# The State Of LuaTEX

Dante 2009

# Mission Statement

We're sort of halfway the development of LuaTEX now. This will be reflected in the release of version 0.50 around euroTEX.

Starting with version 0.30 all one decimal releases are stable and usable for (controlled) production work.

We currently have 0.35 (beta) which is a prelude to the 0.40 stable release for TEXLive.

Although interfaces might change till they're defined stable we expect to have a stable subset defined in 0.50.

We plan to release version 1.00 sometime in 2012, 30 years after TEX82, with 0.60 and 0.70 in 2010, 0.80 and 0.90 in 2011.

February 24, 2009

Taco Hoekwater
Hartmut Henkel
Hans Hagen

# Lua Scripting



Currently we can have multiple instances of Lua active. From there we can feed characters back to TEX either or not under a specific catcode regime.

Future releases might have only one instance as implementing an interface for sharing data across instances has no benefits. It also simplifies the code.

The deferred variant (late Lua) might be extended somewhat.

The interface will be stable in release 0.50.

# Input and Output

It is possible to bring most input under Lua control and one can overload the usual kpse mechanisms.

Logging etc. is also under Lua control. There is no support for writing to TEX's opened output channels except from logs and the terminal.

We're investigating limited write control to numbered channels but this has a very low priority.

There is a new mechanisms for managing catcodes although this is not really needed for practical usage as we aim at full compatibility.

# Interface to TEX

Registers can be accessed from Lua. Grouping is honoured. Internal registers can be accessed (mostly read-only).

Box registers can be manipulated but users need to be aware of potential memory management issues.

There will be provisions to use the primitives related to setting codes. Some of this will be available in version 0.50.

# Fonts

The internal font model has been extended to the full Unicode range. There are readers for OpenType, Type1, and traditional TEX fonts.

Users can create virtual fonts on the fly and have complete control over what goes into TEX. Font specific features can either be mapped onto the traditional ligature and kerning mechanisms or be implemented in Lua.

The interface is rather stable but some of the keys in loaded tables might change. The currently used code (taken from FontForge) will be stripped so that the LuaTEX gets smaller.

Most of the font interface will be stable in version 0.50.

# Tokens

It is possible to intercept tokenization. Once intercepted, a token table can be manipulated and either or not be piped into TEX.

Future versions of LuaTEX might use Lua's so called userdata concept but the interface will mostly be the same. Therefore this subsystem will not be frozen yet in version 0.50.

# Nodes

Users have access to the node lists in various stages. This interface is rather stable but some cleanup might take place.

We have plans for keeping more extensive information with a paragraph (initial whatsit) so that one can build alternative paragraph builders in Lua.

The basic interface will be stable in version 0.50.

# Attributes

This new kid on the block is now available for most subsystems but we might change some of its default behaviour. For instance negative values are not possible now but that will change (with reserved some bits for internal usage).

We also investigate a more efficient variant of attributes with limited possibilities but faster checking.

The basic principles will be stable around version 0.50, but (as with all) we take the freedom to change aspects till we reach version 1.00.

# Hyphenation

Managing patterns as well as hyphenation is reimplemented from scratch but uses the same principles as traditional TEX. Patterns can be loaded at runtime and exceptions are quite efficient now.

There are a few extensions, like embedded discretionaries in exceptions and pre- as well as posthyphens.

On the agenda is fixing some hyphenchar related issues.

Future releases might deal with compound words as well.

There are some known bugs and limitations that we hope to have solved in version 0.50.

# Images

Image handling is part of the backend. This is reimplemented from scratch and can be controlled from Lua. There are already a few more options than in pdfTEX (simple transformations).

The image code will also be integrated in the virtual font handler.

# Paragraph Building

The paragraph builder has been rewritten in C (soon to be webbed). There is a callback related to the builder so it is possible to overload the default linebreaker by one written in Lua.

There are no short-term goals on the agenda, apart from writing an advanced (third order) arabic routine for the Oriental TEX project.

Future releases may provide a bit more control over parshapes and multiple paragraph shapes.

# MetaPost

The closely related mplib project has resulted in a MetaPost library that is now included in LuaTEX. There can be multiple instances active at the same time.

Conversion to pdf is to be done with Lua. On the todo list is a bit more interoperability (pre- and postscript tables) and this will make it into release 0.40.

# Mathematics

Version 0.50 will have a stable version of Unicode math support. Math is backward compatible but provides solutions for dealing with OpenType math fonts.

We provide math lists in their intermediate form (noads) so that it is possible to manipulate math in great detail.

The relevant math parameters are reorganized according to what OpenType math provides (we use Cambria as reference). Parameters are grouped by (cramped) style. Future versions of LuaTEX will built upon this.

There are new primitives for placing accents (top and bottom variants and extensibles), creating radicals, and making delimiters. Math characters are permitted in text mode

There will be an additional alignment mechanisms analogue to what MathML provides.

# Page Building

Not much work has been done on opening up the page builder although we do have access to the intermediate lists.

This might happen before 0.50 (unlikely, apart from fixes).

# Going CWeb

After releasing version 0.50 around euroTEX 2009 there will be a period of relative silence. Apart from bugfixes and (private) experiments there will be no release for a while. The priority then is in converting the code base to cweb (this already done for the reimplemented components).

We hope that the TEXLive 2010 release will have the first full cweb version.

# Cleanup

Cleanup of code is a continuous process. Cleanup is needed because we deal with a merge of traditional TEX, eTEX extensions, pdfTEX functionality and some Omega (Aleph) code.

We also use the opportunity to get away from all the global variables that are used in the Pascal version. On the other hand, we keep using literate programming.

Compatibility is a prerequisite, with the exception of logging and rather special ligature reconstruction code.

# Alignments

**We do have some ideas about opening up alignments, but it has a low priority and it will not happen before the 0.50 release.**

# Error Handling

Once all code is converted to cweb, we will look into error handling and recovery. It has no high priority as it is easier to deal with after the conversion to cweb.

# Backend

The backend code will be rewritten stepwise. The image related code has already be redone, and currently everything related to positioning and directions is redesigned and made more consistent.

We are experimenting with positioning (preroll) and better literal injection (we currently use the somewhat messy pdfTEX methods).

Accuracy of the output (pdf) will be improved and character extension (hz) will be done more efficient.

This will be available in release 0.40 and further cleanup will take place when the cweb code is there.

# Leftovers

There is a bunch of small convenience issues that we deal with in the process of developing LuaTEX.