LATEX 3 project

Frank Mittelbach

Stockholm
18 November 1991

LATEX 2.09 ↪ LATEX3

Frank Mittelbach
Electronic Data Systems
Federal Republic of Germany

Asked to speak about LATEX2.09 → LATEX3 I will try to give a you a picture of the history, the current state, and the future of the LATEX3 project.

1. Historical Remarks

Whenever the future is somewhat unpredictable it seems wise to take a look into history—to find out what is already achieved and what remains to be tackled.

From the history of the LATEX3 project we will first take a look at the growing bulk of syntax descriptions and (partial) implementations that are the results of three years work.

Milestones:
Syntax and implementation

1988
- Some bug fixes send to Dr. Lamport
- Four page sketch of NFSS

1989
- First implementation of NFSS
- Implementation of `amstex.sty`

1990
- New tabular implementation by D. Duchier
- First attribute prototype (thrown away)
- First kernel prototype
- First recovery/help prototype

1991
- Second kernel prototype
- Sketches for style designer interface
- Second description of the attribute concept
- Extended description of the help facility
- Syntax for extended NFSS
- Third kernel prototype
- Release of LATEX 2.09 international with NFSS support

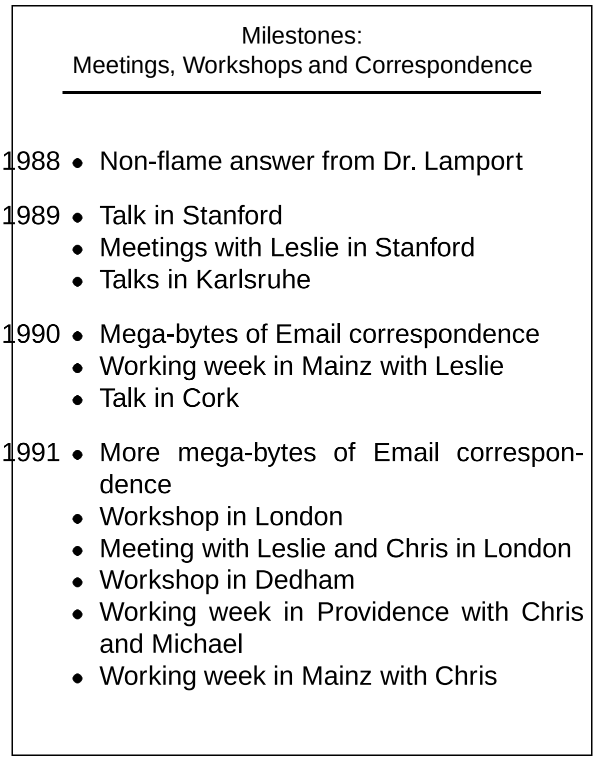
The LATEX3 project was initiated at the Stanford annual meeting in 1989. But first vague plans were already formulated in 1988 when Rainer Schöpf and I, after sending several pages of bug fixes for LATEX 2.09 to Leslie Lamport, received a positive answer.

Given the original goals for a reimplementation described in the Stanford paper, nearly everything seems to be achieved.

- With NFSS there is a far more general font selection available. The extended syntax also provides for font scaling (prototype implementation done).
- With `amstex.sty` the mathematical capabilities of LATEX have reached the standard of AMSTEX.
- With the new tabular implementation by Denys Duchier (that superseeds `array.sty`) and valuable suggestions by several others tabular processing has reached very high quality.
- With the new help/recovery concepts a safe and easy to learn environment is available for novice users.

- With the partially finished concept for specifying attributes to environments and functions a more flexible input language is available. This also allows easy conversion from SGML to LATEX3 DTDs.

How did this happen?



Milestones:
Meetings, Workshops and Correspondence

1988
- Non-flame answer from Dr. Lamport

1989
- Talk in Stanford
- Meetings with Leslie in Stanford
- Talks in Karlsruhe

1990
- Mega-bytes of Email correspondence
- Working week in Mainz with Leslie
- Talk in Cork

1991
- More mega-bytes of Email correspondence
- Workshop in London
- Meeting with Leslie and Chris in London
- Workshop in Dedham
- Working week in Providence with Chris and Michael
- Working week in Mainz with Chris

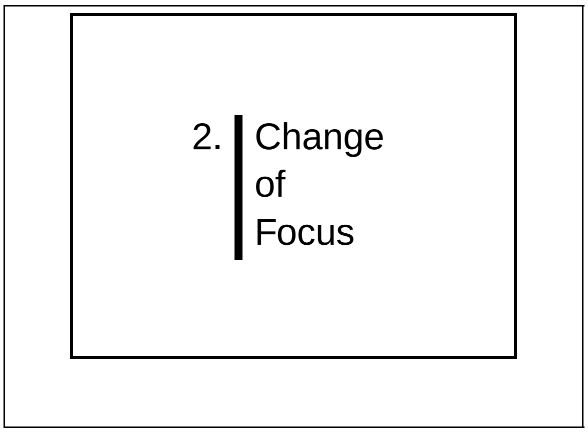
All this work has been carried out in the free time of several individuals and involves, as you can see, some enthusiasm to keep the project alive. So far, more than thirty people have contributed in one way or the other.

One of the major problems is to bring people together to discuss the open questions and find new solutions. This must also involve people outside the project since we do need the opinion and experience of typesetters, publishers, etc. to eliminate the flaws in the system and find new and better solutions.

In this regard both the London and the Dedham workshop have been a great success but further workshops of this kind are definitely necessary to provide LATEX3 with a suitable designer interface.

So why is this project in our opinion still being at its start? Because we have learned that our original goals haven't touched the real problems as we see them today.

This has lead to a

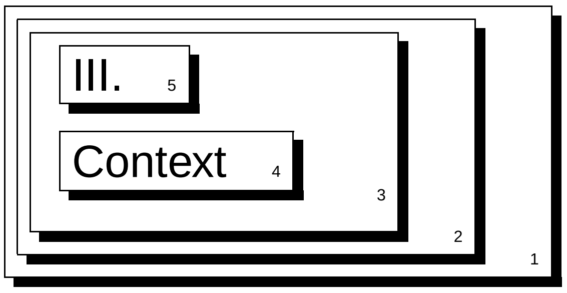


2. Change of Focus

We now feel that one doesn't gain much by providing more and more specialized style files that solve this or that special problem. Instead, we think that the major effort in the future has to go into the design of a suitable style interface that allow easy implementation of various layouts. (Easy, of course, is relative: easy compared to the complexity of the task.)

This change of focus implies
- The development of a new internal language that is more suited to express visual components of the layout process.
- The development of high-level generic functions that allow express most commonly used layout components in an easy way.
- The development of a model for specifying and modifying parameters that influence the layout.

Since the syntax for this internal language is still changing on a daily basis and generic functions mostly depend on it, I like to concentrate today on the model for parameter setting.



1. document context
2. section context
3. heading context
4. title context
5. number context

The slide already shows our main idea for maintaining parameters in the LATEX3 system.

On every point in the document we are in some context that is given by the the nesting and sequencing of entities processed so far.

The major idea of the new system is to allow the specification of parameters within such contexts in a very general way. For example, it is possible to redefine the behavior of lists within footnotes by specifying the values of list-parameters in the context of “footnotes” differently from those applied in the context of, say “floats”.

Parameter is meant in a very general way, e.g., the code that some entity runs is internally a parameter, so that via this concept different generic functions can be run in different contexts.

The concept of a context
Some observations

- The nesting of entities forms the major component for describing layout via contexts.
- The specification of layout by a sequence of contexts is important.
- The context of some entity in a document is not simply given by nesting and sequencing of surrounding entities.
- The context of some entity has a logical and a visual component. The visual component depends on the formatting of other entities.

As a further example, the layout attributes for a table entity in a float may be different then for tables in the main text.

Sequencing is, for example, important in heading→heading situations where intermediate spacing and penalties change if headings follow directly after each other, in list→text/par situations and many other places.

Take, for example, the situation of some footnote or float that appears in a list and itself contains a list. Because of an improper handling of contexts in the current LaTeX the inner list is typeset as a second level list. In other words an entity must be able to (partially) forget about its context, or more generally must be able to manipulate its context.

The fourth point is of theoretical nature. All of the currently available formatters format document entities in a predetermined visual context, i.e., they assume that the visual context can be determined by the logical nesting and sequencing of entities. To a certain extend TeX is an exception as it applies dynamic programming to the process of paragraph formatting which involve recomputation of contexts for ligatures etc.

As an example for the wrong visual context consider a hyphen at the end of a page, that is avoided by TeX by moving the line instead of recomputing the paragraph.

The concept of a context
Some problems

- The user input is not at normalized document—it may contain hidden entities inside of user-defined shorthands that can not be prescanned easily.
- The specification of contexts by sequencing is important but partially restricted by the underlying TeX engine.
- Taking the visual component of contexts into account requires the use of a multi-pass system.

The problem here is that we don’t deal with normalized documents (where every entity is fully tagged) and therefore can not scan for further begin or end tags before we start typesetting. This means that certain decisions have to be taken without knowing what follows. The only solutions to this problem are

- dissallow the use of user defined shorthands
- the use of a two-pass system that normalize the document in the first pass
- the use of a multi-pass system that use sequencing information from the last run.

None of the solutions seem to be feasible for a system that uses TeX as the input language but should be explored further.

Deferring of the typesetting process is generally possible in TeX’s vertical mode where we can wait for the next `\everypar` to regain control.

But TeX has no build in mechanism to detect whether plain character material after a given point is about to be contributed to some horizontal list. Only after material has been contributed to the horizontal list one can deduce this fact by “dirty tricks” with special kerns. But this can be used only for interrupting a context sequence—the contributed material can not be manipulated further.

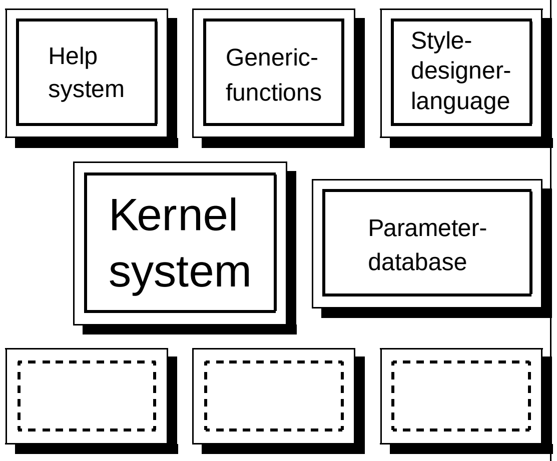
This will draw the boundary (beside processing time) between the ideal model and the real world. So let us now turn to the question of how this context-model is placed within the LaTeX3 system.

4. The Structure of the LATEX3 System

There are many interwoven structures of the LATEX3 system that are worth talking about. In the following I will show how the already achieved results and the ideas about contexts fit together in an extensible modular system.

The LATEX3 system will consist of a kernel system that provides the basic data structures such as lists, stacks, etc. to program higher modules. It will also contain arithmetic functions for integers and dimensions, e.g., it will be possible to express relationships between individual parameters by specifying assignments that contain expressions.

The Structure of the LATEX3 System
Modules



On top of it manipulation functions for the parameter database and generic functions are build. They form the platform for the style designer language.

One important component of the system will be an interactive help system that allows extensive help texts as well as the possibility to define system reaction depending on user action. Help messages and such additional error-correcting code will be held in external files that are read in when an error is detected by the system. In this way an elaborate help and error correcting mechanism will be available while keeping the LATEX3 kernel compact.

We distinguish between document styles that are written in the style designer language (and will probably contain nearly no TEX code in the traditional sense) and additional modules that provide entities for specialized documents. This will include, for example, higher math and we hope that we can provide with the new kernel a programming interface that makes the development of further modules a possible task.

Let me close with this quotation from some unknown novice of LATEX.

The new generation of LATEX users?

"Dear Sir,
I have successfully installed LATEX from the distribution—the file `sample` was just printed. However, somewhere in the README files a similar program called TEX is mentioned. Could you please explain to me how to install this program? . . . "

From a phone call received.

This might sound funny at first, but this extreme is not far from reality.

- Today, there are not many users who have a well-known understanding of the underlying system structure.
- Today, the majority of users use LATEX only. They usually have no knowledge of the TEXbook. This class of users can be nicely classified as "has heard of 'macros', but has never seen one".

TEX and LATEX as its major front-end has to compete with the so called Desktop-Publishing systems. To keep them alive we have to bridge the gap between the "implementor/wizard" type of user of the '80s and the new type that uses the system just as one tool out of manys without understanding its internals.

With the LATEX3 project we hope to achieve this goal as far as the front end is concerned. Our current eMails addresses are:

Frank Mittelbach:
`Mittelbach@mzdmza.zdv.Uni-Mainz.de`
Chris Rowley: `CA_Rowley@vax.acs.open.ac.uk`
Rainer Schöpf: `Schoepf@sc.ZIB-Berlin.de`