LaTeX 3 discussions at Pont-à-Mousson

2005-03-11

Present: J Bezos, D A Kastrup (part time), T Lotze, J L Braams, F Mittelbach, M Høgholm, R Fairbairns, B Bayart, C A Rowley

1 Architecture and terms

JLB outlined an architecture, showing a "document language" interface ("skins"), a "designer language" interface, and a "programmer language interface". There was discussion about whether there's more than one level between designer and programmer languages. For example, one might imagine an inner one where OR and float mechanisms reside.

FMi isolation of user from dirty internals.

```
document language::=''document structure markup''
designer interface::=''templates''
```
(mostly)

note templates are costly only at declaration time, instantiation is typically fast.

discussion of relation of syntaxes at user interface and at designer interface.

In the user's and the programmer's mind, there's an assertion "I want to do $\langle x \rangle$ with information $\langle y \rangle$"; is that compatible with a property list implementation?

outer layer is where the complicated parsing can be done. designer interface we potentially want highly reactive structure: should this be property list, or argument list? probably best to provide argument list, since that scopes the availability of the data (in a way difficult to achieve with property list, which can have real difficulties if a function recurses).

On the other hand, if every call of a \makebox-like function has to carry everything in its interface (language, direction, colour, ...), then we're going to hanker after property lists of "static" information.

positional arguments for the template

CAR: We need a written summary of discussion. (Ha!)

Agreed that property lists at the design language is a bad idea. either pass things down via templates, or via executable arguments.

So inner functions are positional, unless needs demonstrate some other requirement. We should investigate need for other requirements (e.g., property lists, implicit properties, ...).

"What hyperref interfaces?" remains to be discussed.

Colour support will presumably need to transcend levels, so needing several interfaces.

Formal approach to handling information for future runs would probably be useful. Problem with number of open files. No need for separate files for table-of-anything; BB has an implementation that writes a single file.

Need to look at x-packages to see the distribution of "level" within them: for example, xor

- aux file handling
- high-level IO
- maths
- lists
- templates
- environment stack
- tables, arrays
- cross-references
- colour
- "high quality" (tweak) mechanisms
- verbatim
- marks
- float placement
- front matter

Afternoon session

Present: J Bezos, T Lotze, F Mittelbach, M Høgholm, R Fairbairns, C A Rowley

2 Work to do

I: remove the l3in2e stuff from the context of LaTeX 2_ε: will need (notably) on an allocation module. (Johannes)
II: label mechanism. (Morten) including extensibility, both private (for things like hyperref) and public (for the general user).
III: passing information between runs. Note that there's an issue about what is passed between runs and is (in some sense) private, and what is (in some sense) intended to be exposed to the user.

Interaction with Heiko Oberdiek is necessary. We need to know how hyperref works. Perhaps someone just read the thing???

Hyperref currently takes a label as a "point" in the document. (Wild digression on what hyperref/acrobat reader actually do; no mention of what other readers do.)
IV: startup
V: xgalley: to provide a data structure for "vertically things" between commands, so as to avoid some of the anomalies that sometimes arise because of wotsits and the like intervening between an \advspace and its preceding vertical space.
VI: xor: lengthy discussion of what xor is currently capable of.