New Interfaces for LaTeX Class Design: Parts I and II

Frank Mittelbach
LaTeX3 Project
frank.mittelbach@latex-project.org

David Carlisle
LaTeX3 Project
david.carlisle@latex-project.org

Chris Rowley
LaTeX3 Project and Open University, UK
chris.rowley@latex-project.org

Introduction

Traditional LaTeX class files typically implement one fixed design via ad hoc, and often low-level, (La)TeX code. This style of implementation makes it much harder than is either desirable or necessary to produce classes that implement a specific visual design. Moreover, the construction of such classes typically involves a lot of work that is essentially programming and thus does not live easily with the declarative kind of design specification for a document (or range of documents) that would be produced by a professional typographic designer.[1]

This work introduces some extensions to LaTeX that will help to provide a new, more declarative interface that can be used in class files. It is based on the idea of a *template*, which describes how to carry out some action but which provides some flexibility since its code uses the values of a set of named (keyword) parameters. The specific design for this action, as required for a particular class, is then selected by choosing values for the template's named parameters.

Plans

The plan is to provide standard *templates* for a wide range of typographic objects but, of course, new templates for new ideas can be created, possibly by adapting an existing one or by a little LaTeX programming. It is our firm belief that there will soon be a large range of templates available and that it will thus be possible for the majority of class files to be implemented in a declarative way, by simply choosing suitable templates and supplying values for their named parameters.

Spin-off technology

Whilst applying the idea of templates to document design in LaTeX we have had the opportunity to substantially rethink many of the basic concepts of LaTeX's formatting machinery. This has led to the development of major enhancements in the following parts of LaTeX.

Paragraphs: There will be a completely new model and design interface for all aspects of paragraph-making, including: the parameters that control TeX's hyphenation and justification system; special typographical treatment of the beginning and end of paragraphs, e.g., initial letters/words (lettrines), nested run-on headings, etc.

Galleys: The paragraph model will be linked to a new model for the construction of galleys from paragraphs and other material; this model will incorporate current standard LaTeX concepts such as logical labels, marks and colour-change nodes, together with more experimental objects such as hyper-information nodes.

Floats: These elements have undergone a major redevelopment:

Captions: The formatting and positioning of the caption can be decided individually for each float and can depend on exactly where on the spread it appears.

[1] [This is an up-to-the-minute report from the LaTeX3 team; a written version will follow in a future issue of *TUGboat*. – Ed.]

Position: The position specification allows changes if the float does not fit on the current page.

Pages: More flexibility and better specification of both individual float pages and sequences of float pages (e.g., at chapter ends), including more possibilities to choose whether a text page or float page should be used.

Margins: Better integration of floats and marginal material, allowing floats to appear in margins and for text to be changed according to which margin is used.

Alignment: Better support for alignment between 'minipages', specified using logical handles such as 'top centre', 'centre left' or 'first baseline right': the relative positioning of two boxes (with such handles) is done by choosing a handle on each box and the (2-D) offset between these handles.

Page layout: More types of pages and spreads within a document; more layout choices and parameterisations.

Document commands: New tools for providing document-level syntax.

Professional support: For editorial and page make-up processes currently used in the publishing industry: e.g., flexible manual control over positioning of floats, etc. in the final form document; automated handling of complex bibliographic information in the front-matter of journal articles.

These are all, of course, still severely limited by what is practical within current TeX; for example, precise control over page-breaking within paragraphs is simply unobtainable using TeX's standard mechanisms. Nevertheless, we hope that what we have been able to do will inspire others to use the tools we provide in the creation and use of high-quality typographic designs.

Presentations

The two presentations explain these concepts and show examples of their use, covering both the current standard LaTeX designs and some more exciting new possibilities.

We demonstrate working examples of the application of these ideas in most of the major areas of document design, including page layout, section headings, lists and captions.

We also report on the current status of this work and on our plans to complete and publish it.

Post-Conference Update

The slides from the TUG'99 presentation of the talk we gave on a new interface for LaTeX class designers are available from the LaTeX Project website; look for the file `tug99.pdf` at:

`http://www.latex-project.org/talks/`

The slides are also available from the TUG99 website

`http://www.tug.org/TUG99-web/pdf`

in:

`carlisle.pdf`,
`mittelbach.pdf`, or
`rowley2.pdf`.

(All three have the same contents.)

Please note that the accompanying notes were only intended to be informal "speaker's notes" for our own use. We decided to make them available (the speaker's notes as well as the slides that were presented) because several people requested copies after the talk. However, they are *not* in a polished copy-edited form and are not intended for publication.

Prototype implementations of parts of this interface are now available from:

`http://www.latex-project.org/code/experimental/`

We are continuing to add new material at this location so as to stimulate further discussion of the underlying concepts. As of December 20, 1999 the following parts can be downloaded.[2]

xparse This module contains the prototype implementation of the interface for declaring document command syntax. It supports the definition of user commands with a LaTeX 2_ε interface including star forms, optional arguments, and picture mode arguments. See the `.dtx` files for documentation. It is possible to use this interface independently from all other modules described below.

template This module contains the prototype implementation of the template interface. Together with `xparse` it forms the basis of all further modules, i.e., to make use of any of the other modules you need both.

The file `template.dtx` in that directory has a large section of documentation at the front describing the commands in the interface and

[2] Please remember that this material is intended only for experimentation and comments; thus any aspect of it, e.g., the user interface or the functionality, may change and, in fact, is very likely to change. For this reason it is explicitly forbidden to place this material on CD-ROM distributions or public servers.

giving a 'worked example' to build up some templates for caption formatting.

xcontents This module contains the interface description for table of contents data, describing both an extended data model to replace the data deposited by heading commands into a `.toc` file and a number of template type descriptions for manipulating such data. There is currently no code provided to produce specially formatted table of contents; however, usable examples for the templates have been thoroughly discussed on the `latex-l` list, which can be retrieved as explained below.

xfootnote This module contains documented working examples for generating footnotes, etc. It implements some of the functionality of the package `footmisc` by Robin Fairbains and provides, although still incomplete, a good introduction to the usefulness of templates in class design. Due to the fact that support modules for paragraph manipulation, etc. are still under development, most of the implementation should be considered as a first draft only.

Modules currently under development include the following. Some if not all might be publicly available by the time you read this issue of *TUGboat*.

galley2 This module will document a new data structure for galley-related information, i.e., a data structure to better support handling of inter-paragraph material. Beside implementing lower-level programmer mutator functions for this data structure it will provide higher-level templates for declaring paragraph shapes and H&J (hyphenation & justification) specs. Most other modules will depend on its services as paragraph handling is needed for most aspects of layout. The code of the second prototype implementation for the galley data structure is 90% finished and it is targeted for a release date in 1999.

xlists This module documents and implements templates for providing various kinds of list structures. As part of the included examples it will contain a full set of LaTeX 2ε lists compatible in design to the `article` class implemented as instances of the templates provided. The prototype code for this module is finished but requires services from `galley2`.

xinitials This module implements a template for paragraph initials. An example of its use can be admired in the slides of our talk. Since it too relies quite heavily on `galley2` it is not yet released.

All examples are organised in subdirectories and additionally available as `gzip tar` files.

These concepts, as well as their implementation, are under discussion on the list `LATEX-L`. You can join this list, which is intended solely for discussing ideas and concepts for future versions of LaTeX, by sending mail to

`listserv@URZ.UNI-HEIDELBERG.DE`

containing the line

`SUBSCRIBE LATEX-L` *Your Name*

This list is archived and, after subscription, you can retrieve older posts to it by sending mail to the above address, containing a command such as:

`GET LATEX-L LOGyymm`

where `yy`=Year and `mm`=Month, e.g.,

`GET LATEX-L LOG9910`

for all messages sent in October 1999.[3]

[3] No, we don't know whether or not the listserv software is Y2K-compliant.