LaTeX3 architecture
and current work in progress

Frank Mittelbach
I presume this is one of several dozen bugs that would arise over the years if anyone were foolish enough to try allowing "_" in command names.

Leslie Lamport
A Timeline

- 1982 TeX2
  - 4 years later …

- 1986 LaTeX 2.09
  - 4 years later …

- 1990 TeX3
  - 4 years later …
  - 1991 expl3 (first attempt)
  - 1992 LaTeX3 architecture and kernel

- 1994 LaTeX2e
  - 5 years later …

- 1997 LuaTeX beta
  - 5 years later …

- 2012 LaTeX3 beta ???

? What happened with it?
The 1992/93 kernel

## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Primitive data structures and concepts</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>Conventions</td>
<td>8</td>
</tr>
<tr>
<td>1.2</td>
<td>Parameters</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Defining functions</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Defining new functions</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Defining internal functions (no checks)</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Control sequence names</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>Functions</td>
<td>12</td>
</tr>
<tr>
<td>3.2</td>
<td>Predicates and conditions</td>
<td>13</td>
</tr>
<tr>
<td>3.3</td>
<td>Internal functions</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Internal variables</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>LispX3 functions</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Expanding arguments of functions</td>
<td>14</td>
</tr>
<tr>
<td>4.2</td>
<td>Defining new variables</td>
<td>16</td>
</tr>
<tr>
<td>4.3</td>
<td>Manipulating the first argument</td>
<td>17</td>
</tr>
<tr>
<td>4.4</td>
<td>Manipulating two arguments</td>
<td>17</td>
</tr>
<tr>
<td>4.5</td>
<td>Manipulating three arguments</td>
<td>18</td>
</tr>
<tr>
<td>4.6</td>
<td>Internal functions and variables</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Tracing modules and functions</td>
<td>20</td>
</tr>
<tr>
<td>5.1</td>
<td>Functions</td>
<td>20</td>
</tr>
<tr>
<td>5.2</td>
<td>Internal variables</td>
<td>21</td>
</tr>
<tr>
<td>5.3</td>
<td>Tracing LispX3 functions</td>
<td>22</td>
</tr>
<tr>
<td>5.4</td>
<td>Dumping the current definitions</td>
<td>22</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Functions</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>Predicates and conditionals</td>
<td>28</td>
</tr>
<tr>
<td>6.1</td>
<td>Internal functions</td>
<td>28</td>
</tr>
<tr>
<td>6.2</td>
<td>Predicates</td>
<td>28</td>
</tr>
<tr>
<td>6.3</td>
<td>Variables</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>Checking functions and variables</td>
<td>25</td>
</tr>
<tr>
<td>7.1</td>
<td>Functions</td>
<td>25</td>
</tr>
<tr>
<td>7.2</td>
<td>Constants</td>
<td>26</td>
</tr>
<tr>
<td>7.3</td>
<td>Internal functions</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>Queries</td>
<td>27</td>
</tr>
<tr>
<td>8.1</td>
<td>Functions</td>
<td>27</td>
</tr>
<tr>
<td>8.2</td>
<td>Constants</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>Counters</td>
<td>28</td>
</tr>
<tr>
<td>9.1</td>
<td>Functions</td>
<td>28</td>
</tr>
<tr>
<td>9.2</td>
<td>Formatting a counter value</td>
<td>29</td>
</tr>
<tr>
<td>9.3</td>
<td>Variable and constants</td>
<td>30</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Internal functions</td>
<td>30</td>
</tr>
</tbody>
</table>
Reasons for Failure

- Right Architecture (ideas) — Wrong Time
  - Too radical
  - Too experimental
  - Too immature (and unexplained)
  - Far too slow and far too huge (when built on TeX)

- Burning Issues needed resolving first
  - “intermediate” version LaTeX2e was released
At the end of this message I attached the .log file for Frank's test file tparm4. This job crashes on 'pool size exceeded', for which I've been afraid since Frank sent his first proposal for the new kernel.

The new font selection scheme, the new macro-naming convention, the resource database, ... : they all eat truck-loads full of pool space!

I *do* like the interfaces of the modules I just mentioned, but I think this project is definitely going in the wrong direction:

it's nice but impractical!
Drivers (pains and objectives)

- No consistent design model
  - A few generic support commands, e.g., `\@startsection`
  - Limited flexibility, limited scope

- Most design changes required programming

- No (proper) management of logical or visual context
  - except for limited support of context for lists
  - Some hardwired context settings, e.g., footnotes in minipages
A Typesetting Example

described by
adjusted individually
rendered as

<quiz>
<title></title>
<question>
who was the forty-second president of the U.S.A.?
</question>
<answer>
william jefferson clinton
</answer>
<!-- Note: We need to add more questions later. -->
</quiz>

Academy Engraved LET: The quick brown fox jumped over the lazy dog.
ALGERIAN CONDENSED LET: THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
Avant Garde Mono ITC TT: The quick brown fox jumped over the lazy dog.
PORTAGOTIC TT: THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
Blackmoor LET: The quick brown fox jumped over the lazy dog.
RONSOFITC TT-Heavy: The quick brown fox jumped over the lazy dog.
ROMANASCITC TT: THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
Design Templates

Design A

Design B

Abstract Design

Generic Implementation

Underlying support language

offer

distill

provide
Design Templates

Design A

Design B

creates

Values A

Values B

select

select

has

Abstract Design

Parameters

define context

overwrite values in document
Functional Principles

- Clear separation: UI, design, coding
  - Supports reuse and flexibility
- Logical and visual context dependencies are managed
  - Needed for high-quality results

- UI supports formatting adjustments
  - Perhaps strongest point of LaTeX

- Comprehensive, orthogonal programming language
  - Now why would you need this?
Architecture overview
Nesting and sequencing of structural elements in the document defines “context”
Elements belong to one or more classes, e.g., “list” (generic), “itemize” (specific)
Encountered elements update the context and applicable rules are carried out

Notation:

- `<list` start of environment of class “list”
- `itemize>` end of environment of class “itemize”
- `<note>` completed environment of class “note“
- `!head` element of class “head”
- `*` loose nesting
- `.` tight nesting
- `!head<list` sequencing
## LDB – examples

<table>
<thead>
<tr>
<th>Context</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>!head&lt;list</td>
<td>List immediately follows a heading</td>
</tr>
<tr>
<td>&lt;list*&lt;list</td>
<td>List nested within list</td>
</tr>
<tr>
<td>&lt;list*&lt;itemize</td>
<td>An “itemize” nested within some list</td>
</tr>
<tr>
<td>list&gt;&lt;itemize</td>
<td>An “itemize“ starts immediately after a list has ended</td>
</tr>
<tr>
<td>&lt;float*&lt;caption&gt;*&lt;caption</td>
<td>Second “caption” environment within a float</td>
</tr>
</tbody>
</table>

Live demo
Nothing in this architecture really requires TeX as a formatting engine.

But …

With today’s advances in the processing power of the underlying engine the ideas now appear to be feasible (in “x”TeX).
2011 Activities

- Big Bang (documentation and code cleanup)
  - much more consistent documentation
  - clarifying which functions are expandable or not
  - restructuring code into \texttt{l3kernel}, \texttt{l3packages}, \texttt{l3experimental}
    and \texttt{l3trial}

- significant speed improvements in the kernel
  - faster \texttt{prg\_return}....: conditional code, faster \texttt{seq} and \texttt{prop}
  - by a factor at least 3 on sample documents (e.g., \texttt{siunitx})

- expl3 now mostly stable
  - for those parts that have been moved to \texttt{l3kernel}

- GitHub mirror

- work on stand-alone kernel
2011 Activities (cont.)

- modules `l3str` and `l3regex` currently written
  - possible extensions to code highlighting

- module `l3fp` being reimplemented
  - fast expandable IEEE-854 compliant decimal floating point arithmetic and expression parsing

- module `xcoffins / l3coffins`

- module `xgalley`

- initial work on font support (xfss)
  - first task converting the highly optimized NFSS to less optimal but more readable code

- reinitiate work on LDB
Properly integrate template and LDB

Define a mechanism to overwrite template instance values on document level

Define standard environment management

Finish galley mechanism

Rework output routine concepts
Introducing new concepts and ideas ...

There is nothing more difficult to take in hand, more perilous to conduct or more uncertain in its success than to take the lead in the introduction of a new order of things.

Machiavelli
Questions?