LuaTEX

closing in on 0.50

TUG 2009
Since we reported on the state of LuaTEX at KTUG, Dante and Bachotek no fundamental extensions have been made. The focus has been on a 0.40 freeze (for TEXlive), debugging, cleanup and converting the Pascal code into C as a prelude to 0.50 (euroTEX). We promised a production version. The 0.40 version that will be on TEXlive is quite stable and usable. Currently we're moving to 0.50.

The current version of LuaTEX is quite compatible with pdfTEX. Of course documents can come out differently because of the following reasons:

- We have Unicode hyphenation patterns independent of the font encoding.
- There is more granularity in metrics when not using traditional fonts.
- Hyphenation, ligaturing and kerning are separated stages.

But such differences occur anyway over time (as one updates resources) unless a frozen distribution is used.
Roadmap

The paragraph builder will become more accessible (easier to do now).

The output routine will be opened up.

Directional issues will be cleaned up (oriental T EX project).

The backend will be restructured (in progress).

We will explore and implement some pending wishes.

Of course we will do whatever comes to our mind.

It's up to you (and your package writers) to use it or not.
Luafication

One can ignore the opening up and just use Lua for manipulations, and generation and piping of data. In ConTEXt we make extensive use of the opening up. The development of LuaTEX and ConTEXt MkIV goes in sync.

We're quite lucky that ConTEXt users are testing new releases as soon as they show up and the contextgarden has binaries for many platforms.

At the upcoming ConTEXt meeting we hope will discuss the impact of what has been done so far and will be done after 0.50. Areas that have been touched are:

- all input and output
- most of fonts, especially OpenType
- many manipulations like character casing
- much of math (and more to come)
- everything structure and referencing
- color and other attributes
- the complete pdf backend
- a tree based xml fontend
- the usual tools (scripts)

Some of the changes are real large (state).
Usability

No matter what nice things we do, the big question is: can LuaTEX be used at all, given all the changes in the code base.

We ship some basic plain TeX support as part of the ConTeXt distribution as proof that you can use LuaTEX without ConTeXt as well, but of course we don't touch plain itself.

I do use an older version (0.30) in a few small and non-critical workflows, but especially the rewritten structure related code has quite some impact.

All rewrites go through several stages because we adapt LuaTeX when we feel it makes sense (beta features).

We measure usability in several ways:

- How fast is a simple run in an editor (startup, pdf building)?
- How well does it integrate in workflows (toolkits, overhead)?
- How fast is a run (or multiple runs when needed)?
- Some aspects are out of our control, like for instance the impact of the console on runtime (per char, refresh delay, fonts, buffer).
Let's take a closer look at some tests:

We constantly check changes with a couple of test documents: \texttt{mk}, \texttt{metafun}, \texttt{luatexref-t}. We only release when these documents process well.

I made a rather dumb baseline (mostly otr and backend) performance test (just the word \texttt{test} on many pages).
MK, the history of LuaTeX upto 0.50
<table>
<thead>
<tr>
<th>Engine</th>
<th>Dell M90 Laptop, Vista Ultimate / 32 engine</th>
<th>Dell 2950 Server, SuSe Linux 11 / 64 engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>xetex</td>
<td>1.81 (16) 2.45 (122) 6.97 (286) 29.20 (342)</td>
<td>0.92 (32) 1.89 (158) 8.74 (228) 42.19 (237)</td>
</tr>
<tr>
<td>pdftex</td>
<td>1.28 (23) 2.07 (144) 6.96 (287) 30.94 (323)</td>
<td>0.49 (61) 1.14 (262) 5.23 (382) 24.66 (405)</td>
</tr>
<tr>
<td>luatex</td>
<td>1.48 (20) 2.36 (127) 7.85 (254) 34.34 (291)</td>
<td>1.07 (27) 1.99 (150) 8.32 (240) 38.22 (261)</td>
</tr>
</tbody>
</table>

Baseline performance