

L^AT_EX3 News

Issue 4, July 2010 (L^AT_EX release 2010-07-01)

Now that we're back from the T_EX Users Group conference in San Francisco, it's time to discuss what's been going on over the last six months. Due to some extra travel plans after the conference, this issue is slightly late in coming out.

expl3 in practice

Joseph Wright and Will Robertson have both released significant new versions of their packages, resp., `siunitx` and `fontspec`. These have been re-written in the L^AT_EX3 programming language `expl3`, which we have discussed here previously. Using `expl3` for production code has been very successful, both in demonstrating that the concepts are sound and highlighting areas that still need some attention.

In the case of `fontspec`, `expl3` programming is being used to target L^AT_EX running on either X_YL^AT_EX and LuaT_EX. In the latter case, the package is a mixture of Lua code and `expl3` code; Will presented the `unicode-math` package at TUG 2010, which is developed in the same style.

New xpackages

Frank Mittelbach has started to work on a new experimental L^AT_EX3 package `xhead` that provides templates for one of the most complex areas of document design: section headings and document divisions. This is the beginning of an ambitious idea to map out the requirements for typesetting most documents currently processed with L^AT_EX.

One of the challenges here is providing a “natural” design language for describing the two-dimensional spatial relationships of objects participating in the design, e.g., the placement of a heading number in relation to the heading title, a possible sub-title, etc. In answer to this challenge Frank developed the `xcoffin` package, which he presented at TUG 2010. It is designed as a high-level interface for placing and aligning boxes on a page, allowing a ‘designer’s approach’ for indicating the positional relationship between boxes. (A ‘coffin’ is a box with handles.) As an example, it is possible to represent ideas such as ‘align the lower-left corner of box A with the upper-right corner of box B after rotating it ninety degrees’, without having to calculate the intermediate positions.

We expect a future version of `xcoffin` (after some further work on its interface layer and its internal imple-

mentation) to play a major role in all packages providing layout templates for higher-level document objects, such as table of contents designs, floats, etc. Finally, Joseph Wright has begun work with the current ‘galley’ packages, producing the new, minimal, `xgalley` based on `xfm-galley` as a testbed for what we need and what will work.

Developments with expl3

Meanwhile, Joseph’s *also* been writing a new floating-point calculation module, called `l3fp`, for `expl3`. This module allows manipulation and calculation of numbers with a much larger range than T_EX allows naturally. The `l3fp` module has already been utilised in the `xcoffin` code for calculations such as coordinate rotations and intersection points of vectors.

The modules `l3io` and `l3file` have been revised, rethinking the way that read and write streams are dealt with. T_EX has a hard limit of sixteen input and output streams open at any one time, and the new implementation for `expl3` provides more flexibility in how they are allocated; there’s now much less chance of running into a ‘No room for a new \read’ (or \write) error. Sometimes we discuss ideas for `expl3` that *don’t* end up making it into the final code. One example of this is the concept of having ‘local registers’ for integers, boxes, and so on, that do not survive outside of the group they are defined in (in contrast to Plain T_EX and L^AT_EX, where allocators such as `\newcount` and `\newbox` are always global). Despite the scope for some small benefit, we decided that the extra complexity that the additional functions required, in both syntax and documentation, was not justified.

TUG 2010 reflections

Our interpretation of the broad themes discussed at the conference are that T_EX-based systems are still thriving and there are some big problems to solve with robust solutions to transform L^AT_EX source, including mathematics, into a form such as HTML. While there are big pushes for standardising various aspects of the L^AT_EX syntax, we also believe that it is L^AT_EX’s very flexibility—its inherently non-standardised markup—that has allowed it to survive for so many years. There is a delicate trade-off here between moving forward into more standards-based territory while also retaining the extensibility of the third-party package system.