `tex`). Man kann die Präfix `\luatex@` für interne Makros nehmen, da Primitiven niemals ein `@` enthalten, aber auch das könnte Verwirrung stiften. Außerdem verwenden Sie wahrscheinlich ohnehin schon eine einmalige Präfix in allen internen Makros in Ihren Paketen, nicht wahr?

In Lua sollte der globale Namensraum so sauber wie möglich gehalten werden. Benutzen Sie also eine Tabelle `meinpaket` und setzen sie alle Funktionen und Objekte in diese Tabelle. Das obsolete `module()` in Lua sollte nicht mehr verwendet werden. Andere Ansätze zur Modulverwaltung in Lua werden in Kapitel 15 von *Programming in Lua* unter `http://www.lua.org/pil/15.html` vorgestellt; weitere Beispiele können `luatexbase-modutils.pdf` entnommen werden. Es hat sich eingebürgert, `local` für interne Variablen und Funktionen zu verwenden. Um Konflikte mit zukünftigen Versionen von LuaTEX zu vermeiden, dürfen die Namensräume der Standardbibliotheken von LuaTEX nicht geändert werden.

2.2.2 Engine- und Mode-Erkennung

Es gibt mehrere Pakete, die es erlauben, die Engine, die gerade zur Bearbeitung des Dokuments eingesetzt wird, abzufragen.

ifluatex **Engine(s):** alle. **Formate:** LaTeX, Plain.
Autor(en): Heiko Oberdiek.
CTAN: `macros/latex/contrib/oberdiek/`.
Definiert den Befehl `\ifluatex` und stellt sicher, dass `\luatexversion` verfügbar ist.

iftex **Engine(s):** alle. **Formate:** LaTeX, Plain.
Autor(en): Vafa Khalighi.
CTAN: `macros/latex/contrib/iftex/`.
Projekt-URL: `http://bitbucket.org/vafa/iftex`.
Stellt `\ifPDFTeX`, `\ifXeTeX`, `\ifLuaTeX` bereit und entsprechende `\Require`-Befehle.

expl3 **Engine(s):** alle. **Formate:** LaTeX.
Autor(en): The LaTeX3 Project.
CTAN: `macros/latex/contrib/expl3/`.
Projekt-URL: `http://www.latex-project.org/code.html`.
Neben *vielen* weiteren Dingen, werden `\luatex_if_engine:TF`, `\xetex_if_engine:TF` und ihre Varianten bereitgestellt.

ifpdf **Engine(s):** alle. **Formate:** LaTeX, Plain.
Autor(en): Heiko Oberdiek.
CTAN: `macros/latex/contrib/oberdiek/`.
Stellt den `\ifpdf`-Schalter bereit. LuaTEX kann, wie pdfTEX, entweder PDF oder DVI ausgeben;

letzteres ist weniger sinnvoll, da DVI keine modernen Features wie Unicode oder die entsprechenden Font-Formate erlaubt. Der `\ifpdf`-Schalter ist nur dann „wahr“, wenn pdfTeX oder LuaTeX im PDF-Modus betrieben wird. XeTeX wird nicht unterstützt, weil es mit PDF anders umgeht.

2.2.3 Grundlegende Pakete

luatexbase **Engine(s):** LuaTeX. **Formate:** LaTeX, Plain.
Autor(en): Élie Roux, Manuel Pégourié-Gonnard & Philipp Gesang.
CTAN: `macros/luatex/generic/luatexbase/`.
Projekt-URL: `https://github.com/lualatex/luatexbase`.
Die Formate Plain und LaTeX stellen Makros bereit und verwalten grundlegende Ressourcen von TeX, beispielsweise Zähler oder Boxenregister. LuaTeX führt neue Ressourcen ein, die mit anderen Paketen geteilt werden sollen. Dieses Paket enthält hierfür elementare Werkzeuge für die erweiterten konventionellen TeX-Ressourcen, Catcode-Tabellen, Attribute, Callbacks sowie für das Laden und Erkennen von Lua-Modulen. Es enthält auch einige Werkezeuge zum Umgang mit Kompatibilitätsproblemen für älteren Versionen von LuaTeX.
Vorsicht. Es gibt derzeit einen Konflikt mit dem Paket luatex, da beide fast dasgleiche tun. Die jeweiligen Paketautoren wissen hierum und beabsichtigen, die Pakete in Zukunft zusammenzulegen. Sie wissen aber noch nicht, wann das sein wird.

luatex **Engine(s):** LuaTeX. **Formate:** LaTeX, Plain.
Autor(en): Heiko Oberdiek.
CTAN: `macros/latex/contrib/oberdiek/`.
Siehe die vorstehende Beschreibung zu luatexbase. Das Paket bietet dieselben Features, außer für die Verwaltung von Rückruffunktionen und die Erkennung von Lua-Modulen.

lualibs **Engine(s):** LuaTeX. **Formate:** Lua.
Autor(en): Élie Roux & Philipp Gesang.
CTAN: `macros/luatex/generic/lualibs/`.
Projekt-URL: `https://github.com/lualatex/lualibs`.
Sammlung von Lua-Bibliotheken und Ergänzungen zu den Standard-Bibliotheken, meist abgeleitet aus ConTeXt-Bibliotheken. Grundlegende Funktionen, die in Lua fehlen, könnten in diesem Paket enthalten sein.

2.2.4 Pakete zum Umgang mit Fonts

Diese Pakete werden von fontspec geladen, um einige Low-Level-Probleme mit Fonts und Kodierungen zu beheben. Der normale Anwender sollte nur fontspec benutzen, aber für einen Entwickler könnten die Kenntnis dieser Pakete nützlich sein.

luaotfload **Engine(s):** LuaTeX. **Formate:** LaTeX, Plain.
Autor(en): Élie Roux, Khaled Hosny & Philipp Gesang.
CTAN: `macros/luatex/generic/luaotfload/`.
Projekt-URL: `https://github.com/lualatex/luaotfload`.

Zum Laden von OpenType-Fonts (low-level), übernommen aus ConTEXt. Verwendet die Lua-Bibliothek `fontloader` und die entsprechenden Callbacks, um eine Syntax für die `\font`-Primitive bereitzustellen, die derjenigen von XETEX ähnlich ist, um die entsprechenden Font-Features zu implementieren. Verwaltet auch eine Fontdatenbank für den Zugriff auf Systemfonts und Fonts aus der TEX-Distribution gleichermaßen nach Fontfamilien- oder Dateinamen, sowie einen Fontcache, um den Ladevorgang zu beschleunigen.

euenc **Engine(s):** XETEX, LuaTEX. **Formate:** LATEX.
Autor(en): Will Robertson, Élie Roux & Khaled Hosny.
CTAN: `macros/latex/contrib/euenc/`.
Projekt-URL: `https://github.com/wspr/euenc`.
Implementiert die EUx-Unicode-Font-Kodierungen für das fontenc-System von LATEX. Aktuell verwendet XETEX `EU1`, während LuaTEX `EU2` benutzt. Einschließlich der `fd`-Dateien für Latin Modern, dem Standardfont, den fontspec lädt.

Genaugenommen enthält euenc nur die Kodedeklaration, enthält aber keine Definitionen für LICR-Makros; das wird über das Laden von xunicode mit `\UTFencname` (definiert in `EU1` oder `EU2`) durch fontspec erreicht. Die Definitionen selbst sind dieselben, es ist aber praktisch, sie unterschiedlich zu bezeichnen, um unterschiedliche `fd`-Dateien je nach der verwendeten Engine verwendet werden können (was bei Latin Modern tatsächlich geschieht).

3 Weitere Pakete

Die folgenden pakete sind in keiner bestimmten Reihenfolge aufgeführt.

3.1 Für Anwender

luatextra **Engine(s):** LuaTEX. **Formate:** LATEX.
Autor(en): Élie Roux & Manuel Pégourié-Gonnard.
CTAN: `macros/luatex/latex/luatextra/`.
Projekt-URL: `https://github.com/lualatex/luatextra`.
Lädt übliche Pakete, derzeit sind das: fontspec, luacode, metalogo (Befehle für Logos, einschließlich `\LuaTeX` und `\LuaLaTeX`), luatexbase, lualibs, fixltx2e (Bugfixes und Verbesserungen für den LATEX-Kernel).

luacode **Engine(s):** LuaTEX. **Formate:** LATEX.
Autor(en): Manuel Pégourié-Gonnard.
CTAN: `macros/luatex/latex/luacode/`.
Projekt-URL: `https://github.com/lualatex/luacode`.
Stellt Befehle und Makros bereit, die dabei behilflich sind, Lua-Code in eine TEX-Quelle einzubinden, insbesondere für Sonderzeichen.

luainputenc **Engine(s):** LuaTEX, XETEX, pdfTEX. **Formate:** LATEX.
Autor(en): Élie Roux & Manuel Pégourié-Gonnard.
CTAN: `macros/luatex/latex/luainputenc/`.
Projekt-URL: `https://github.com/lualatex/luainputenc`.

Ist dabei behilflich, Dokumente zu kompilieren, die in älteren Kodierungen voliegen (sowohl beim Quelltext als auch bei den Fonts). Wurde schon in der Einleitung vorgestellt. Bei XeTeX lädt das Paket nur xetex-inputenc; unter pdfTeX wird das normale inputenc verwendet.

luamplib **Engine(s):** LuaTeX. **Formate:** LaTeX, Plain.
Autor(en): Hans Hagen, Taco Hoewater & Philipp Gesang.
CTAN: `macros/luatex/generic/luamplib/`.
Projekt-URL: `https://github.com/lualatex/luamplib`.
Schönes Interface zur Lua-Bibliothek `mplib`, die MetaPost in LuaTeX einbettet.

luacolor **Engine(s):** LuaTeX. **Formate:** LaTeX.
Autor(en): Heiko Oberdiek.
CTAN: `macros/latex/contrib/oberdiek/`.
Ändert die Low-Level-Implementierung von Farben so, dass sie mit LuaTeX-Attributen verwendet werden kann. Dadurch wird die Implementierung robuster, es werden auch einige merkwürdige Bugs beseitigt, so etwa falsche Ausrichtung von Text, wenn `\color` zu Anfang einer `\vbox` verwendet wird.

3.2 Für Entwickler

pdftexcmds **Engine(s):** LuaTeX, pdfTeX, XeTeX. **Formate:** LaTeX, Plain.
Autor(en): Heiko Oberdiek.
CTAN: `macros/latex/contrib/oberdiek/`.
Obwohl LuaTeX überwiegend eine Obermenge von pdfTeX ist, wurden doch auch ein paar Primitiven entfernt (jene, die sozusagen von Lua übernommen worden sind) oder umbenannt. Dieses Paket stellt sie wieder bereit mit konsistenten Namen für alle Engines, einschließlich XeTeX, bei dem kürzlich einige dieser Primitiven eingebaut worden sind, etwa `\strcmp`.

magicnum **Engine(s):** LuaTeX, pdfTeX, XeTeX. **Formate:** LaTeX, Plain.
Autor(en): Heiko Oberdiek.
CTAN: `macros/latex/contrib/oberdiek/`.
Stellt hierarchischen Zugriff auf „magische Zahlen“ wie z. B. Catcodes, Group Types usw. zur Verfügung, die TeX und seine Nachfolger intern verwenden. Unter LuaTeX gibt es eine effizientere Implementierung und ein Lua-Interface.

lua-alt-getopt **Engine(s):** `texlua`. **Formate:** Kommandozeile.
Autor(en): Aleksey Cheusov.
CTAN: `support/lua/lua-alt-getopt`.
Projekt-URL: `https://github.com/LuaDist/alt-getopt`.
Ein Kommandozeilenparser, der größtenteils POSIX- und GNU-Getopt-kompatibel ist, um Lua-Skripten von der Kommandozeile aus zu verwenden, etwa `mkluatexfontdb` aus luaotfload.

4 Die Formate `luatex` und `lualatex`

Dieser Abschnitt wendet sich nur an Softwareentwickler und neugierige Anwender. „Normale“ Benutzer können ihn bei der Lektüre ohne weiteres überspringen. Die nachfolgenden

Angaben beziehen sich auf TEX Live 2010 und wahrscheinlich auch auf MikTEX 2.9, obwohl ich das nicht überprüfen kann. Frühere Versionen von TEX Live enthielten etwas andere und weniger vollständige Lösungen.

Primitiven Wie schon in Abschnitt 1.4 erwähnt, sind die LuaTEX-spezifischen Primitiven im `lualatex`-Format und im LuaTEX-Handbuch nicht dieselben. Im LuaTEX-Format (also in LuaTEX mit dem Plain-Format) stehen die Primitiven sowohl mit ihren normalen Namen zur Verfügung als auch mit ihren Entsprechungen mit der Präfix `lua`, so dass die Entwicklung generischer Pakete einfacher ist.

Dem liegt folgendes Prinzip zugrunde, das aus der Datei `lualatexiniconfig.tex` entnommen wurde, die dies alles für das `lualatex`-Format implementiert:

1. Alle aktuellen Makropakete laufen problemlos unter pdf(e)tex, daher bleiben ihre Primitiven unverändert.
2. Andere Non-TeX82-Primitiven können sich in LuaTEX mit Makros aus anderen Makropaketen wegen Namensgleichheit überschneiden, besonders wenn sie „natürliche“ Namen verwenden, wie etwa `\outputbox`, `\mathstyle` usw. So eine Möglichkeit ist unerwünscht, weil Namensgleichheiten unerwünscht sind, denn je mehr bestehende LATEX-Dokumente unter LuaTEX ohne Änderungen kompiliert werden können, umso besser.
3. Das LuaTEX-Team möchte kein systematisches Regelwerk für den Umgang mit Präfixe aufstellen, sondern hat stattdessen freundlicherweise ein Tool bereitgestellt, das auf Präfixe angewandt werden kann. Also haben wir dieses verwendet. Früher hatten wir sogar die Extra-Primitiven deaktiviert. Heute meinen wir dagegen, es wäre besser, sie mit einem systematischen Prfixing zu aktivieren, um zu vermeiden, dass jedes einzelne Makropaket (und jeder einzelne Benutzer) sie mit verschiedenen und inkonsistenten Präfixe verwendet (einschließlich einer leeren Präfix).
4. Die `luatex`-Präfix wurde gewählt, weil sie schon als Präfix in Gebrauch ist, etwa in `\luatexversion`: So kommt es bei diesen Primitiven auch nicht zu Doppelpräfices. (Einzelheiten können `tex.enableprimitives` im LuaTEX-Handbuch entnommen werden.
5. Die Primitive `directlua` steht sowohl mit ihrem „natürlichen“ Namen zur Verfügung (damit sie von LuaTEX einfach erkannt werden kann) als auch in einer präfizierten Fassung
6. The `\directlua` primitive is provided both with its natural name (allowing easy detection of LuaTEX) and a prefixed version `\luatexdirectlua` (zur Konsistenz mit `\luatexlatelua`).
7. Einige Anmerkungen:
   - Der offensichtliche Nachteil einer solchen Vorgehensweise ist, dass die Namen, die LATEX oder ein Autor, der generische Makros schreibt, verwenden, nicht mit den Namen übereinstimmen werden, die dem Handbuch zu entnehmen sind. Wir hoffen, dass dies durch die Rückwärtskompatibilität wieder aufgewogen wird.
   - Alle Primitiven, die mit Mathematik in Unicode arbeiten, beginnen mit `\U`. Vielleicht werden ihre Namen eines Tages dieselben sein wie bei XETEX, ggf. war das Prefixing für sie gar nicht notwendig oder gar nicht wünschenswert. Wir haben versucht, die zugrundeliegende Regel so einfach wie möglich zu machen, si dass der vorgenannte Punkt nicht noch schlimmer wird.
   - Vielleicht werden wir eines Tages zu dem Schluss kommen, dass es besser wäre, alle Primitiven ohne Präfixe zu führen. In diesem Fall wird es besser sein, die Primitiven ohne Präfixe dem Format hinzuzufügen und die Namen mit Präfixe aus Kompatibilitätsgründen

beizubehalten. Umgekehrt wäre das nicht möglich. Wenn wir nachträglich merken würden, dass wir Primitiven ohne Präfix vorteilhafter fänden, würden die zwischenzeitlich geschriebenen LuaTEX-spezifischen Pakete nicht mehr funktionieren.

\jobname Der LaTeX-Kernel (und viele weitere Pakete) verwenden aus verschiedenen Gründen Konstrukte wie `\input\jobname.aux`. Wenn aber `\jobname` Leerzeichen enthält, funktioniert der Befehl nicht, weil das Argument von `\input` schon am ersten Leerzeichen endet. Um dies zu umgehen, wird `\jobname` von pdfTEX automatisch mit Quotes maskiert – was LuaTEX aber aus gutem Grunde nicht tut. Die LaTeX-basierten Formate enthalten (anders als die Plain-basierten) einen fast vollständigen Workaround für dieses Problem.

Er funktioniert aber nicht, wenn LuaTEX als `lualatex '\input name'` anstelle des gebräuchlicheren `lualatex name` aufgerufen wird. Um diese Beschränkung, die dem Format innewohnt, zu umgehen, sollte ein Jobname ausdrücklich vergeben werden, wie in `lualatex jobname=name '\input name'`. Noch besser wäre es natürlich, keine Leerzeichen in den Namen seiner TEX-Dateien zu verwenden.

Weitere Einzelheiten können in den Diskussionen http://www.ntg.nl/pipermail/dev-luatex/2009-April/002549.html und http://tug.org/pipermail/luatex/2010-August/001986.html aus der LuaTEX-Mailingliste nachgelesen werden. Die Implementierung des Workarounds kann man in `lualatexquotejobname.tex` nachlesen.

Trennmuster LuaTEX lädt Trennmuster dynamisch. Die Unterstützung hierfür in babel und polyglossia gibt es zwar erst seit TEX Live 2013, sie sollte aber schon zuverlässig funktionieren.

Die Dokumentation und die Implementierung im einzelnen sind in der Datei `luatex-hyphen.pdf` enthalten. Die Quellen gehören zum TEX-Hyphen-Projekt: http://tug.org/tex-hyphen/.

Kodierungen Die Engine selbst setzt keine `\catcode`s, `\lccode`s usw. für Zeichen außerhalb des ASCII-Zeichensatzes. Vor allem korrekte `\lccode`s sind grundlegend für die Zeichentrennung. Die Formate für LuaTEX enthalten nun alle `luatex-unicode-letters.tex`, eine modifizierte Fassung von `unicode-letters.tex` aus XeTEX, die dafür sorgt, dass diese Werte gemäß dem Unicodestandard gesetzt werden.

Diese Ergänzung war nach der Veröffentlichung von TEX Live 2010 hinzugefügt worden. Wer mit Nicht-ASCII-Texten arbeitet, sollte also gegebenenfalls seine Installation auf den neuesten Stand bringen.

5 Was funktioniert, was teilweise funktioniert und was (noch) nicht funktioniert

5.1 Was funktioniert

Unicode Normales LaTeX kann schon recht gut mit UTF-8-kodierten Texten in Eingabedateien umgehen. Trotzdem bleibt es problematisch, dass Nicht-ASCII-Zeichen intern aus mehreren Teilen zusammengesetzt sind – TEXniker nennen diese Teile *Token*. Deshalb haben einige Pakete, die Text zeichenweise oder noch kleinteiliger verarbeiten (die etwa Catcodes verändern), immer

wieder Probleme, wenn sie in konventionellem LaTeX mit UTF-8-Kodierung zu tun bekommen. Beispielsweise kann man nicht jedes beliebige Nicht-ASCII-Zeichen für wortwörtliche Texte mit dem Paket fancyvrb setzen.

Die gute Nachricht ist, dass einige dieser Pakete mit LuaTeX ohne Einschränkung funktionieren, ohne dass man das Paket ändern müsste. Die schlechte Nachricht ist, dass das nicht immer wahr ist. Näheres ist dem folgenden Abschnitt zu entnehmen.

5.2 Was teilweise funktioniert

microtype Das Paket microtype verfügt über eine eingeschränkte LuaTeX-Unterstützung. Genauer gesagt, stehen seit Version 2.5 2013/03/13 optischer Randausgleich und Schriftstärkenveränderung (*font expansion*) zur Verfügung und sind im PDF-Modus standardmäßig aktiviert. Kerning, Spacing und Tracking fehlen aber weiterhin (vgl. Tabelle 1 in Abschnitt 3.1 von `microtype.pdf`).

Andererseits unterstützt das Paket luaotfload, das von fontspec geladen wird, viele mikrotypografische Features. Das einzige, was derzeit fehlt, ist also ein einheitliches Interface hierzu.

xunicode Die wichtigste Aufgabe des Pakets xunicode besteht darin sicherzustellen, dass die üblichen Escape-Sequenzen für Nicht-ASCII-Zeichen, also etwa `\'e`, in Unicode das richtige bewirken. Das Paket wird *wahrscheinlich* mit LuaTeX funktionieren, es prüft aber ausdrücklich nur auf XeTeX. fontspec verwendet einen Trick, um es trotzdem zu laden. Man kann es also nicht ausdrücklich laden, aber das ist auch nicht nötig, weil fontspec dies übernimmt.

Kodierungen Wie in dem vorhergehenden Abschnitt schon erwähnt, funktioniert einiges von dem, was unter UTF-8 mit konventionellem LaTeX problematisch war, mit LuaTeX– aber nicht alles. Zum Beispiel kann man mit dem Paket listings in LuaTeX nur Zeichen unter 256 innerhalb von Listings verwenden, also den Zeichensatz Latin-1 (darüber hinaus steht außerhalb von Listings natürlich der vollständige Unicode-Zeichensatz zur Verfügung).

Metriken Bei diesem Punkt geht es eigentlich nicht um „funktionieren“ oder „nicht funktionieren“, sondern eher ums „Nicht-Funktionieren“ wie im Fall von pdfTeX und XeTeX. Es kann nämlich zu geringfügigen Unterschieden bei Layout und Silbentrennung zwischen den Engines kommen.

Das kann daran liegen, dass die Engines für eine Sprache verschiedene Versionen desselben Fonts verwenden, dass die Silbentrennung anders funktioniert, dass bei Ligaturen oder beim Kerning andere Algorithmen zum Einsatz kommen (beispielsweise können jetzt das erste Wort eines Absatzes und Wörter, die aus verschiedenen Fonts gesetzt werden, getrennt werden) oder dass es Unterschiede bei den Trennmustern gibt (die Muster, die pdfTeX verwendet, sind eingefroren worden, aber LuaTeX und XeTeX verwenden für einige Sprachen neuere Versionen).

Wenn Sie eine größere Abweichung zwischen pdfTeX und LuaLaTeX mit demselben Font beobachten, kann es sich um einen Fehler in LuaTeX[10] oder in dem Font handeln. Bitte stellen Sie, wie sonst auch, sicher, dass Ihre Distribution auf dem neuesten Stand ist, bevor Sie einen Bugreport schreiben.

[10] Beispielsweise hatte LuaTeX 0.60 einen Bug, der die Silbentrennung nach einer –-Ligatur bis zum Ende des Absatzes verhinderte.

babel Funktioniert größtenteils ohne Probleme bei Sprachen, die mit lateinischen Buchstaben geschrieben werden. Bei anderen Sprachen kann das anders sein. Aber selbst bei lateinischen Buchstaben kann es zu Problemen bei der Kodierung kommen.

polyglossia Als ein moderneres, aber noch lückenhaftes Paket für den mehrsprachigen Satz, sollte polyglossia bevorzugt werden, auch wenn es noch nicht alle Features von babel enthält.

5.3 Was (noch) nicht funktioniert

Altkodierungen Pakete, die bei den Eingabe- oder bei den Ausgabekodierungen (Sourcen- bzw. Fontkodierungen) ansetzen, werden mit LuaTEX sehr wahrscheinlich nicht mehr funktionieren. Das betrifft die Pakete inputenc, fontenc, textcomp und wahrscheinlich auch die meisten klassischen Pakete für Schriften wie mathptmx oder fourier. Die gute Nachricht ist, dass Unicode sowieso ein modernerer Ansatz für die Probleme bei den Kodierungen ist, die die alten Pakete lösen sollten. Andererseits ist aber noch nicht alles in die schöne neue Welt von Unicode portiert worden, es kann also sein, dass man für eine gewisse Weile eine kleinere (oder einfach eine andere) Auswahl an verfügbaren Lösungen haben wird, vor allem für den Umgang mit Schriften.

Leerzeichen Leerzeichen in Dateinamen werden ganz allgemein in der TEX-Welt nicht besonders gut unterstützt. Das wird mit LuaTEX auch nicht besser. Besonders schlimm wird es, wenn die Haupt-TEX-Datei Leerzeichen enthält *und* LuaTEX nicht auf die übliche Weise aufgerufen wird. Wenn Sie LuaTEX auf die übliche Weise aufrufen, sollte alles funktionieren – ich werde Ihnen aber nicht erzählen, was bei einem unüblichen Aufruf herauskommt. Bitte lesen Sie im übrigen den Punkt über jobname in Abschnitt 4 wegen des dort genannten Workarounds und der technischen Einzelheiten. Oder, noch besser, verwenden Sie keine Leerzeichen in den Dateinamen Ihrer TEX-Dateien