Zpravodaj Československého sdružení uživatelů TeXu

Hans Hagen
LuaTeX version 1.0.0

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 28 (2018), No. 1-4, 38–42

Persistent URL: http://dml.cz/dmlcz/150105

LuaTEX version 1.0.0

Hans Hagen

After ten years of development, the first stable version of the LuaTEX engine was released at the 10th International ConTEXt Meeting 2016. The article describes the beginnings, the development, and the future of LuaTEX.

Keywords: Lua, LuaTEX, ConTEXt

The release

After some ten years of development and testing, on September 9, 2016, we released LuaTEX 1.0.0! It happened at the tenth meeting of the ConTEXt users and developers group in the Netherlands.

Instead of staying below one and ending up with versions like 0.99.1234, we decided that the moment was there to show the TEX audience that LuaTEX is stable enough to lose its beta status. Although functionality has evolved and sometimes been replaced, we have been using LuaTEX ourselves in production right from the start. Of course there are bugs and for sure we will fix them.

Our main objective was and still is to provide a variant of TEX that permits user extensions without the need to adapt the inner workings. We did add a

doi: 10.5300/2018-1-4/38

few things here and there but they mostly relate to opening up the inner parts and/or the wish to influence some hard-coded behaviour. Via Lua we managed to support modern functionality without bloating the code or adding more and more dependencies on foreign code. In the process, a stable and flexible MetaPost library became part of the engine.

The functionality as present now will stay. We might open up some more parts, we will stepwise clean up the code base while staying as close as possible to the Knuthian original, we will try to document bits and pieces. We might also experiment a bit with better isolation of the backend, and simplify some internals. For that we can use the experimental version but if we diverge too much we may need to give that another name.

We want to thank all those who have tested the betas and helped to make LuaTeX better.

Hans Hagen
Hartmut Henkel
Taco Hoekwater
Luigi Scarso

The past

Originally we planned to release the first version a few years ago but our ambitions didn't work out well with that schedule so we finally took a decade to get there. For the record it is good to summarize what happened during those years.

- Around 2005, after we talked a bit about extending TeX in a flexible way and Hartmut added the Lua scripting language to pdfTeX as an experiment. This add-on was inspired by the Lua extension to the SciTe editor that I (still) use.
- At that time one could query counter registers and box dimensions and print strings to the TeX input buffer.
- The Oriental TeX project then made it possible to go forward and come up with a complete interface. For this, Taco converted the code base from Pascal to C, quite an impressive effort.
- We spent more than a year intensively discussing, testing and implementing the interface between TeX and Lua. Many binaries and lots of test code were flying between Taco and my machine as we progressed and decided what directions to go. These were really interesting times.
- In successive years, we polished things and extended bits and pieces and in recent years we cleaned up interfaces, polished some code, filled in gaps and reached the point where we were more or less satisfied.
- The core is still traditional TeX, but has been extended with pdfTeX protrusion and expansion (reworked) and directional features from Aleph (cleaned

up). We did add some extensions (in ε-TEX fashion) but removed most of the ones that we inherited from pdfTEX because LUA could do better.

- The backend and extension interfaces are now mostly separated and although we don't expect to add more backends, it makes the code somewhat cleaner because all kinds of PDF-related issues are no longer mixed with front-end mechanisms.
- The font subsystem is no longer limited to 8-bit fonts. It must be noted that, for instance, OpenType support is done in LUA, which provides a lot of flexibility. This also serves as an example of extensibility. A small TEX core, independent of libraries, was definitely an objective and it works out well.
- The (rewritten but compatible) hyphenation machinery can use runtime loaded (and extended) patterns. There are a few extensions and, of course, one can revert to LUA for more.
- Already at an early stage, hyphenation, ligaturing and kerning were separated, which was one step in adding callbacks to nearly every stage in the typesetting process.
- Math supports wide (more than 8-bit) characters too so that one can implement Unicode math easily. The machinery has OpenType math code paths because there are some fundamental differences with traditional TEX math fonts.
- Although the kpse library is still the default interface to the file system, all in- and output can be controlled and intercepted, for instance for input filtering or re-encoding on the fly.
- The token scanner has been opened up so that one can write (simple) parsers. Experimental interception code didn't prove to be useful, so that interface has been dropped. We kept it simple and efficient.
- During callbacks related to the node lists, individual nodes can be accessed and manipulated at will. Of course, one needs to know a bit about the internals and not mess up the lists to the extent that TEX will choke on it: things that 'can't happen' now can. Most of the original documentation of the code by Don Knuth still applies (which was another objective) but of course directional support and such go beyond that. And it's surprisingly fast.
- Images and reusable boxes are now native nodes; they travel through the system as special kinds of rules instead of whatsits with dimensions. Users can define their own rule types too.

There is more to say but much has been reported already in articles in this and other journals. In the CONTEXT distribution, there are four documents describing aspects of the development and choices we have made (`mkiv.pdf`, `hybrid.pdf`, `about.pdf` and `still.pdf`) and we keep writing (`onandon.pdf`). One thing will hopefully be clear by now: the choice of LUA was a good one.

The future

The project is driven by ConTeXt users and ConTeXt development, which is why we found it proper to release version one at the tenth meeting. Right from the start, ConTeXt supported LuaTeX and this means that most mechanisms have been tested in production. There is some risk in this as users then are always forced to update the binary with the macros, but the ConTeXt garden provides easy ways to deal with this. In fact, most users switched to the new engine pretty soon after we started rewriting ConTeXt. We greatly appreciate their patience with us.

Raw performance of LuaTeX is of course less than 8-bit pdfTeX but in practice and on modern machines LuaTeX behaves well. In fact, many mechanisms, like native xml handling and MetaPost processing, are way faster in ConTeXt MkIV then in the now frozen MkII version. Given the fact that we're using Unicode and more complex fonts, one can safely assume that in ConTeXt, the overhead due to delegation to Lua has no real drawbacks.

We will continue development, but functionality will stay stable within versions. The code will be further streamlined and documented. We deliberately postponed some cleanup till after version one. And, of course, bugs will be fixed. We hope to stepwise improve the manual too. So what will the future bring?

- So far, we have managed to avoid extensions beyond those needed as part of the opening up. We stick close to Don Knuth's concepts so that existing documentation still conceptually applies. We keep our promise of not adding to the core. But we might open up (make configurable) some of the remaining hard-coded properties.
- Some node lists can use a bit of (non-critical) cleanup, for instance passive nodes, localpar nodes, and other leftovers. Maybe we should add missing left/right skips.
- We can optimize some callback resolution (more direct) so that we can gain a little performance.
- Inheritance of attributes needs checking and maybe we need to permit some more explicit settings.
- We will move some more code to the api file and plan to update the global pdf and Lua states consistently (there are some leftovers from the early days). Some C macros can probably go away.
- We can possibly minimize some return values of Lua functions and only return nil when we expect multiple calls in line. This might be more efficient. We plan to look into Lua 5.3 but we might well conclude that it's better to stick with 5.2.
- We have to figure out a way to deal with literals in virtual characters. This relates to font switching in the result.

- Maybe we will reorganize some code, so that documentation is easier. We hope to continue to stick close to what Don Knuth documents.
- We can clean up and isolate the backend a bit more. We could also add a few more options to delegate actions to Lua and we should get rid of some historic pdf artifacts.

Of course, we have some ideas of what to do next but these don't need an extension to the engine because we can use Lua for that.

In that perspective, it is tempting to think of a (lean and mean) LuaTEX variant for ConTEXt: one close to the traditional core with many hooks and a minimal number of dependencies on libraries and such. In a ConTEXt setup, a user only needs LuaTEX because all (workflow) related scripts are written in Lua and if additional functionality (like graphic conversions) is needed, it can easily be provided by external programs.

We will not touch the stable version unless it concerns bug fixes and/or simple extensions, but we will keep exposing ConTEXt users to the experimental branch (as we do now). Of course users of other macro packages can pick up binaries from the compile farm that has been set up by Mojca and friends.

So ... be prepared.

Verze 1.0.0 stroje LuaTEX

Po deseti letech vývoje byla na jubilejním desátém Mezinárodním setkání uživatelů ConTEXtu v roce 2016 vydána první stabilní verze TEXového stroje LuaTEX. Článek popisuje začátky, vývoj a budoucnost LuaTEXu.

Klíčová slova: Lua, LuaTEX, ConTEXt

Hans Hagen, pragma@wxs.nl

Poznámka redakce: Začátkem roku 2019 dojde k vydání verze 1.1.0 stroje LuaTEX. Tato verze bude součástí distribuce TEX Live 2019. Změny od verze 1.0.0 zahrnují přechod z verze 5.2 jazyka Lua na verzi 5.3, náhradu knihovny poppler pro čtení pdf dokumentů za knihovnu pplib a podporu procesorové architektury arm.