Just in time:

things we can do with LuaTEX only

TUG, June 28, 2010

- TEX can do a lot, often more than you can imagine. This is no surprise as it is a programming language and its interpreter has some hooks for extensions.
- We have been (and are) using TEX mostly for direct xml to pdf workfows and for manuals and fun using TEX input. But often regular TEX users are mode demanding than publishers.
- Till a couple of years ago, all that I wanted to achieve was somehow possible although not all solutions can win a programming beauty contest.
- Examples of challenging features are html like tables and page crossing backgrounds behind arbitrary text, both mechanisms are 10 years old now.
- Also, sometimes a bit freaky input encoding and font technologies stretch traditional TEX somewhat to its limits and related mechanism had become quite complex and interwoven.
- When you operate in the field of unattended automated typesetting (where performance has to be guaranteed) one needs tricks that are not really in traditional TEX's repertoire. And spending weeks programming something that should take days is no fun.
- This is why we started the LuaTEX project: combine the power of traditional TEX with the modern scripting language Lua and give it access to TEX's internals. And, as a bonus, integrate METAPOST.

The power of TEX

- Input encoding is no longer handled at the TEX end although we still support mixed encodings. Most users have switched to utf anyway.
- Support for (mixed) font encoding (in traditional TEX needed in order to mix languages) is gone, as are weird hacks due to 8 bit fonts (or encodings) lacking characters.
- There is no longer a relationship between the encoding of a font and hyphenation i.e. no more multiple instances of patterns. We already used patterns in utf for a long time anyway.
- As we are using Unicode, (mixed) math encodings have been replaced by Unicode math.
- We still use MkII as the (frozen) reference implementation and most major mechanisms will behave similar (multiple instances of features, inheritance, configurability, etc.) but some have been be reimplemented so that extending is easier.

We could thrash a couple of things

- All (multipass and runtime) data is managed in Lua which frees a lot of hash space and removes some bounds (structure, positioning, etc.)
- A couple of initializations is done at the Lua end using tables with metadata. Examples are characters and math.
- All structure related management issues are supported by Lua code. This has a large impact on sectioning, lists, descriptions, notes, itemization, synonyms, etc.
- Numbering of floats, formulas, pages etc, descriptions, enumerations, itemize and whatever needs numbers has been reimplemented and we carry status info around.
- Floats are partly managed by Lua (less code, same functionality) and at some point (re)placement will follow.
- Register (index) management is reimplemented and no external sorting scripts are needed (this used to be done by TEXexec).
- Case swapping, digit normalization, special language related kerning are all delegated to the engine using node list manipulations.
- Hyperlinks (ConTEXt always had a rather extensive model) are now parsed and managed at the Lua end and injection of the backend code is done by Lua.

Some changes are rather fundamental

- Numeric conversions are done in Lua.
- Graphic analysis uses the img library and file lookups as well as image database support are now implemented using Lua.
- Dealing with special breakpoints (related to hyphenation) is now done in a different stage by manipulating the node lists.
- Runtime METAPOST support is now dealt with by using mplib which reduced the overhead to nearly zero.
- Buffers are now kept in memory instead of in files.
- Verbatim now relies on Lua and pretty printing has been reimplemented (downward compatible, additional support is on the agenda).
- Experimental MkII features like multiple and parallel streams are being rewritten but in a MkIV way.

Existing mechanisms have been implemented in cleaner ways

We get a couple of things for free

- We have a nice language on board with powerful expression machinery. We provide some helper function and more will show up.
- The file (io) handler is not using kpse so that we can do the things we wanted to do 10 years ago (zip, schemes, more control, etc).
- One of the first things that has been implemented (also as an exercise) is an xml interpreter that replaces the existing TEX based MkII one. The new one operates on trees by using expressions and plugins. (Some core modules, like MathML has been rewritten in the process.)
- The runner script no longer is responsible for mechanisms that depend on sorting as we do this now internally. This removes a dependency. We are also removing the dependency on the bibTEX program.
- The ConTEXt related toolkit has been rewritten in Lua and as LuaTEX can operate as Lua interpreter we are independent of other languages and interpreters.
- We have extensive tracing options and debugging features that would have put too much burden at the TEX end. This also provides more insight in what happens inside.
- As less code is needed we moved support for chemical typesetting into the kernel (also as a prelude to math extensions). It also serves as example of mplib usage.
- We experiment with more advanced script support (interesting and fun to do) and due to the Oriental TEX project we started with Arabic.

- The (still somewhat) experimental code in MkIV will be cleaned up, generalized and completed.
- After this warming up we will start looking into more advanced trickery. Of course a further opening up will trigger new usage.
- Eventually all mechanisms will be evaluated, improved (or rewritten) and extended.
- There will be additional mechanisms for advanced font usage, language specific typesetting, multiple math environments (fields).
- Eventually ConTEXt will be more modularized so that we can make specialized (leaner and meaner) macro packages (a subproject tagged MetaTEX).
- Anything we can dream of.

We have lots of plans